October 30, 1985

SFG05 Music BIOS

Reference Manual

v2.0

September, 11, 1985

Nippon Gakki Co.,
Software Development

October 30, 1985

INTRODUCTION

This reference manual is intended to explain the utilization of the "SFG-05" Control Program version 2.0" which resides within the internal 32Kbyte ROM of the "SFG-05".
All the explanations for the chord-key function, auto-bass function, and auto-rhythm functions are eliminated in this manual.
If these functions are to be used, refer to SFG-01 MBIOS manual.

TABLE OF CONTENTS

Chapter IV M-BIOS Syntax

Appendix

SFG-0: MBIOS
Supplementary Reference

October 30, 1985

CHAPTER I Outline

1-1 Program Configuration

The basic configuration of this program (SFG-05 Control Program) follows that depicted in the following figure.

The M-BIOS (Music Bios) controls the hardware of the SFG-05.
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-05.
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) Loading/saving of voice/automatic performance data onto CMT(Cassette Magnetic Tape recorder)
3) BDOS call entry(with the provision of IRQ handling and slot management)

In SFG-05, MBIOS functions can be invoked under three operating modes. They are operating mode 1.0, 1.1, and 2.0.

Operating mode 1.0 is compatible to MBIOS of SFG-01. That is, all the MBIOS functions work using fixed address working area under the interrupt mode 1. Bugs found in SFG-01 are now corrected in operating mode 1.0.

In operating mode1.1, the base address of MBIOS working area can be allocated anywhere in available RAM. This enables MBIOS to run under disk environment. However, in the operating mode 1.1, interrupt mode of Z80 is still assumed to be under mode1.

In operating mode 2.0, the base address to MBIOS work is not only relocatable, but interrupt mode is also assumed to be under mode2. This allows disk access and the fast interrupt handling of MIDI

caused by SFG-05 cartridge. For fast MIDI data handling, FIFO buffer handling for transmitter and receiver function is now added to the operating mode 2.0.

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".
The M-Monitor can be invoked either from the operating mode 1.0 or from the operating mode 2.0, respectively from the different entry.
If called as M-Monitor 1.0, it will not support disk functions.

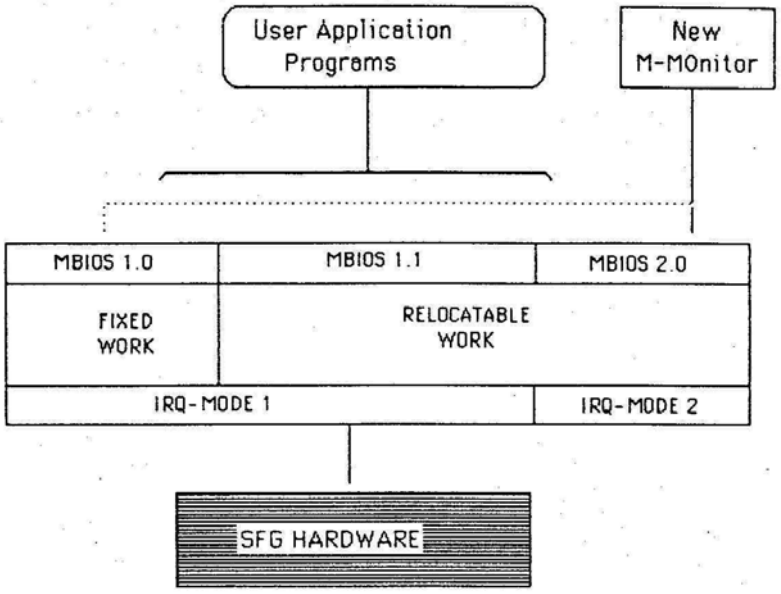


FIG 1.1 MBIOS CONFIGURATION

1-3 Hardware Configuration

The configuration of the hardware is shown in the following figure.

The MSX main unit and the SFG-05 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-05.
Additionally, the right channel of the SFG-05 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.

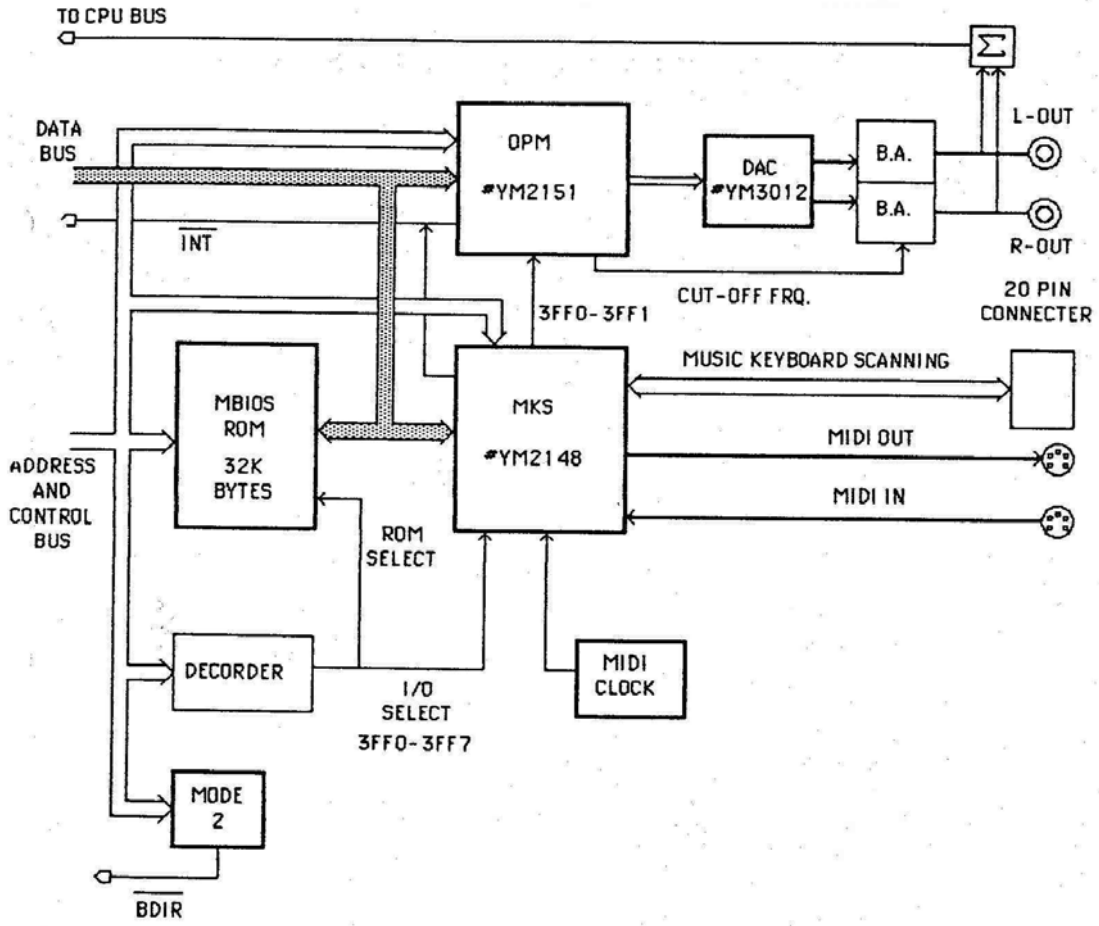


FIG. 1.2. HARDWARE CONFIGURATION OF SFG-05

1-4 Interface With MSX-BASIC

[Header]

The first 16 bytes starting from Address 4000h of MBIOS is a header to enable the MSX-BASIC to jump to the Music-Monitor of MBIOS.

4000h (ID)	"AB"
4002h (INIT)	Address of RETURN instruction
4004h (STATEMENT)	Address of "CALL MUSIC"
4006h (DEVICE)	0000h
4008h	0000h
400Ah and up	0000h

[VDP reseting]

During CSM vocal synthesis and in the operating mode 2.0, IRQ from VDP is reset directly.

[PLAY queue-buffer usage]

MBIOS uses PLAY-queue buffer area as an indirect addressing pointer buffer.
Therefore when MBIOS is active, MBIOS alters the PLAY-hook so as the PLAY-queue buffer not to be referenced accidentally by BASIC's PLAY routine.

PLAY-queue buffer usage is:

MBIOS	----	F975h - F9F4h
M-MONITOR	----	F9F5h - FAF4h

1-4 Interface With MSX-BASIC

[Header]

The first 16 bytes starting from Address 4000h of MBIOS is a header to enable the MSX-BASIC to jump to the Music-Monitor of MBIOS.

4000h (ID)	"AB"
4002h (INIT)	Address of RETURN instruction
4004h (STATEMENT)	Address of "CALL MUSIC"
4006h (DEVICE)	0000h
4008h	0000h
400Ah and up	0000h

[VDP reseting]

During CSM vocal synthesis and in the operating mode 2.0, IRQ from VDP is reset directly.

[PLAY queue-buffer usage]

MBIOS uses PLAY-queue buffer area as an indirect addressing pointer buffer.
Therefore when MBIOS is active, MBIOS alters the PLAY-hook so as the PLAY-queue buffer not to be referenced accidentally by BASIC's PLAY routine.

PLAY-queue buffer usage is:

MBIOS	----	F975h - F9F4h
M-MONITOR	----	F9F5h - FAF4h

1-5 Versions

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

0080h:	"MCHFM0"	Program ID code
0086h:	08h	ROM serial #
0087h:	00h	FM sound chip type
0088h:	10h	software version #

For the identification of either SFG-01 or SFG-05, it is only necessary to look for the first 6 byte code "MCHFM0" from address 0080h.
However, the contents of address 0088h provide the information for the type of SFG.
For SFG-01, (0088h)=00h - 0Fh:
for SFG-05, (0088h)=10h and up.

October 16, 1985

Chapter II Basic Functions

2-1 Instrument, Queue, and Keyboard

The Instrument, queue, and keyboard form the fundamental structure of M-BIOS.
The following illustration depicts the relationship of these three main functional units.

(Instrument)

Instrument is depicted at the rightmost portion of the figure.
The instrument is meant to be an event processing system.
It processes the incoming event requests to realize the musical performance such as playing melody.
The instrument is defined by a control block called IDB (Instrument Definition Block).
There are up to 8 instruments that can be defined with IDB with IDB#0 to IDB#7.
Each IDB can be assigned with 1 to 8 cahnnel resources of FM sound LSI. Channel assignement is dynamically performed for each instrument by MBIOS. This way, as long as total channel resources of FM LSI does not exceed the maxumum of 8, the user of the MBIOS can define as many as 8 different instruments with any number of channels (between 1 and 8) linked up with each instrument.

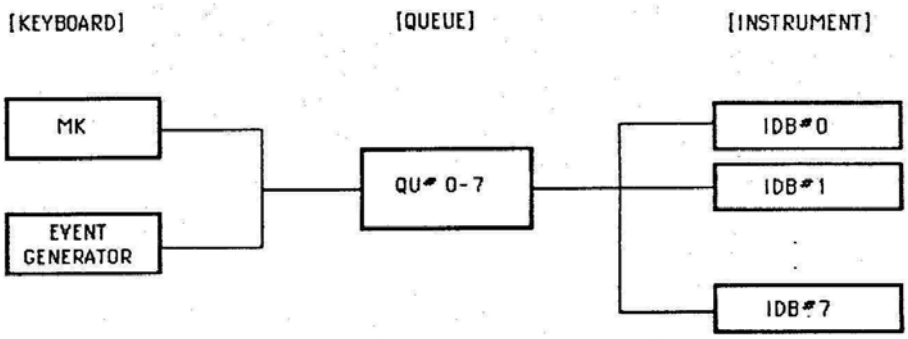
(Queue)

In the middle ofthe figure, Queue is depicted.
This functional module acts as processing buffer between input (events) and the next functional module, called Instruments.
There are 8 queues that can be used by the instruments.
Queues are numbered from QU#0 to QU#7.

(Keyboard)

In the figure, the keyboard is an input to the music processing system. It issues key-on/off requests and associated velocity inputs. Since the requests are time dependent, they are called events.
MK a music keyboard attached to SEG-05 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.


[KEYBOARD]
[QUEUE]
[INSTRUMENT]
MK
EVENT GENERATOR
QU# 0-7
IDB#0
IDB#1
IDB#7

December 3, 1985

[IDB and Event]

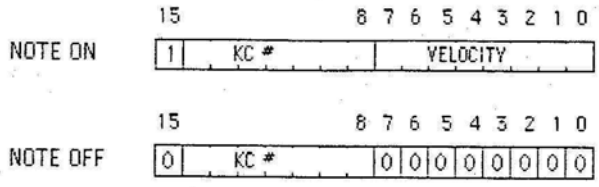
Related SV-calls:

S-00	define IDB
S-09	assign channel
S-0A	assign IDB to queue and/or MIDI channel
S-0B	All-Note-Off by IDB
S-12	define play-mode
R-00	Damp (issue All-Note-Off to all queues and damp all the currently engaged voices to channels)
R-01	All-Note-Off by queue
R-02	set event into queue
P	play

(Event)

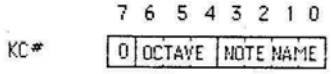
The instrument is the most important functional unit in MBIOS.
It is defined by a 128 byte control block called IDB. Service call S-00 is used to define IDB.
The IDB contains the information of the run-time parameters for the performance, as well as the preset data for the FM sound chip. The IDB should also be assigned with the channels usage of the FM sound LSI. This is done by S-09.
Input to IDB is event data. The event buffer called queue, should be linked to the IDB. To do this, S-0A call is used.
The event data is fed to the queue in the form of argument to R-02 service call (All-Note-Off event is however issued as the service call R-01). The events are then 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.



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-05 as shown in the following.



<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> -

(Event output to MIDI)

When the events are transmitted from MIDI interface, they are packed into the standard MIDI message format with MIDI destination channel number, MIDI-KC#, and MIDI-velocity.
They are as following:

NOTE-ON

7	6	5	4	3	2	1	0
1	0	0	1	MIDI #			
0	MKC#						
0	MVELOCITY						

NOTE-OFF

7	6	5	4	3	2	1	0
1	0	0	0	MIDI #			
0	MKC#						
0	1	0	0	0	0	0	0

ALL-NOTE-OFF

7	6	5	4	3	2	1	0
1	0	1	1	MIDI #			
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	1	1	MIDI #			
0	1	1	1	1	0	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.

{difference between MBIOS event format and MIDI format}

User is advised to note the difference between MBIOS event format and MIDI data format.
While the SFG-05 internal KC# looks as shown before, 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#0	00h	0Dh

A4	4Ah	45h
C7	7Eh	6Ch

Also observe that MBIOS velocity is different from MIDI velocity format.
The SFG-05 internal velocity code covers an 8-bit range between 00h (Minimum) and FFh (Maximum).
MIDI velocity can be derived from MBIOS velocity format by dividing it by 2.

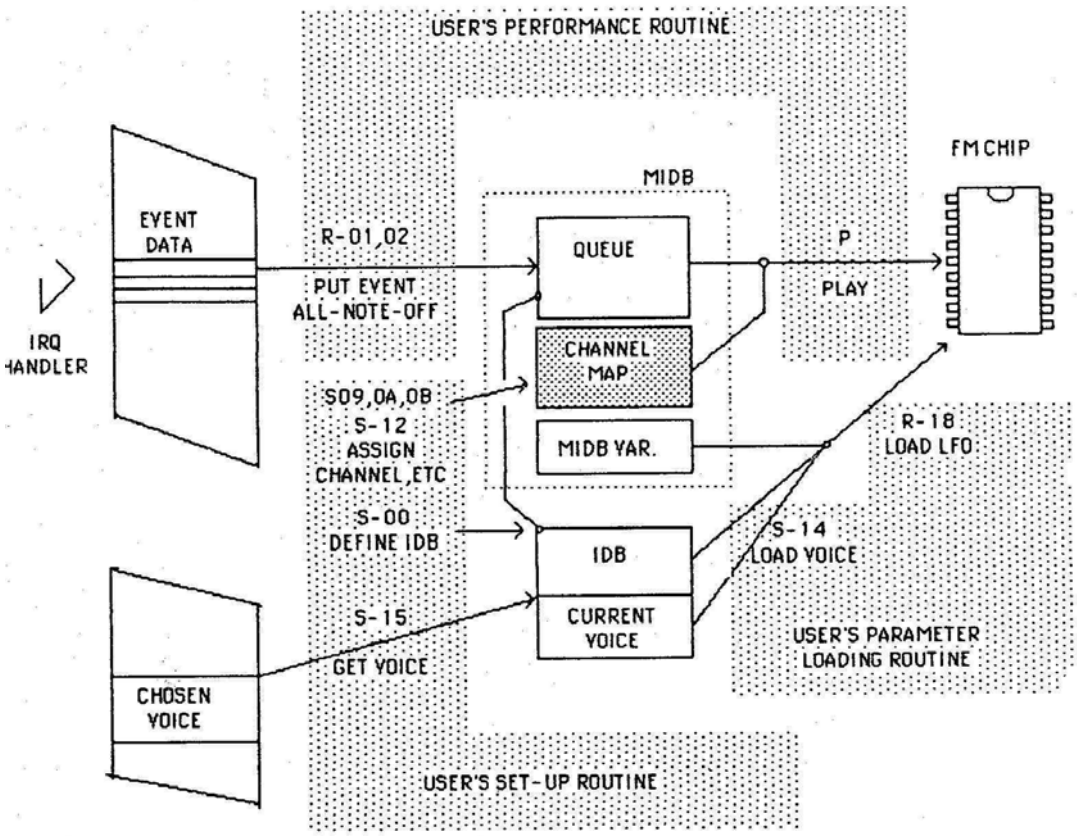
(channel assignemnet)

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.

1) 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).

2) 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).


USER'S PERFORMANCE ROUTINE
FM CHIP
MIDB
EVENT DATA
R-01,02
PUT EVENT ALL-NOTE-OFF
QUEUE
P
PLAY
IRQ HANDLER
CHANNEL MAP
S09,0A,0B S-12 ASSIGN CHANNEL,ETC
MIDB VAR.
R-18 LOAD LFO
S-00 DEFINE IDB
IDB
S-14 LOAD VOICE
S-15 GET VOICE
CURRENT VOICE
USER'S PARAMETER LOADING ROUTINE
CHOSEN VOICE
USER'S SET-UP ROUTINE

October 30, 1985

[Queue]

Related SV-calls:

R-00	Damp
R-01	All-Note-Off by queue
R-02	set event into queue

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).
Queue's are named as QU#x, and refered as.

QU#x

0 thru 7 QU#0 thru QU#7

(QU numbers between 8 and 15 are reserved for future use).

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.

October 16, 1985

[Keyboard]

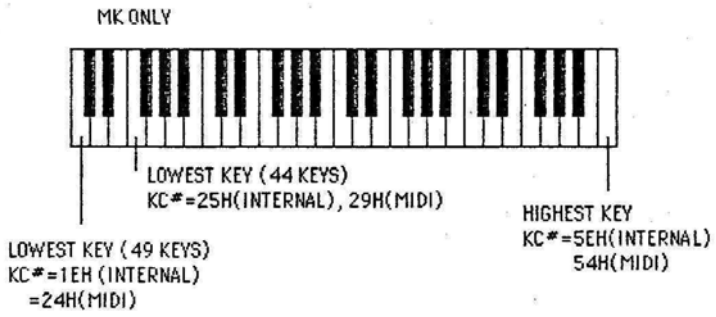
The keyboard is, in effect , the event generator.
The external keyboard attached to the SFG-05 (MK) is a music keyboard.
An automatic performance buffer is treated as a music keyboard, too.
For "keyboards", like MK or automatic peformance buffer, events are then input to QU#0-QU#7.

[Music Keyboard]

Related SV calls:

K-00	initialize MK
K-01	scan MK
K-02	report MK status
AST-01	MK trigger by asynchronous trap

M-BIOS supports a 49 key keyboard.
It is as shown below. It is used as the mounted keyboard (MK).



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

2-2 Performance parameters for the instrument (IDB)

[Portamento]

Related SV-calls & memory reference:

S-12	define play-mode
R-19	load KC
i.port	IDB(w)/portamento speed
m.clka	MIDB(w)/clock-A interval

This is applicable to only single voiced IDB (IDB assigned with one channel).

Portamento mode can be enabled via S-12 call, together with appropriately set portament parameters into IDB.

The method to implement portamento requires repeated updates of KC with specified parameters. To do this, R-19 call (to update KC with specified pitch increments and load it into channel) should be called by an appropriate ,say interrupt clock-A, interval.

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.

[Trigger Type]

Related SV-call:
S-12 define play-mode

Applicable only to single voice IDB.
The trigger type can be determined by using the Trigger Mode (set by S-12 call).
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 but to change only the pitch during the key stacking. Thus the envelope continues as it is when the second key is on. This is called Single Trigger mode.

[Sustain Control]

Related SV-call & memory reference:
S-12 define play mode
i.sust IDB(w)/sustain RR

This is applicable to all IDB's.
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 usual. 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.

[Keycode Sensible Range]

Related SV-call & memory reference:

S-00	define IDB
i.krng	IDB(w)/key code range by instrument

It is possible to set the KC sensitivity range of the IDB, where only KC#'s within that range are only accepted.
This function is useful to implement a register sensitive keyboard, such as split keyboard.
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.
With All-Note-Off, since it is granted and performed regardless of the keycode range, the care must be paid when IDB is linked-up with MIDI device. For the MBIOS will issue All-Note-Off to the outside of the keycode range of MIDI device (where the events may not necessarily need to be turned off).

[Transposition]

Related SV-call & memory reference:

S-00	define IDB
R-19	load KC
i.trns	IDB(w)/transpose by instrument
m.trns	MIDB(w)/master 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.
Note that transpositions by IDB or MIDB will only be realized by issuing R-19, for it is R-19 that loads the parameters to FM shynthesizer LSI from i.trns or m.trns.
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.

[Pitch Bend]

Related SV-calls & memory reference:

S-11	set pitchbend
R-19	load KC
i.pchb	IDB(w)/pitchbend depth

Pitch bend is like a portamento function.
When pitch bend parameter is handed to MBIOS via R-11, it requires for KC to be updated and loaded into channel, with appropriate interval using R-19.
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.

[Volume, Brilliance]

Related SV-calls:

S-10	set brilliance
S-13	set volume

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).

[Noise]

Related SV-call & memory reference:

R-18	load LFO
m.nois	MIDB(w)/noise frequency

The OPM has a noise generator which is started when operator 3 of channel 7 receives a key-on command.
Noise parameter such as noise frequency (seed to generate random number) can be loaded into the channel via R-18.
If the voicing parameter data that has been loaded into the channel 7 has been noise-enabled, it will generate noise.

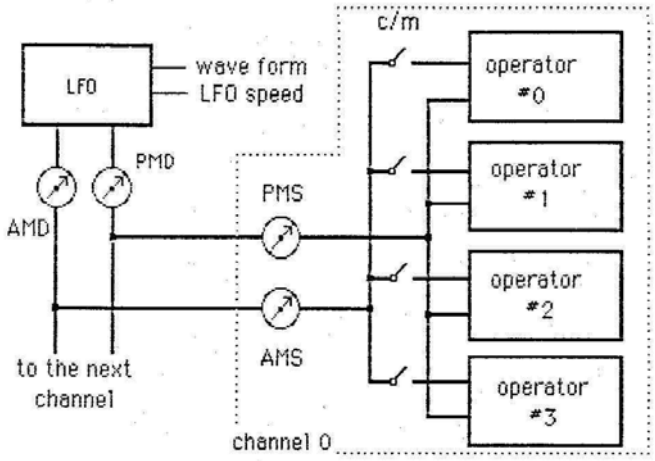
[LFO]

Related SV-call and memory reference:

S-18	set AMS/PMS
R-18	load LFO
m.lfo	MIDB(w)/LFO frequency
m.amd	MIDB(w)/AMD
m.pmd	MIDB(w)/PMD
m.ctrl	MIDB(w)/LFO waveform

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 the following figure, LFO outputs adjusted by AMD and PMD are common to all the instruments.
Above items that are common to the system (in the MIDB) are loaded to the FM LSI chip by R-18 call.

AMS and PMS set the sensitivity to LFO modulation for each individual instrument. Note that AM (amplitude modulation) is only effective to the carriers of the current operator algorithm.
The Triggered Sync function initializes the LFO phase to 0(zero) in synchronization with Note-on events.
AMS, PMS and Sync are voice data dependent parameters.



2-3 Voice Library

Related SV-calls:

S-03	define UVL
S-14	load voice
S-15	get voice
S-16	put voice
S-19	get and load voice
S-21	read UVL
S-22	write UVL

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 the following page.

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 chord generation, voices 40 - 41 for bass note, voice 44 and 45 for percussion , are used by M-Monitor.
Voice 46 is reserved for use by the CSM voice sysnthesis driver (IDB#CSM).

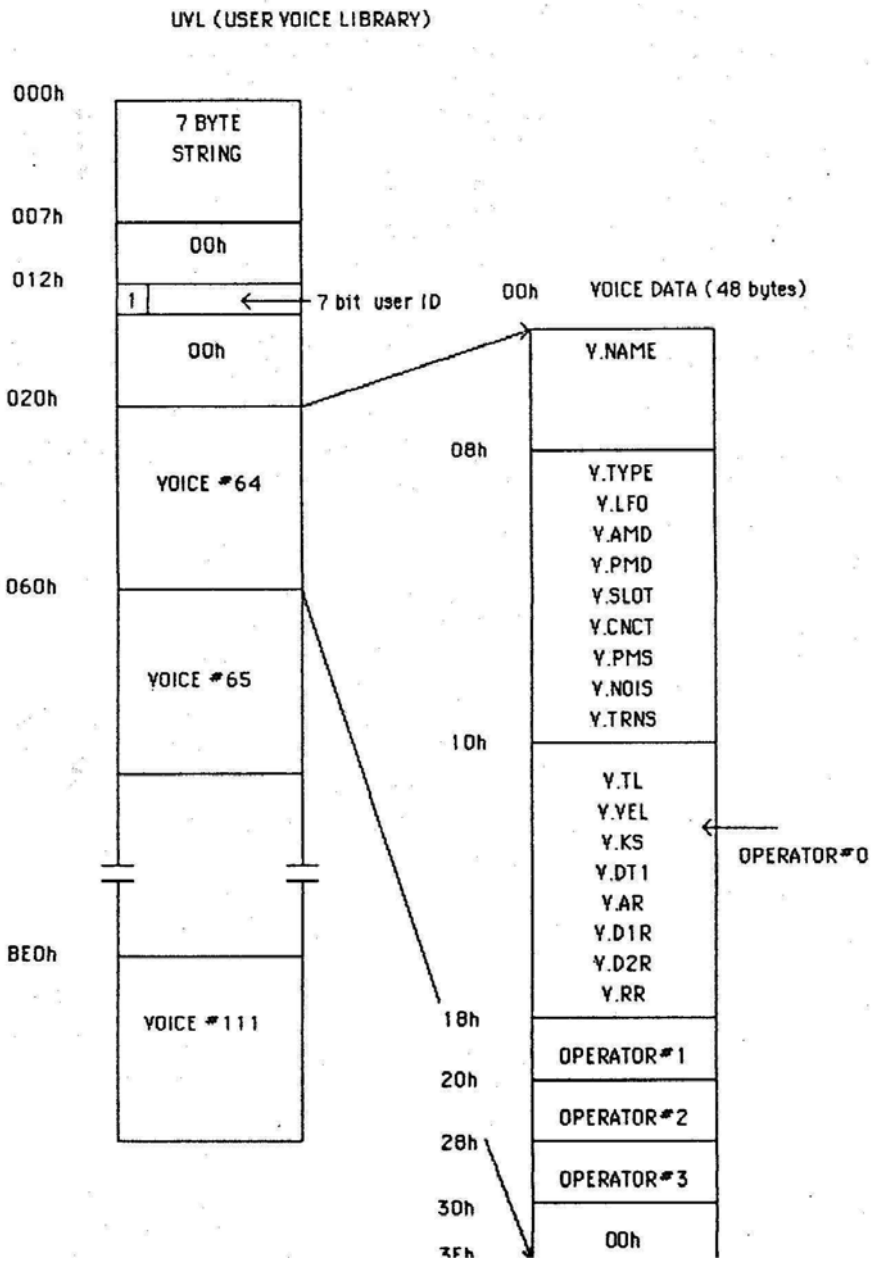
The following is a directory of SVL.

1 brass1	17 piccolo	33 lo string
2 brass2	18 oboe	34 horn lo
3 trumpet	19 clarinet	35 whisle
4 string1	20 glocken	36 storm
5 string1	21 vibraphone	37 rm.bass
6 epiano1	22 xylophone	38 rm.flt

7 epiano2	23 koto	39 rm.guit
8 epiano3	24 zither	40 rm.horn
9 guitar	25 clav	41 r1.bass
10 ebass1	26 harpsichord	42 r2.bass
11 ebass2	27 bells	43 snaredrum
12 eorgan1	28 harp	44 rd cymbal
13 eorgan2	29 smadsyn	45 perc 1
14 porgan1	30 harmonica	46 perc 2
15 porgan2	31 steeldrum	47 csm param
16 flute	32 timpani	48 no voice

MBIOS is equipped with a file utility to save UVL and load it back to UVL using CMT, data cartridge, and the disk.

In the following pages, the format of UVL is depicted with further breakdown of voice data format and detailed voice parameter map.


UVL (USER VOICE LIBRARY)
000h
7 BYTE STRING
007h
00h
012h
1
7 bit user ID
00h
020h
YOICE #64
060h
YOICE #65
BE0h
YOICE #111
00h
YOICE DATA (48 bytes)
Y.NAME
08h
Y.TYPE
Y.LFO
Y.AMD
Y.PMD
Y.SLOT
Y.CNCT
Y.PMS
Y.NOIS
Y.TRNS
10h
Y.TL
Y.VEL
Y.KS
Y.DT1
Y.AR
Y.D1R
Y.D2R
Y.RR
OPERATOR#0
18h
OPERATOR#1
20h
OPERATOR#2
28h
OPERATOR#3
30h
00h

October 17, 1985

VOICE DATA MAP

Name	Address Offset	Contents	Explanations
v.name	00h	[7 byte string]	{name of the voice}
v.type	07h	[**** ****]	{user code (any number)}
v.lfo	08h	[**** ****]	{LFO frequency} 00h=0.0008Hz (slow) 80h=0.2134Hz FFh=52.9Hz(fast)
v.amd	09h	[xyyy yyyy]	{LFO enable & AMD} x=<1> enable to load LFO y=0-7Fh, deepest AMD
v.pmd	0Ah	[xyyy yyyy]	{sync LFO & PMD} x=<1> sync LFO at key-on y=0-7Fh, deepest PMD
v.slot	0Bh	[0xyz u000]	{enable operator} x=<1> operator #1 y=<1> operator #2 z=<1> operator #3 u=<1> operator #4
v.cnct	0Ch	[xyzz zuuu]	{LR, feedback & algorithm} x=<1> stereo L output y=<1> stereo R output z=0-7 (0-4 pai) ,feedback level u=0-7, algorithm number
v.pms	0Dh	[0xxx 00yy]	{pms and ams} x=0 +/- 0 cent (pms) x=1 +/- 5 cents x=2 +/- 10 cents x=3 +/- 20 cents x=4 +/- 50 cents x=5 +/- 100 cents

			x=6 +/- 400 cents x=7 +/- 700 cents y=0 0dB (ams) y=1 -24 dB y=2 -48 dB y=3 -96 dB
v.nois	0Eh	[xyyz zzzz]	(noise/lfowave/noise freq.) x=<1> enable noise y=<00> sawtooth y=<01> rectangler y=<10> triangler y=<11> sample and hold z = noise frequency
v.trns	0Fh	[**** ****]	(transposition by voice) -12700 cents to 12700 cents (2's complement)

---

operator #0

v.tl	+00h	[0*** ****]	(OL original level) 0 dB to -95.25dB (by 0.75dB)
v.vel	+01h	[xyyy 000z]	(key level scaling & velocity) x=<0> type0 (low pass type) x=<1> type1 (high pass type) y= velocity sencitivity carrier 0 to +-10.5 dB (1.5 dB) index 0 to +-5.25 dB (0.75dB) z=<1> enable brilliance
v.ks	+02h	[xxxx yyyy]	(key level scaling depth, adj.) x= key level scaling depth 0 to -22.5dB (by 1.5 dB) y= TL (total level adjust) 0 to -11.25dB (by 0.75dB)
v.dt1	+03h	[-xyy zzzz]	(detune & multiple) x=<0> positive (up) detune x=<1> negative (down) detune

			y= 0 - 3 detune value (max=3) z=multiple z=0 pitch*0.5 z=1 pitch*1 ..... z=15 pitch*15
v.ar	+04h	[xx0y yyyy]	{key rate scaling & attack rate} x= 0 - 3 (max(fastest rate)=3) y= 0 - 31(max=31)
v.dir	+05h	[x00y yyyy]	{ams enable & decay rate-1} x=<0> disable ams (modulator) x=<1> enable ams (carrier)
v.d2r	+06h	[xxxy yyyy]	{detune-2 and decay rate-2} x=0 DT is *0 x=1 DT is *1.41 x=2 DT is *1.57 x=3 DT is *1.73 y=0 - 15 decay rate-2(max=15)
v.rr	+07h	[xxxx yyyy]	{sustain level and release rate} x= 0 to 14 (0 to -42 dB by 3dB) x=15 (-93 dB) y= 0 - 15 release rate(max=15)

---

operator #1

18h-1Fh

---

operator #2

20h-27h

---

operator #3

28h-2Fh

---

all 0's

30h-3Fh

2-4 Important Voicing Parameters

Related SV-call:

S-14	load voice
S-19	get and load voice

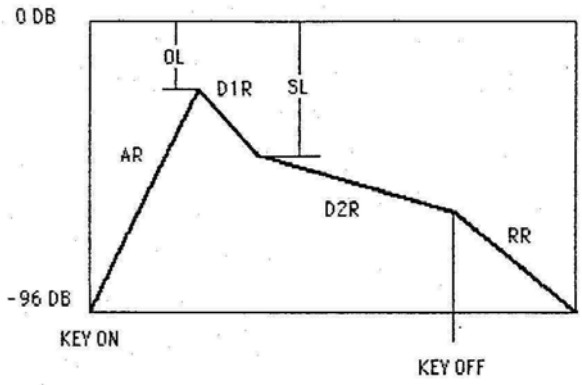
[Velocity]

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.

[Envelope]

The Envelope can be set independently for each operator.

October 16, 1985

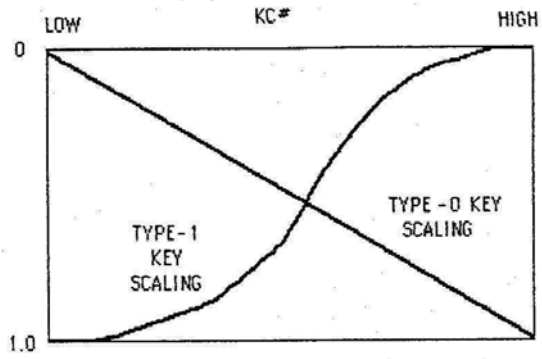


ENVELOPE

As shown here, 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.

[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.



KEYBOARD SCALING

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.
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).

[Multiples]

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]

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]

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.

[Feedback]

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]

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

[Stereo L/R]

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.

Related SV-calls & memory reference:

S-02	define EVB
S-04	initialize EVB
S-23	read EVB
S-24	write EVB
R-08	start recording
R-09	set recording clock
R-0A	stop recording
R-0B	start playback
R-0C	set playback clock
R-0D	stop playback
f.evb	MIDB(R)/recording, playback status
m.fevb	MIDB(W)/EVB file name

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 file transfer of the EVB to save or load with CMT, data cartridge, and disk.

2-7 CSM Vocal Synthesis

Related SV-calls:

S-0	define IDB (for IDB#CSM)
S-09	assign channel
S-28	CSM voicing

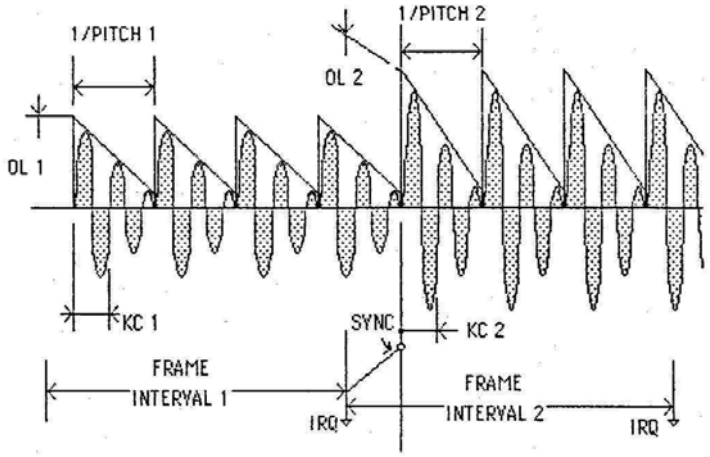
The vocal synthesis supported by MBIOS is based upon the technique called CSM (Composite Sinusoidal Method).
It simulates the spectrum characteristics of the human voice by the generation of a few sine waves of different frequencies.
In the case of MBIOS, 4 operators (4 sine waves) approximate the spectrum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

$$Y(t) = \sum_{0}^{3} A_i \cdot E(t) \cdot \sin(w_i\ t + \varphi_i)$$

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.



The above illustrates the principle of CSM voice synthesis. That is, one spectrum component of possible four formants is generated. Within the frame interval, exponentially decaying sin-wave (with its formant frequency and intensity being KC and OL respectively) is repeatedly calculated with the given pitch.

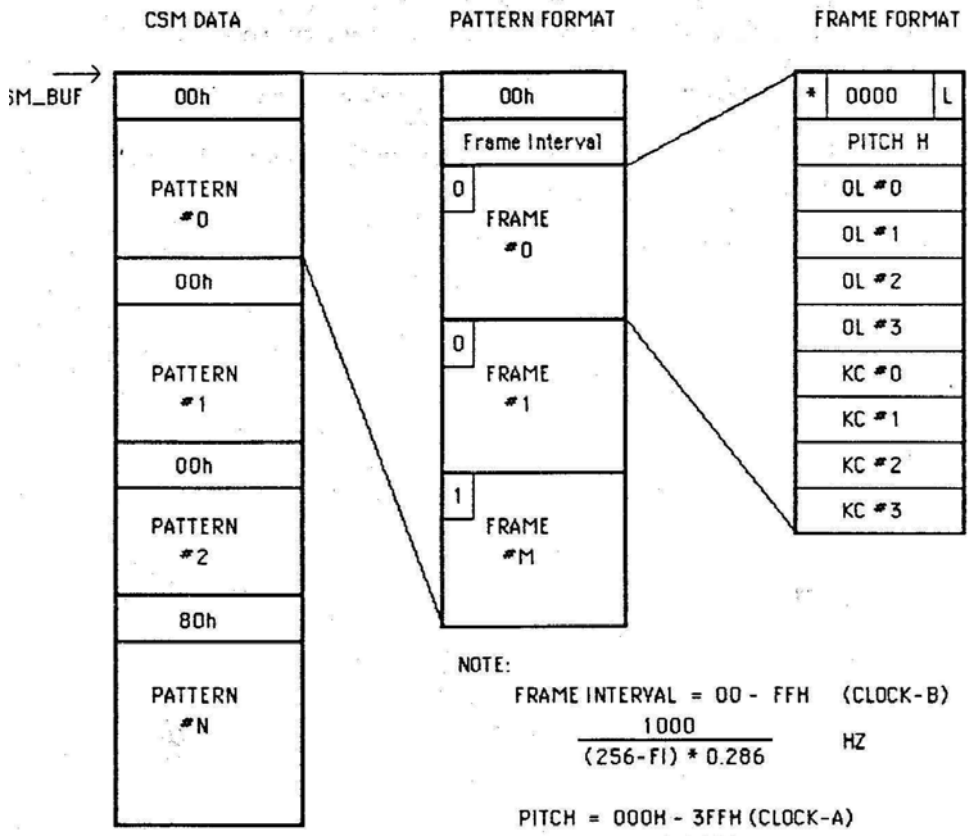
The voice pitch is actually the analyzed data from the original sound, and realized in such a way that envelope repeats by itself with interval given by 1/pitch. This repetition is internally done within the hardware of FM chip, using clock-A (no interrupt occurs).

Frame interval is the timing that input data for the genaration of formant should be updated. This is actually the frame length that was used in the formant analysis (frame length of auto correlation window, say 20 ms).

This interval is determined by appropriatelyloaded clock-B interval timer. When the interval depletes, IRQ is generated to MBIOS, so that MBIOS knows the timing of data update with next frame data. Note that, actual loading of the frame data into the calculation circuit of LSi is synchronized with the end of current frame calculation. LSI has latches to hold these data until clock-B timer is depleted.

In order to call CSM driver, whole CSM data (as depicted in the following page) should be prepared with its top address pointed by

the argument to the CSM-call.
The whole CSM data is comprised of successive voice pattern information. The voice pattern information is a meaningful unit of voice.
In each pattern, there are succession of formant frames. Each frame was derived per one CSM analysis window.
Finally frame data is comprised of four fomant frequency (KC) and four formant intensity data.
Naturaly, while MBIOS is running CSM sysnthesis, four sin-wave generations are taking place simultaneously.



NOTE:

FRAME INTERVAL = 00 - FFH (CLOCK-B)

$$\frac{1000}{(256\text{-FI}) * 0.286} \text{ HZ}$$

PITCH = 000H - 3FFH (CLOCK-A)

$$\frac{1000000}{(1024\text{-P}) * 17.88} \text{ HZ}$$

OL = 00H - 7FH (FORMANT INTENSITY)
-0.75 DB STEP

KC = [0XXXYYYY] (FORMANT FREQUENCY IN KC#)

XXX=OCTAVE NUMBER
YYYY=NOTE NUMBER

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).

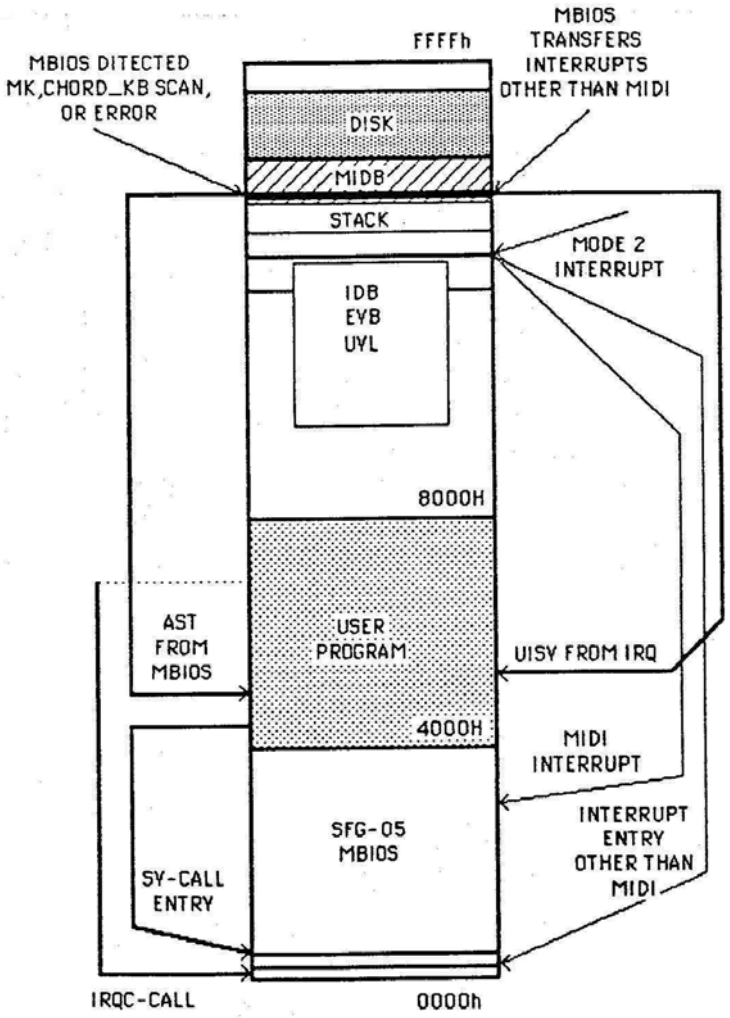
The genral format to transfer data between the MBIOS and the user program is by means of registers and tables (or buffers).
The latter include the MIDB (Master Instrument Definition Block), IDB (Instrument Definition Block), EVB (Event buffer), and UVL (User Voice Library).
These are the buffers that 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.

User program to MBIOS interface:

SV calls (Service calls)
IRQC (Interrupt entry)

MBIOS to user program interface:

AST (Asynchronous System Trap)
UISV (User Interrupt Service Vector)



MBIOS INTERFACING

[Slot switching]

The management of the slot switching 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 the service of file transfer with CMT, data cartridge or disk, is requested, MBIOS switches the slot (assuming the primary slot#0 for BASIC only) by itself appropriately to appropriate the file services provided in the repertoire of BASIC interpreter. Thus, during these services, there is no need for the user to worry about slot switching and interrupt handling.

[Memory map]

The memory allocation under MBIOS control is as shown in the following.
In the case of SFG-01, or under operating mode 1.0, 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, UVL, and the stack area can be allocated anywhere between 8000h and EC00h.

The above fixed mapping:, that is, the operating mode 1.0 conforms to SFG-01 MBIOS memory mapping.
However in the case of operating mode 1.1 and 2.0, the base address for MIDB and MBIOS work (780h bytes in total) can be allocated anywhere in RAM (between BOTTOM and HIMEM).
Selection of the operating mode and the allocation of base address are done by I-call (1.0, 1.1, or 2.0).
In doing indexed addressing of the variables in MIDB and MBIOS work, MBIOS internally uses 128 bytes (between F975h and

F9F5h) of PLAY queue buffer. Remaining 256 bytes of the queue buffer can be freely used by application program.
While MBIOS is active, to avoide possible invokation of PLAY routine of BASIC, PLAY hook is replaced with NOP routine at the beginning of I-call.

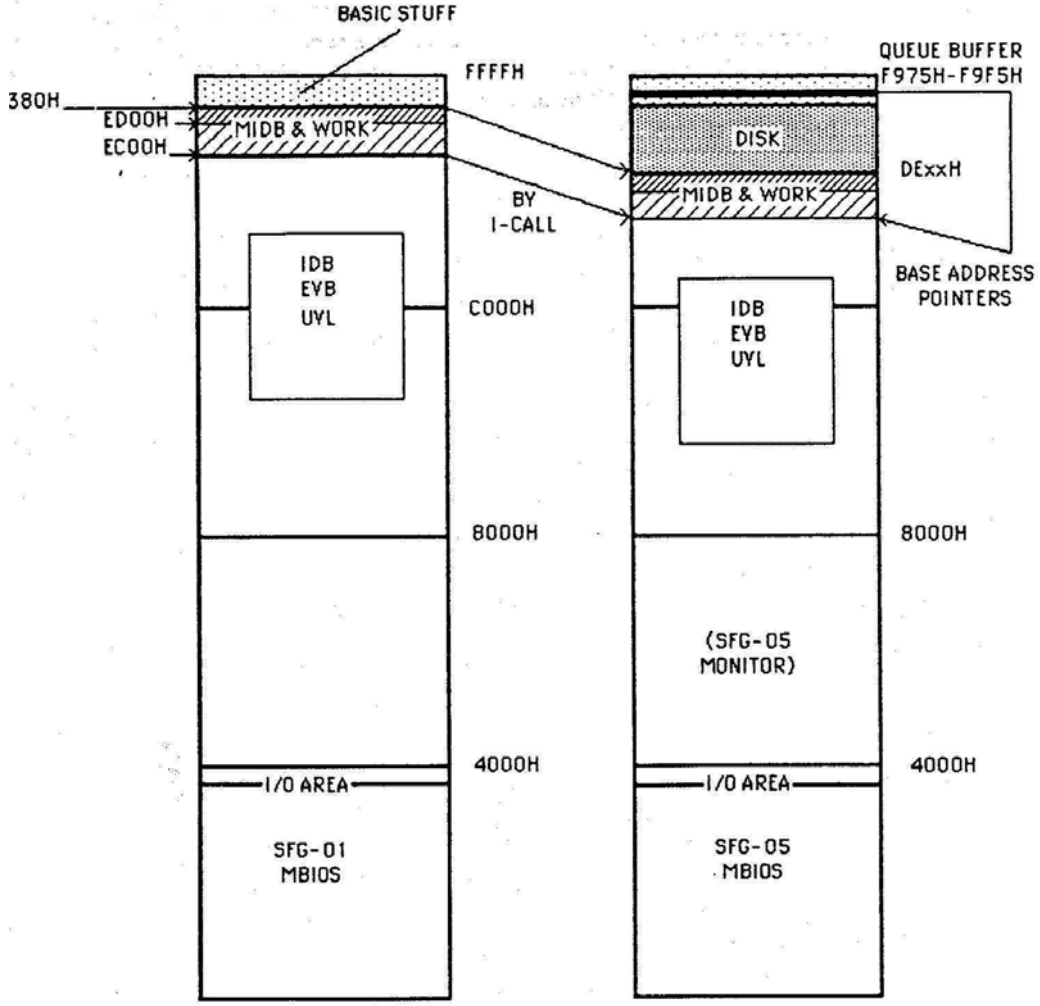


FIG 3.1 MEMORY MAP OF SFG-01 AND SFG-05

3-3 Direct Access to MIDB and IDB

[MIDB]

The MIDB (Master Instrument Definition Block) occupies a 256 byte area starting at base address specified by I call (in the case of SFG-01, it is fixed to EC00h).
The map of MIDB is shown in the following pages.

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 UISV (User Interrupt Service 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.

MIDB MAP

Name	Address Offset	R/W	Contents	Explanations
m.clka	00h	W		(clock-A interval)
			[**00 0000]	clock-A low
			[**** ****](+1)	clock-A high
				0000h=18.2 ms (max)
				8000h=9.1 ms
				FF00h=0.071 ms (min)
m.clkb	02h	W		(clock-B interval)
			[---- ----]	clock-B low
			[**** ****](+1)	clock-B high
				0000h=72.8 ms (max)
				8000h=36.4 ms
				FF00h=0.285 ms (min)
m.trns	04h	W		(Master transposition)
			[**** ****]	Fractional pitch
			[**** ****]	KC (2's complement)
m.lfo	10h	W	[**** ****]	(LFO frequency)
m.amd	11h	W	[0*** ****]	(AMD)
m.pmd	12h	W	[0*** ****]	(PMD)
m.ctrl	13h	W	[0000 00**]	(Wave form of LFO)
m.nois	14h	W		(Noise control of channel 7)
			[x00- ----]	<1> noise enable
			[-00y yyyy](+0)	noise frequency
f.evb	1Bh	R		Recording/Playback status
			[0000 00x-]	<1> recording,<0> playback
			[0000 00-y](+0)	<1> busy
m.chrd	23h	R		(reserved)
m.sram	28h	R	[2 byte L/H]	(s-ram size)
m.f8	2Ah	W		(MIDI-out switch)
			[0000 00-x]	<1> F8h to MIDI by Timer-A
			[0000 00y-](+0)	<1> F8h to MIDI by Timer-B

m.i38h	30h	W	[2 byte vector]	(0038h UISV)
m.icka	32h	W	[2 byte vector]	(IRQ-A UISV)
m.ickb	34h	W	[2 byte vector]	(IRQ-B UISV)
m.iund	38h	W	[2 byte vector]	(unidentified-IRQ UISV)
m.trmd	3Ah	W	[2 byte vector]	(MIDI fifo/receive vector)
m.trmk	3Ch	W	[2 byte vector]	(MK trigger trap)
m.trer	3Eh	W	[2 byte vector]	(error trap vector)
m.fevb	44h	W	[6 byte string]	(file name for EVB data)
t.rhy0	88h	W	[96 byte]	(reserved)
m.thru	E8h	W		(MIDI/thru assign)
			[0000 000*]	<1> F8h thru enable
			[0000 000*](+1)	<1> F9h thru enable
			.....	
			[0000 000*](+7)	<1> FFh thru enable

[IDB]

IDB is a 128 byte control block to represent currently opend instrument.
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 the following pages, the IDB map is depicted.

IDB MAP

Name	Address offset	R/W	Contents	Explanation
i.krng	00h	W	[0*** ****] [0*** ****](+1)	(Key code range) Highest key code Lowest key code
i.pchb	02h	W	[0000 ****]	(Pitch-bend depth) 00h=0 cent 01h=100 cents 08h=1200 cents
i.trns	03h	W	[**** ****] [**** ****](+1)	(Trnsposition by instrument) Fractional pitch KC, (2's complement)
i.port	05h	W	[**** ****]	(Portamento speed) 00h=no portamento 01h=fast speed FFh=slowest
i.sust	06h	W	[0000 ****]	(RR(Release rate) when sustain/on is requested) applies to all 4 operators
v.name	10h	R/W	[48 byte voice]	(voice data) loaded by MBIOS upon issuing SV-call

3-4 IRQ Processing via UISV

[IRQ modes]

MBIOS operates on IRQ mode of either 1 or 2 of the Z80 CPU, depending on the choice of operating mode (1.0,1,1. or 2.0).
IRQ mode 1 is set during the processing by the I-call1.0/1.1.
IRQ mode 2 is set during the processing by the I-call 2.0.
IRQ mode 1 is compatible to MSX standard interrupt, 38h as its interrupt entry.
IRQ mode 2 is vectorized interrupt. This is espetially useful when fast processing of MIDI reception is to be implemented.

In addition to the IRQ mode of the system, whether the state should be EI'ed or DI'ed differs the types of SV-calls to be used.
In the operating mode2.0, the following services should be requested under IRQ-disabled condition.

I-call, End-call, M_MONITOR -call
RDSLT, WRSLT, CALSLT, ENASLT, CALLF, RDPP.0

The following service requests can be made under IRQ-Enabled condition.

R, S, P, K, M, F-call, BDOS-call
UISV (m.icka, m.ickb, m.iund, m.trmd)
AST (m.trmk, m.trer)

[Source of interrupts]

For MBIOS, there are 3 sources of interrupts that are generated by the hardware of SFG-05; i.e., Clock-A and clock-B and MIDI chip. Including VDP as another source of interrupt, 4 sources of interrupts should be considered.
When I-call 1.0/1.1 is first issued to initialize MBIOS, clock-A and clock-B are automatically interrupt enabled.
For applications that require disabling the clocks afterward, refer to section 4-5.

If MIDI-FIFO has been defined and if I-call2.0 is requested, MIDI chip is also interrupt enabled.

[Interrupt control flow in the operating mode 1.0/1.1]

In the following figure-A, operating mode 1.0, and 1.1 are illustrated.
This mode is compatible to Interrupt handling of SFG-01.
Note that when mode 1 interrupt hits to 38h IRQ entry address, the interrupt is disabled until necessary processing is completed.

If the interrupt directly hits 38h, there is an option provided by the MBIOS by which the transfer is made solely to user's processing routine.
This can be done with MBIOS by refering to UISV i38h.

If that UISV is not defined, the control flow goes to the next stage where the source of interrupts are polled.
The polling is made according to the priority depicted in the figure; that is, clock-A, clock-B, and other source (normally VDP).
(However in operating mode 2.0, the priority order between clocks A and B can be changed by S-0E call).

If the souce of the interrupt is identified, and if the corresponding UISV is defined in MIDB, then the jump is made to the designated user's routine.
When clock-A or clock-B is detected, the source of interrupt is reset by MBIOS before the jump is made. However, for the last branch (other cause, VDP, etc.) no attempt is made to reset the interrupt request registers. Thus it is a user's responsibility to reset the interrupt cause in that routine.
User also has to enable interrupt (EI) before he issues RET in his handling routine.

Also in operating mode 1.0 or 1.1, IRQC-call (009Fh entry) is provided in order for the user to jump into the interrupt processing from the different slot. This is usefull if the user detects the interrupt in the different slot, say in BASIC interpreter, and tries to route the control to the MBIOS by way of BASIC's interrupt hook.

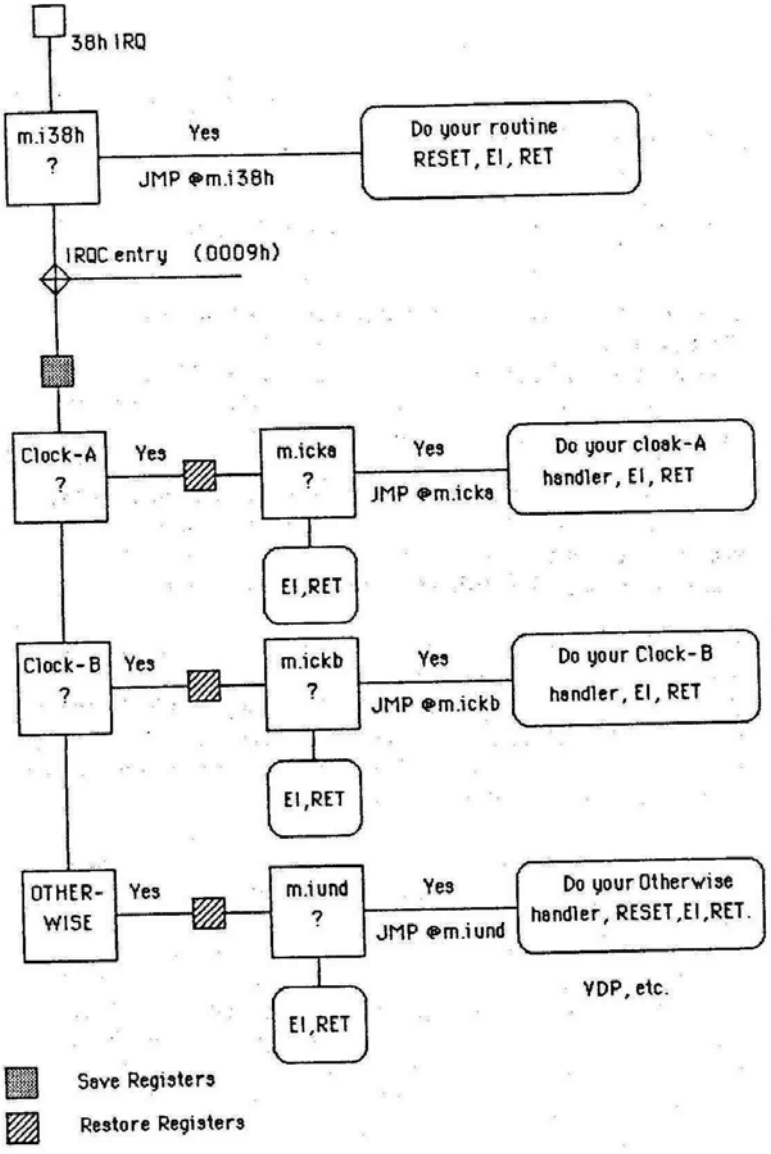


Fig. 3.2-A IRQ Control Flow in Operating mode 1.0 & 1.1

[Interrupt control flow in the operating mode 2.0]

In the following figure-B, the procedure for the operating mode 2.0 is illustrated. Here, mode2 interrupt is of cource assumed. Mode 2 vector is defined in the work space of MBIOS, and there it is invisible from the user.
Defining the vector and setting mode2 are done during the I-call with operating mode 2.0.

When MIDI interrupts caused by Tx RDY or Rx RDY hits the system, the processing is done by the MBIOS totally independent upon the rest of interrupt processing.
It is assumed that 256 byte FIFO for MIDI reception has already been defined. That is, by way of a separate vector, when RxRDY is the cause of the interrupt, the received MIDI data is buffered into FIFO buffer.
When MIDI data is stored into the FIFO, MBIOS raises the flag to tell that FIFO is significant, which will be further processed later on.
Also MIDI data is written out when TxRDY is the cause of interrupt.
The interrupt is disabled during the above processing.
However, the processing is completed very quickly so that no possibility of UART over-run is anticipated.

In the meantime, if the cause of the interrupt is something other than MIDI (this is judged by interrupt arbitration logic in the MIDI chip), the separate vector routes the control to the process depicted in the left hand half of the figure-B.
Here, the option to the private processing is provided via UISV, m.i38h. However, for MIDI processing, it is strongly advised not to use this option due to the complexicity of DI/EI management.

The polling method of the interrupt cause is somewhat diferrent from that of the operating mode 1.0/1.1.
Here, after m.i38h check, the possible causes of interrupts are polled at once and corresponding flags are made. IRQ causing registers are also cleared. And EI is issued at this point.
This enables the MIDI interrupt still hits the system.
Naturally, the other source of the interrupt might come to the system as well.
Therefore during this process of job schecduling, the use-count

management is involved.

Now, in the process, the first flag to be scanned is MIDI flag. If it is on, and if UISV m.trmd is defined, the jump is made to designated user routine where received MIDI data should be fetched out of FIFO. Since EI has already been issued, there is no need for user's individual routine to issue EI before RET.

After MIDI flag, clock-A, clock-B, and VDP are scanned in that priority order. Branching method is the same as done in operating mode 1.0/1.1.

Also notice that IRQC-call is not available in the operating mode2.0.

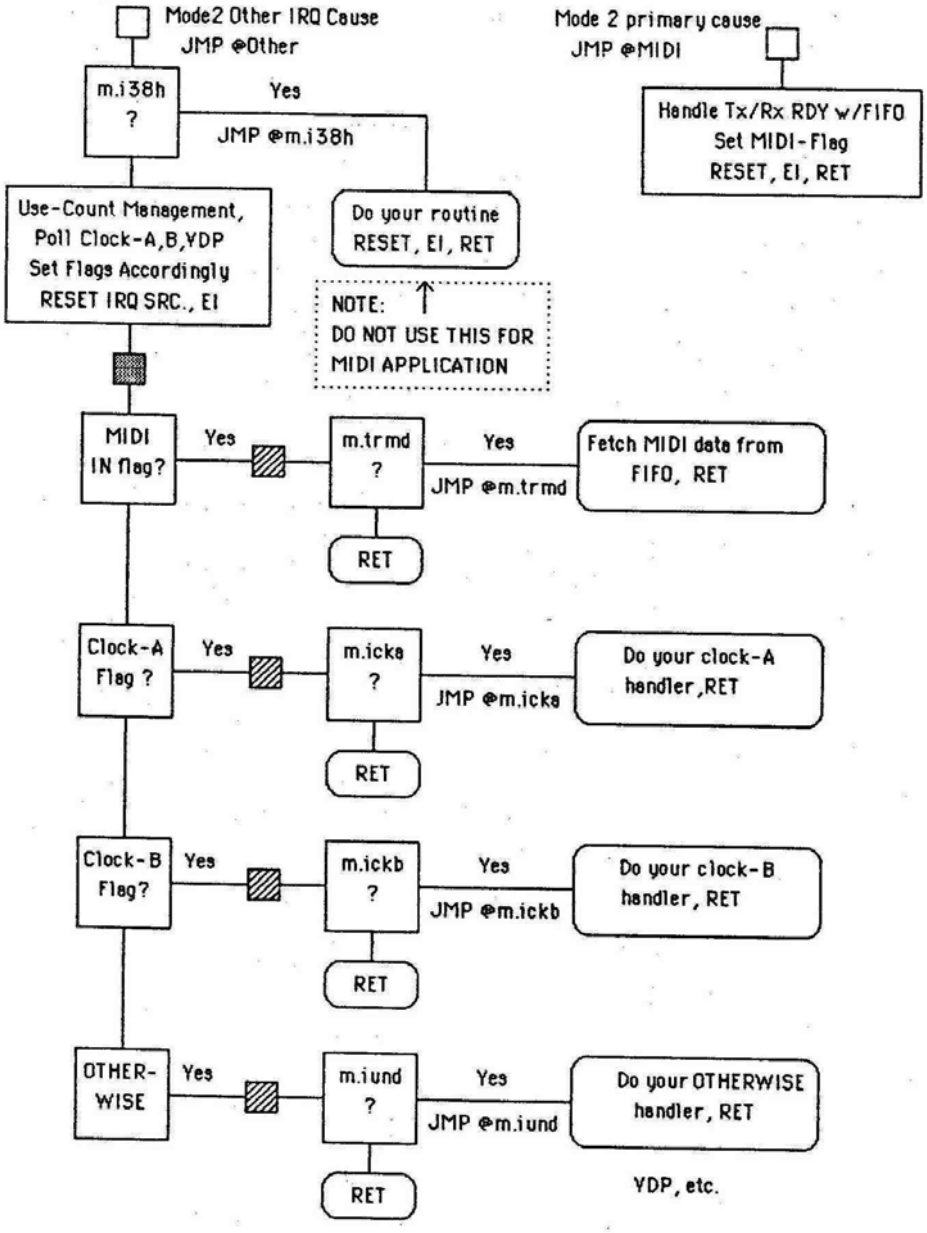


FIG. 3.2-B IRQ control Flow in Operating mode 2.0

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 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 and <M.TRER> for error.
<M.TRMK> is a vector to transfer the key-on/off information (which is asynchronous to the user routine by nature) from MBIOS. Music keyboard should be scanned . Hence, <M.TRMK> should be used together with service call K-01 (scan MK).

Since AST may be generated right in the middle of an SV-call, in operating mode 1.0/1.1, issuing another SV-call or enabling the interrupt is not allowed in the AST handler.
However, in operating mode 2.0, the system assumes that interrupt is left enabled even when trap is occurred. However, in AST, no SV-call but F-call should be requested. BDOS-call should not be called either.

When returning from the AST routine, issue a <RET>.
Restoring the registers is not necessary.
In the following figure, typical AST usage is depicted.

Register contents when AST is invoked:

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

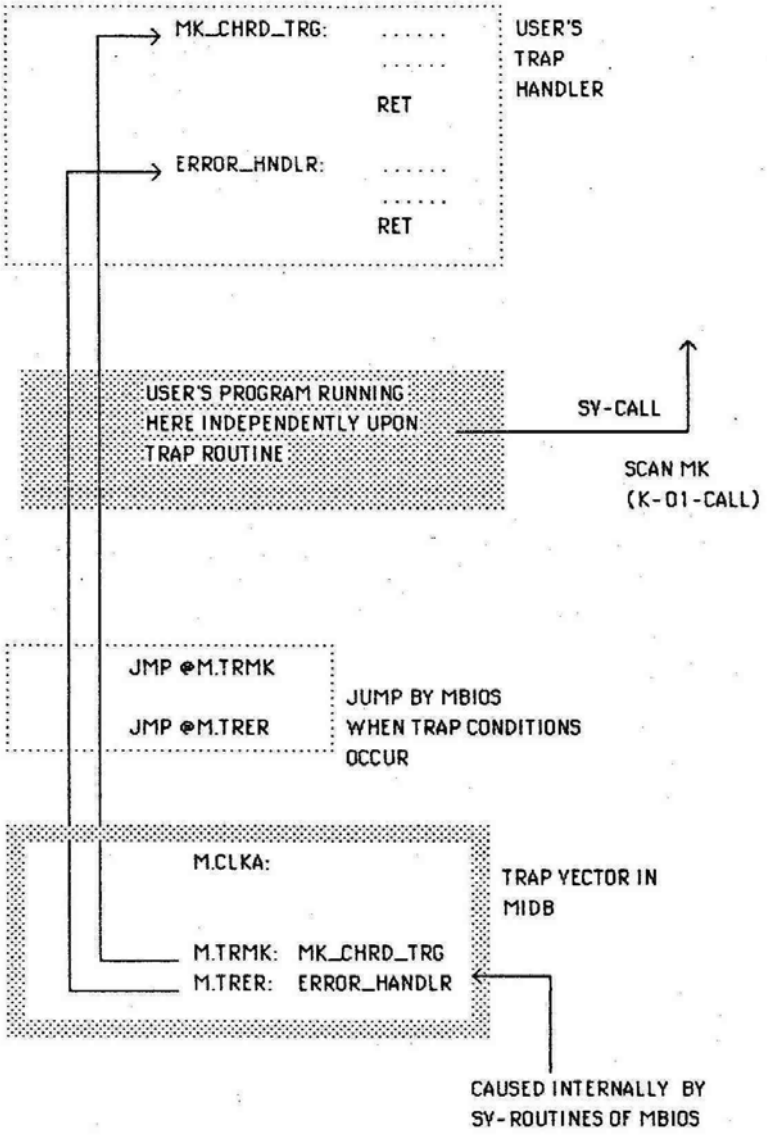


FIG. 3.5 AST USAGE

There are 9 different SV-call's available, as follows:

I-call	(Initialize)
E-call	(end MIDB)
R-call	(real time)
K-call	(Music keyboard)
P-call	(Play)
S-call	(Set up)
M-call	(Receive MIDI)
F-call	(FIFO management for MIDI application)
BDOS-call	(Disk access)

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 other by waiting for the previous one to finish.
Most of the above calls 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 in the UISV interrupt routine.
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
CALL 0090h  (operating mode 1.0)
     00A8h  (operating mode 1.1)
     00ABh  (operating mode 2.0)
```

(call is returned with interrupt status being disabled).

register conditions:

	in	out
[A]	-	*
<C>	-	*
[BC]	-	*
[DE]	MIDB base address (for operating mode1.1/2.0)	*
[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

---

1 I-call is an initialization requirement for MBIOS.

2 The interupts should be disabled before I-call.

3 Three different entries for three different operating modes. (This conforms to the compatibility to the program written on SFG-01).

4 I-call 1.0 (operating mode 1.0):
-Compatible to MBIOS of SFG-01.

-Fixed address MIDB and MBIOS work.
-sets IRQ mode 1.

5 I-call 1.1 (operating mode 1.1):
-Base address of MIDB is set anywhere in RAM.
-sets IRQ mode1.

6 I-call 2.0 (operating mode 2.0):
-Base address of MIDB is set anywhere in RAM.
-sets IRQ mode 2.

7 In processing of I-call, MBIOS grabs and intializes BASIC's PLAY buffer (between F975h and F9F5h) for necessary indirect addressing of MIDB.
MIDB also alters the PLAY hook to NOP so as to prevent PLAY queue area from being accidentally accesed by PLAY entry.
It will be freed when END-call is issued.

[E-call (End MBIOS)]

Calling sequence:

```
DI
CALL      00Bth
(call is returned with interrrupt status being disabled)
```

Registers:

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

---

1 Closes MBIOS and disables all the interrupt sources of SFG-05.

2 Frees queue buffer area of PLAY by BASIC routine.
(The queue buffer between F975h and F9F5h has been used as fixed address pointer area for indirect addressing of MIDB and MBIOS work).

3 The interrupt mode is reset to mode-1.

[R-call (Real time)]

Calling sequence:

Opearating mode 1.0/1.1:
```
DI
CALL  0093h (or 0008h)
```
(call is returned with interrupt staus being disabled)

Operating mode 2.0:
```
EI
CALL  0093h (or 0008h)
```
(call is returned with interrupt status being enabled)

Registers:

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

---

1 R-call is a real time processing call.

2 The functions of R-calls involve generation of events and clocks.

3 R-call can be issued in UISV's.

4 R-call can be issued in AST-01 but not in AST-02.

5 Run time error, if detected, will be indicated by the ‹C› flag.

[K-Call (Music keyboard)]

Calling sequence:

Operating mode 1.0/1.1:
DI (or EI)
CALL 0096h
(call is returned with interrupt status being same as it entered)

Operating mode 2.0:
EI
CALL 0096h
(call is returned with interrupt status being enabled)

Registers:

	in	out
[A]	func#	0
<C>	-	busy
[BC]	arg	*
[DE]	arg	*
[HL]	arg	*
[IX/IY]	-	o
[alternate R.]	-	o

---

1 K-call is used for the initialization and scanning of MK.

2 The busy condition (<C>=1) occurs when a K-call is issued before the previous K-call has been completed.

3 K-call can be issued in UISV traps.

4 K-call can not be issued in AST's.

[P-Call (Play)]

Calling sequence:

Operating mode 1.0/1.1:
```
DI/EI
CALL    0099h
(call is returned with interrupt status being same as it
 entered)
```

Operating mode 2.0:
```
EI
CALL    0099h
(call is returned with interrupt staus being enabled)
```

Registers:

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

---

1 P-call retrieves events from the queue and plays them using the corresponding IDB.

2 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>.

3 P-call can be issued in UISV traps.

4 P-call can not be issued in AST's.

[S-Call (Set up)]

Calling sequence:

Operating mode 1.0/1.1:

```
DI
CALL    009Ch (or 0010h)
(call is returned with interrupt status being diabled)
```

Operating mode 2.0:

```
EI
CALL    009Ch (or 0010h)
(call is returned with interrupt status being enabled)
```

Registers:

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

---

1 S-call is a request that does not require real time processing.

2 <C> =1 indicates that error has been detected.

3 IF <C>=1 and;

A=00h , then busy condition (P-call, K-call or another S-call is busy).
A is non-zero, then other error was detected.
For detail, see individual S-call syntax.

4 S-call can be issued in UISV traps.

5 S-call can not be issued in AST's.

[M-call (Receive MIDI)]

Calling sequence:

Operating mode 1.0/1.1:

```
DI
CALL   00A5h
```

(call is returned with interrupt status being disabled)

Opearating mode 2.0:

```
EI
CALL   00A5h
```

(call is returned with interrupt status being enabled)

Registers:

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

---

1 M-call scans the MIDI input port or MIDI-FIFO.

2 If data is present at the port, it fetches the data in the D register.

3 E register contains, when returned, MIDI interface status.

4 M-call can be issued in UISV traps and AST's.

[F-Call (FIFO management)]

Calling sequence:

Available only in operating mode 2.0:
EI
CALL 00B1h
(call is returned with interrupt status being enabled)

Registers:

	in	out
[A]	func#	*
<C>	-	*
[BC]	parameter	*
[DE]	parameter	*
[HL]	-	*
[IX/IY]	-	o

---

1 F-00 clears the reception FIFO buffers for MIDI communication.

2 F-01clears the transmission FIFO buffers for MIDI communication.

3 FIFO buffers can be defined by S-05 call.

4 F-02 is to reset the error flag of MIDI communication channel, occured during the MIDI reception.

5 F-03 is a function to send MIDI-real-time-data (F8h-FFh). This is equivalent to R-21 call.

6 F-call can be issued in UISV traps and AST's.

[BDOS-call (Disk access routines)]

Calling sequence:

operating mode 1.1
DI
CALL 00B4h
(call is returned with interrupt status being disabled)

Operating mode 2.0
EI
CALL 00B4h
(call is returned with interrupt status being enabled)

Registers

	in	out
[A]	see BASIC	status
<C>	_	<0> normal <1> illegal call
[BC]	see BASIC	see BASIC/substatus
[DE]	see BASIC	see BASIC
[HL]	see BASIC	see BASIC

---

1. MBIOS disables IRQ state in [h.kei] routine.

2. Illegal call conditions

   -MBIOS is already performing BDOS-call.
   -Disk does not exist.
   -MBIOS is already performing SV-call or UISV.

3. Status
   -FEh; disk error (with sub status)
   -other ; see BDOS status manual

4. Substatus
   <01h> write protected
   <02/03h> drive not ready (R/W)

<04/05h> CRC error (R/W)
<06/07h> seek error (R/W)
<08/09h> record not found (R/W)

5 BDOS-call can not be issued in UISV or AST.

CHAPTER IV MBIOS Syntax

4 - 1 I - call

(Initialize MBIOS: Opearating mode 1.0)

Entry address: 0090h

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

1 This is a compatible mode to I-call of SFG-01.
Necessary MBIOS work is allocated to the same memory address as that fixed to SFG-01.

2 DI prior to I-call.

3 Although in this mode, the base address of MIDB is fixed to EC00h, The internal addressing to MIDB still uses indirect addressing using 128 bytes of BASIC's queue buffer (F975h) for pointers.

To prevent this buffer from being accidentally used, BASIC's PLAY entry is deactivated by I-call processing.

4 Initialization includes the following operation.

i Initialize the MIDB.
ii Clear AST and USIV tables.
iii Clear IDB, UVL, and EVB buffers.
iv Set Z80 interrupt mode to mode 1.
v Enable clock-A, and clock-B.
vi Initialize PLAY-queue area with pointers
v Load 00h into MUSICF (FB3Fh), and replace hook H.PLAY (FFC5h) with:

```
pop  hl
ret
```

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.amd:	00h
PMD	m.pmd:	00h
Noise	m.nois:	00h (disabled)
UISV table	m:i38h:	all 00h
AST table	m.trmk:	all 00h

System status:

Default RHB pointed	RHB#0
IDB's	all cleared
EVB and UVL	all cleared
Brilliance	FFh
Pitchbend	00h

I - call

(Initialize MBIOS: Opearating mode 1.1)

Entry address:	00A8h	
Registers:		
	In	Out
[A]	-	*
<C>	-	-
[BC]	-	*
[DE]	MIDB base address	*
[HL]	-	*

1 This is an entry to operating mode 1.1.
In operating mode 1.1, MBIOS work (MIDB) can be allocated anywhere in ram.

2 Z80's interrupt mode is set to mode 1.

3 In order to implement indirect addressing to MIDB, MBIOS uses 128 bytes of BAISC's queue buffer (F975h) as a pointer bank. To prevent this buffer from being accidentally used, BASIC's PLAY entry is deactivated by I-call processing.

4. Initialization includes the following operation.

i Define and initialize the MIDB (780h bytes).
ii Clear AST and USIV tables.
iii Clear IDB, UVL, and EVB buffers.
iv Set up pointers into PLAY-queue buffer.
v Load 00h into MUSICF (FB3Fh), and replace hook H.PLAY (FFC5h) with:

```
pop  hl
ret
```

vi Set Z80 interrupt mode to mode 1.
vii Enable clock-A, and clock-B.

I - call

(Initialize MBIOS: Opearating mode 2.0)

Entry address: 00ABh

Registers:

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

---

1 This is an entry to operating mode 2.0

2 In operating mode 2.0, MBIOS work (MIDB) can be allocated anywhere in ram.

3 Z80's interrupt mode is set to be mode 2.

3 In order to implement indirect addressing to MIDB, MBIOS uses 128 bytes of BASIC's queue buffer (F975h) as a pointer bank. To prevent this buffer from being accidentally used, BASIC's PLAY entry is deactivated by I-call processing.

4 Initialization includes the following operation.

i Define and initialize the MIDB (780h bytes).
ii Clear AST and USIV tables.
iii Clear IDB, UVL, and EVB buffers.
iv Set up pointers for indirect addressing of MIDB into BASIC's PLAY-queue buffer.
v Load 00h into MUSICF (FB3Fh), and replace hook H.PLAY (FFC5h) with:

```
pop  hl
ret
```

vi Set Z80 interrupt mode to mode 2.
vii Set internal mode-2 vector.
viii Enable clock-A, clock-B, and UART.

[4-2] END-call

(Shut down MBIOS : operating mode 1.1 and 2.0)

Entry address: 00B7h

Registers:	in	out
[A]	-	*
<C>	-	*
[BC]	-	*
[DE]	-	*
[HL]	-	*
[IX/IY]	-	*

1 This resets MBIOS to normal MSX operating environment.
-Frees BASIC's queue buffer
-resets PLAY hook to normal
-set Z80's interrupt mode to mode1

2 do not issue END-call while END-call has already been issued.

R - 00 System All-note-off (damp)

(This issues All-note-off events into all 7 Queues and damp all the engaged channels)

Entry address: 0093h

Registers:

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

R - 01 All-note-off

(Issues All-note-off event to designated Queue)

Entry address:		0093h	
Registers			
	[A]	01h	00h
	<C>		<0>
	[BC]	QU#/-	*
	[DE]	-	*
	[HL]	-	*

1 QU# [B] [00000xxx] xxx=0 - 7 ,QU#

R - 02 Set Event into Queue

(Sets event into designated Queue)

Entry address: 0093h

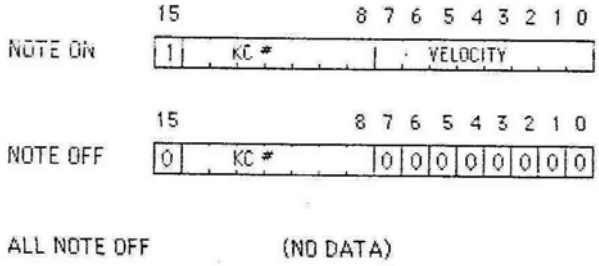
Registers:

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

1 QU# [B]
[00000xxx] xxx=0 - 7 ;QU#

2 An error <C> will be set when the QUEUE is already full, and the corresponding event will not be registered into the Queue.

3 The event format:



ALL NOTE OFF (NO DATA)

R - 08 Start Recording

(Start recording from the designated Queue to the EVB)

Entry address: 0093h

Regisiters:

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

1 QU# [B]
[10000xxx] xxx=0 - 7, QU#

2 This will be ignored when the EVB is undefined.

R - 09 Set Recording Clock

(This provides the timing clock for recording).

Entry address: 0093h

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 in each interrupt handler routine.

R - 0A Stop Recording

(This stops the recording from Queue)

Entry address: 0093h

Registers:

[A]	0Ah	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 any other mode than recording.

R - 0B Start Playback

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

Entry address: 0093h

Registers:

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

1 QU# [B]
[10000xxx] xxx=0 - 7 ;QU#

2 This will be ignored when the EVB is undefined, or while playback/ recording is already busy.

R - 0C Set Playback Clock

(This provides the timing clock for playback).

Entry address: 0093h

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 each clock interrupt routine.

R - 0D Stop Playback

(This stops the playback).

Entry address: 0093h

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 in any other mode than playback.

R-18 Load LFO

(Load LFO parameters into the FM sound generator IC).

Entry address: 0093h

Registers:

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

1 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).

Entry address: 0093h

	in	out
[A]	19h	00h
‹C›	-	‹0›
[BC]	-	*
[DE]	-	*
[HL]	-	*

1 It is used to load KC information to currently engaged channel.

2 For portamento application:

Issue this command in synchronization with CLOCK-A.
Portamento rate information stored in currently engaged IDB will be added to the currently accumulated KC information and be loaded into assigned channel of the chip.
This way, KC_pitch of the instrument will be updated with given pitch interval at time interval rate of CLOCK-A.

3 For transposition:

Load MIDB or IDB appropriately with transposing information, and issue this command once.

4 for pitchbend:

Load MIDB or IDB appropriately with pitch information, and issue this commands succeedingly with appropriate interval.

R-20 Send MIDI message

(Outputs a given MIDI message MIDI-FIFO for transmission).

Entry address: 0093h

Registers:

	in	out
[A]	20h	00h
<C>	-	<0>
[BC]	message size	*
[DE]	message address	*
[HL]	-	*

1 This transmits a block of data specified by size and its buffer address into MIDI-FIFO.

2 This is generally used to transmit one complete MIDI message packet at once into MIDI- FIFO.

3 Care must be taken when non-MIDI packet is to be sent.
For there might be a conflict with other requesting routine for the same FIFO.
This call makes no attempt to merge more than one messages, that are requesting the same FIFO, into one message stream.

4 If FIFO is full, it waits until the FIFO becomes ready.

5 If FIFO is not defined, it transmits a block of data from the MIDI-port directly.

6 If the port is busy, it waits until the port becomes ready.

E-21 Send a byte to MIDI

(Outputs a given single byte of MIDIdata into MIDI-FIFO for transmission).

Entry address: 0093h

Registers:

	in	out
[A]	21h	00h
<C>	-	<0>
[BC]	-/data	*
[DE]	-	*
[HL]	-	*

1 This sends a byte of data to FIFO for transmission.

2 This is usally used to trnsmit only real time data between F8h and FFh.

3 Care must be taken when a byte data between 00h and F7 is to be sent.
For there might be other routine which is using the same FIFO for its own data transmission.
This routine makes no attempt to merge more than one MIDI messages into a consistent MIDI message stream when multiple number of FIFO requesting sources collide.

4 If FIFO is full, it waits until the FIFO becomes ready.

5 When no FIFO is defined, access is made to the MIDI port directly.
If port is busy, it awaits until the port becomes ready.

[4-4] K-Call

K-00 Init MK

(Initializes MK, sets velocity for MK,
and establishes the link up between the MK and the specified queue).

Entry address: 0096h

Registers:

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

1 link [B] [x0000yyy]

x=<0> no link with queue
x=<1> link with queue
yyy=0-7 , QU#

2 mode [C] [0000000] always 0.

3 Velocity [D] [xxxxxxxx]

xxxxxxxx=00h (minimum velocity) to
=FFh (maximum velocity)

[illegible]-01 Scan MK

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

Entry address: 0096h

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, by each interrupt cycle.

3 If AST vectors for MK has been specified in the MIDB, it will cause AST trapping via AST vectors.

K-02 Report MK status

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

Entry address: 0096h

Registers:

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

1 The buffer contents will be comprised of 9 bytes as shown below. Each bit represents the on/off status of corresponding key position.

	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 queues, and plays them}

Entry address: 0099h

Registers:

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

1 Queue-map [DE]
[00000xxxhgfedcba]

xxx ; reserved (fill zero's).

a QU#0
b QU#1
.....
h QU#7

2 PLAY processes all the event requests in the assigned queues by retrieving and playing accordingly(note-on/off) one event by one until the queue gets empty.

S - 00 Define IDB

(Either defines or cancels the IDB).

Entry address: 009Ch

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.

3 The size of IDB is 128 bytes.

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	C0h
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)).

Entry address: 009Ch

Registers:

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

1 The EVB is canceled 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))

Entry address: 009Ch

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)).

Entry address: 009Ch

Registers:

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

S-05 Define FIFO

(This defines FIFO buffer for MIDI transmitter/receiver)

Entry address: 009Ch

Registers:

	in	out
[A]	05h	00h
<C>	-	busy
[BC]	FIFO/Tr address	*
[DE]	FIFO/Re address	*
[HL]	-	*

1 This SV-call includes the function of S-0C (initialize MIDI port). If this is used in operating mode 1.1, the effect is same as that of S-0C call.

2 The size of FIFO buffer is 100h bytes each for FIFO/Tr and FIFO/Re.

3 FIFO/Tr or FIFO/Re will be cancelled if 0000h is specified as FIFO address.

4 Interrupt RxRDY IRQ is enabled only while FIFO/Re is defined.

5 Interrupt TxRDY IRQ is enabled only while FIFO/Tr is defined.

S - 09 Assign channel

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

Entry address: 009Ch

Registers:

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

1 [BC]
[xxxxyyyyzzzztttt]
xxxx=0 - 7 ; IDB# assigned to channel 0
yyyy=0 - 7 ; IDB# assigned to channel 1
zzzz= 0 - 7 ; IDB# assigned to channel 2
tttt= 0 - 7 ; IDB# assigned to channel 3
[DE]
[ppppqqqqrrrrssss]
pppp=0 - 7 ; IDB# assigned to channel 4
qqqq=0 - 7 ; IDB# assigned to channel 5
rrrr= 0 - 7 ; IDB# assigned to channel 6
ssss= 0 - 7 ; IDB# assigned to channel 7

2 IDB# 8h - Fh are reserved for future usage.

3 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 IDB).

Entry address: 009Ch

Registers:

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

1 Queuelink [D]
[10000xxx] xxx= 0 - 7 ; QU#

MIDIlink [E]
[x000yyyy] x=<1> ; MIDI is assigned
x=<0> ; MIDI is not assigned
yyyy=0 - Fh ;MIDI channel #

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

3 Only MIDI channel (without using any channel of FM LSI) can be assigned.

S - 0B All-Note-Off by IDB

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

Entry address: 009Ch

Registers:

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

1 IDB# [B] [00000xxx] xxx= 0 - 7 ;IDB#

S - 0C Initialize MIDI

(This initializes the MIDI port).

Entry address: 009Ch

Registers:

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

1 In operating mode 1.0/1.1, this routine disables both the RxRDY and TxRDY interrupts of MIDI.

Thus in operating mode 1.0/1.1, MIDI is driven only under non-interrupt condition.

S-0D Enable/disable OPM interrupt

(This allows the user to freely control the interrupts of clock-A/B)

Entry address: 009Ch

Registers:

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

1 Mode [C]
[000000yx] x=1 ;enable clock-A
y=1 ;enable clock-B

S-0E Select UISV priority mode

(This selects polling priority mode of UISV in operating mode 2.0)

Entry address: 009Ch

Registers:

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

1 Mode [C]
[0000000x] x=0 MIDI > clock-A > clock-B > VDP
x=1 MIDI > clock-B > clock-A > VDP

S - 10 Set Brilliance

(This sets the system parameter , Brilliance).

Registers:

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

1 Brilliance [C]
[xxxxxxxx] xxxxxxxx = 00h (dark) to FFh (brightest)

2 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]	-	*

1 Pitchbend [C]
[xxxxxxxx] xxxxxxxx = -128 to +127 (2's complement)
xxxxxxxx = 00h 0%
80h(-128) -100%
7Fh (127) +100%

2 As in portamento, to realize pitchbend, an R-19 call (updating KC information) should be repeatedly issued in syncronization to each A-clock interrupt.

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]	-	*

1 IDB# [B]

[00000xxx] xxx= 0 - 7; IDB#

2 Mode [C]

[0000tzyx]

x=<1>	sustain on
y=<1>	single triggered
y=<0>	multiple triggered
z=<1>	full portamento
z=<0>	fingered portamento
t=<1>	disable pitchbend
t=<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]	-	*

1 IDB# [B] [00000xxx] xxx=0 - 7 ; IDB#

2 Volume [C] [xxxxxxxx] xxxxxxxx= 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]	-	*

1 IDB# [B]
[0000xxxx] xxx=0 - 7 ; IDB#

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#*	
[DE]	-	*
[HL]	-	*

1 IDB# [B]
[00000xxx] xxx=0 - 7 ;IDB#

2 Voice# [C]
[xxxxxxxx] xxxxxxxx=00h - 2Fh for SVL
4oh - 6Fh for UVL

3 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]	-	*

1 IDB# [B]
[00000xxx] xxx= 0 - 7 ;IDB#

2 Voice# [C]
[xxxxxxxx] xxxxxxxx= 00h - 2Fh for SVL
40h - 6Fh for UVL

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

S-17 Set stereo L/R

(This sets L/R control of currently engaged voice with IDB)

Entry address: 009Ch

Registers:

	in	out
[A]	17h	ooh
<C>	-	busy
[BC]	IDB#/data	*
[DE]	-	*
[HL]	-	*

1 data [C] [yx000000] x=1 ;enable left
y=1 ;enable right

S-18 Set pms/ams

(This sets pms/ams to currently engaged voice with IDB)

Entry address: 009Ch

Registers:

	in	out
[A]	18h	ooh
<C>	-	busy
[BC]	IDB#/data	*
[DE]	-	*
[HL]	-	*

1 data [C]

	in	out
[0yyy00xx]	xx	pms
	yyy	ams

S-19 Get & load voice

(This gets and loads voice into channels hooked up with IDB. Same as issuing S-14 after S-15).

Entry address: 009Ch

Registers:

	in	out
[A]	19h	ooh
<C>	-	busy
[BC]	IDB#/voice#	*
[DE]	-	*
[HL]	-	*

Entry address: 009Ch

Registers:

	in	out
[A]	20h	error location
<C>	-	error
[BC]	-	-/error status
[DE]	control block address	*
[HL]	-	*
[Alternate Rg's]		destroyed

1 This call should not be issued in UISV.

2 Control block specification
-17 byte control block

.byte access code#
00h= read file
01h= write file
02h= read UVL
03h= write UVL

.byte device#
00h= CMT
10h= s/ram (data cartridge)
20h= default disk drive
21h= A:
22h= B:

.word buffer address for access code# 00h/01h

.word buffer size for access code#00h/01h

.blkb 8 (8 byte file name string)
device#00h & access code#0/1 ---6 characters
device#00h & access code#2/3 -- "VOICE " used by MBIOS
device#10h --- not used
device#20h/21h/22h --- 8 characters

.blkb 3 (3 byte file name extension)
device#00h --- not used
device#10h --- not used
device#20h/21h/22h --- 3characters
("VOG" as default by MBIOS,
for acs-code#2/3)

3 <C> =1 indicates error has been detected.

4 Error status [C]

01h=	disk i/o error (write protected)
02h=	disk i/o error (drive not ready)
03h=	disk i/o error (crc error)
04h=	disk i/o error (seek error)
05h=	disk i/o error (record not found)
06h=	disk i/o error (write fault)
07h=	i/o error (undefined disk i/o error, cmt i/o error, s-ram i/o error)
08h=	BDOS error
09h=	invalid parameter (invalid access code, device#)
0Ah=	invalid file type error (invalid BASIC id#, invalid YAMAHA id#)
0Bh=	file body size is larger than buffer
0Ch=	DOS is busy

5 Error location [A]

01h=	error has occured before physical access
02h=	open error
03h=	read error in BASIC's header file
04h=	read error in file body
05h=	close file error
06h=	create file error
07h=	write error in BASIC's header file
08h=	write error in file body

S - 21 Read UVL

(This reads in the UVL from the CMT).

Entry address: 009Ch

Registers:

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

1 <C> =1 indicates error has been detected.
If so, further information is provided by [A].

2 Error# [A]

[xxxxxxxx]	xxxxxxxx=00h	Normal end
	FFh	Size Error
	Other non-0#	MSX-BASIC error

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

4 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).

Entry address: 009Ch

Registers:

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

1 <C>=1 indicates error has been detected.
Further information will be provided by [A].

2 Error# [A]

[xxxxxxxx]	xxxxxxxx=00h	Normal end
	non-0#	MSX-BASIC Error

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

4 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).

Entry address: 009Ch

Registers:

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

1 <C>=1 indicates error has been detected.
Further information will be provided by [A].

2 Error# [A]

[xxxxxxxx]	xxxxxxxx=00h	Normal end
	non-0#	MSX-BASIC Error

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

4 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).

Entry address: 009Ch

Registers:

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

1 <C>=1 indicates error has been detected.
Further information will be provided by [A].

2 Error# [A]

[xxxxxxxx]	xxxxxxxx=00h	Normal end
	non-0#	MSX-BASIC Error

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

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

(This will call the CSM vocal synthesis driver).

Entry address: 009Ch

Registers:

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

1 <C>=1 indicates error has been detected.
Further information will be provided by [A].

2 Error# [A]

[xxxxxxxx]	xxxxxxxx=00h	Normal end
	01h	run-time error

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

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

4-6 M - call

M Receive MIDI

(In operating mode 1.0/1.1, this scans the input port of the MIDI interface, and returns the data if there is a data).
(In operating mode 2.0, data is retrieved from FIFO/Re buffer)

Entry address: 00A5h

Registers:

	in	out
[A]	-	*
<C>	-	*
[BC]	-	0
[DE]	-	data/staus
[HL]	-	0

1 Status [E] [00zy00x0]

x	RxRDY
y	Overrun error (op 1.0/1.1) FIFO/Re overflow (op 2.0)
z	Framing error

4-7 F-Call

F-00 Clear FIFO/Re

(This clears FIFO/Re buffer and resets error flag)

Entry address: 00B1h

Registers:

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

1 Error flag is reset.

F-01 Clear FIFO/Tr

(This clears FIFO/Tr buffer)

Entry address: 00B1h

Registers:

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

1 Error flag is reset.

F-02 Reset Error Flag

(This resets error flag of MIDI interface register)

Entry address: 00B1h

Registers:

	in	out
[A]	02h	*
<C>	-	*
[BC]	-	*
[DE]	-	*
[HL]	-	*

F-03 Send a byte of MIDI data

(This sends a byte of MIDI data to FIFO/Tr)

Entry address: 00B1h

Registers:

	in	out
[A]	03h	*
<C>	-	*
[BC]	-/data	*
[DE]	-	*
[HL]	-	*

1 This is equivalent to R-21, but is faster.

2 Normally this is used to send MIDI real time data between F8h and FFh.

3 The same care must be taken if a data between 00h and F7h is to be trasmitted.

4 If FIFO/Tr is full, it waits until FIFO/Tr becomes available to send.

5 If no FIFO is defined, it sends a data directly to MIDI port.

(Disk access routine)

Entry address: 00B4h

Registers

	in	out
[A]	see BASIC	status
<C>	_	<0> normal <1> illegal call
[BC]	see BASIC	see BASIC/substatus
[DE]	see BASIC	see BASIC
[HL]	see BASIC	see BASIC

---

1 In operating mode 1.1, interrupt should be disabled before and after BDOS-call.

2 In operating mode 2.0, interrupt can be enabled before the call. If it was called with EI, call is returned with interrupt being enabled.

3 MBIOS disables IRQ state in [h.kei] routine.

4 Illegal call conditions

-MBIOS is already performing BDOS-call.
-Disk does not exist.
-MBIOS is already performing SV-call or UISV.

5 Status
-FEh; disk error (with sub status)
-other ; see BDOS status manual

6 Substatus

<01h>	write protected
<02/03h>	drive not ready (R/W)
<04/05h>	CRC error (R/W)
<06/07h>	seek error (R/W)
<08/09h>	record not found (R/W)

7 BDOS-call can not be issued in UISV or AST.

4-9 AST (Asynchronous System Trap)

AST - 01 MK trigger

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

MIDB vector: offset 3Ch (m.trmk)

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.
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).

MIDB vector:		offset m.trer (3Eh)
Registers:		
		out
	[A]	02h
	<C>	-
	[BC]	-
	[DE]	error#/-
	[HL]	-

1 Error [D]

[00010xxx]	QU# has overflowed. xxx= 0 - 7 ;QU#
[00100000]	EVB is full (end of recording, All-Note-Off issued)
[00100001]	EVB end is encountered (end of playback)
[0011xxxx]	MIDI time-out error xxxx= 0 - Fh ;MIDI channel#

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

4-10 UISV trap

(This is an interrupt trap caused by user defined UISV vector)

MIDB vectors:	offset 30h (m.i38h) offset 32h (m.icka) offset 34h (m.ickb) offset 36h (m.iund) offset 3Ah (m.irmd)

Registers:

[A]	stacking count
<C>	*
[BC]	*
[DE]	*
[HL]	*
[IX/IY]	*

1 In operating mode 2.0, when called via UISV, [A] contains stacking count of IRQ's.
For <m.trmd> call, however, stacking count is not maintained.

2 In operating mode 1.0/1.1, JMP is used to reach user's routine via UISV.
In operating mode 1.0/1.1, IRQ should not be enabled until RET.

3 In opearting mode 2.0, CALL is used to reach user's routine via UISV.
In operating mode 2.0, IRQ should be left enabled.

Appendix SFG-05:

MBIOS supplementary reference

date: Oct,26,1985
Dec,05,1985

compatibility legend:

012	Used in operating mode 1.0/1.1/2.0
01	Used in operating mode 1.0/1.1
12	Used in operating mode 1.1/2.0
2	Used in operating mode 2.0
(0e)	If used in SFG-01, MBIOS may crash.
(2e)	If used in operating mode 2.0, MBIOS may crash.
*	Command remains same. However some defference in usage and meaning exist.

[1]. List of important memory address

id area:

0080h	"MCHFM0"	(ASCII 6 chr.)
0086h	---- ----	ROM id#
0087h	---- ----	Hardware type#
0088h	---- ----	Software version#

SFG-01:	0080h	"MCHFM0"
	0086h	.....
	0087h	..<00h>..
	0088h	..<0Xh>..

SFG-05:	0080h	"MCHFM0"
	0086h	.....
	0087h	..<00h>..
	0088h	..<1Xh>..

entry address:

000Ch	RDSLT	12
0014h	WRSLT	12
001Ch	CALSLT	12
0024h	ENASLT	12
0028h	CALLF	12(0e)
003Ah	RDPP.0	12(0e)
003Dh	EVCNV0	12(0e)
00C0h	EVCNV1	12(0e)
0090h	I-call (1.0)	012
00A6h	I-call (1.1)	12(0e)
00A5h	I-call (2.0)	12(0e)
0087h	END-call	12(0e)
0042h	M-Monitor 1.0	012
00AEh	M-Monitor 2.0	12(0e)
0084h	BDOS-call	12(0e)
0093h	R-call	012
0096h	K-call	012
0099h	P-call	012
009Ch	S-call	012
00A5h	M-call	012
0031h	F-call	2(0e)
009Fh	IRQC	01 (2e)
0008h	R-call	12(0e)
0010h	S-call	12(0e)

[2]. List of SV-calls

R-call:

R-00	Damp	012
R-01	All Note Off (by Queue)	012
R-02	Set event into Queue	012
R-03	(reserved)	
R-04	Set event into Chord-K3	012
R-05	Set Chord# into Chord-K3	012
R-06	-	
R-07	-	
R-08	Start Recording	012
R-09	Set Recording Clock	012
R-0A	Stop Recording	012
R-0B	Start Playback	012
R-0C	Set Playback Clock	012
R-0D	Stop Playback	012
R-0E	(reserved)	
R-0F	-	
R-10	Start Auto Rhythm	012
R-11	Set Auto Rhythm Clock	012
R-12	Stop Auto Rhythm	012
R-13	Select Auto Rhythm Queue	012
R-14	Select RHB	012
R-15	-	
R-16	-	
R-17	-	
R-18	Load LFO	012
R-19	Load KC	012
R-1A	(reserved)	
R-1B	(reserved)	
R-1C	-	
R-1D	(reserved)	
R-1E	-	
R-1F	-	
R-20	Send "MIDI message"	12
R-21	Send "MIDI real time message"	012

K-call:

K-00	Init MK	012
K-01	Scan MK	012
K-02	Report MK status	012

S-call:

S-00	Define IDB	012
S-01	-	
S-02	Define EVB	012
S-03	Define UVL	012
S-04	Initialize EVB	012
S-05	Define FIFO	12*
S-06	-	
S-07	-	
S-08	(reserved)	
S-09	Assign Channel	012
S-0A	Assign Queue and MIDI channel	012
S-0B	All Note Off (by IDB)	012
S-0C	Initialize MIDI	012
S-0D	Enable/Disable OPM-IRQ	12
S-0E	Select UISV priority mode	2
S-0F	-	
S-10	Set Brilliance	012
S-11	Set Pitchbend	012
S-12	Define Play mode	012
S-13	Set Volume	012
S-14	Load Voice	012
S-15	Get Voice	012
S-16	Put Voice	012
S-17	Set stereo L/R	12
S-18	Set AMS, PMS	12
S-19	Get & Load Voice	12
S-1A	-	
S-1B	-	
S-1C	-	
S-1D	-	
S-1E	-	
S-1F	-	
S-20	File driver	12
S-21	Read UVL	012
S-22	Write UVL	012
S-23	Read EVB	012
S-24	Write EVB	012
S-25	-	
S-26	-	
S-27	-	
S-28	CSM Voicing	012

F-call:

F-00	Clear FIFO/Re	2(0e)
F-01	Clear FIFO/Tr	2(0e)
F-02	Reset error flag	2(0e)
F-03	Send "real time message"	2(0e)

[3]. MIDI compatibility

```
-------------------------------------------------------------------------
        m.sram   28h[R]  s/ram size                        12
        m.f8     2Ah[W]  thru ( timer ---> F8h )            2
                         [0000 00--]  <1> timer/a ---> MIDI(out)
                         [0000 00*-]  <1> timer/b ---> MIDI(out)
        -
        m.i38h   30h[W]                                    01
        m.icka   32h[W]                                    012
        m.ickb   34h[W]                                    012
        -
        m.iund   38h[W]                                    012
        m.trmd   3Ah[W]  data exists in FIFO/Re            2
        m.trmk   3Ch[W]  AST#1                             012
                         AST#3                             012
        m.trer   3Eh[W]  AST#2                             012
        -
        m.thru      [W]  MIDI/thru table                   2
                 F8h     [0000 000*]  <1> enable F8h thru
                 ...     ...
                 FFh     [0000 000*]  <1> enable FFh thru
-------------------------------------------------------------------------
```

[4]. memory, slot map

operating mode 1.0:

Memory map:	0000h - 3FFFh	SFG-05 rom
	4000h - 7FFFh	user's area
	(bottom)- EBFFh	interface area (IDB, UVL, stack, ...)
	EC00h - F37Fh	MIDB & bios' work
	F380h - F974h	basic work area
	F975h - F9F4h	bios work area
	F9F5h - FAF4h	user's area
	FAF5h - FFFFh	basic work area

operating mode 1.1/2.0:

Memory map:	0000h - 3FFFh	SFG-05 rom
	4000h - 7FFFh	user's area
	(bottom)-(himem)	interface area (MIDB, IDB, UVL, stack, ...)
	(himem)- F974h	basic work area
	F975h - F9F4h	bios work area
	F9F5h - FAF4h	user's area
	FAF5h - FFFFh	basic work area

[5]. Calling Sequences (outside of m-bios repertoire)

```
rdslt:          see basic
wrslt:          see basic
calslt:         see basic
enaslt:         see basic
callf:          see basic

rdop.0:         get slot# of related address
-------------------------------------------------------------------
        <sequence>      di
                        call    009Ah
                        (IRQ/disabled when returned)

        <interface>     <in>            <return>
                [a]      -              slot#
                <c>      -                 *
                [bc]     -                 *
                [de]     -                 *
                [hl]    target address     *
                [ix/iy] -                  *

evcnv0:         convert event(opm) ---> event(MIDI)
-------------------------------------------------------------------
        <sequence>      call    00B5h

        <interface>     <in>            <return>
                [a]      -                 *
                <c>      -                 0
                [bc]    -/MIDI-ch#      -/1st-byte(9Xh)
                [de]    event(kc#/vel)  2nd-/3rd-byte
                [hl]     -                 *
                [ix]     -                 o
                [iy]     -                 o

evcnv1:         convert event(MIDI) ---> event(opm)
-------------------------------------------------------------------
        <sequence>      call    00C0h

        <interface>     <in>            <return>
                [a]      -                 *
                <c>      -              <0> valid  <1> invalid message
                [bc]    -/1st-byte      -/MIDI ch#
                [de]    2nd-/3rd-byte   event(kc#/message)
                [hl]     -                 *
                [ix]     -                 o
                [iy]     -                 o

        . 1st byte of valid message is 8Xh or 9Xh.
-------------------------------------------------------------------
```

[6]. Calling sequences (m-bios repertoire)

I-call: initialize m-bios

```
        <sequence>      di
                        call    0090h   (bios 1.0)
                                0048h   (     1.1)
                                0048h   (     2.0)
                        (IRQ/disabled when returned)

        <interface>     <in>            <return>
                [a]      -                *
                <c>      -                *
                [bc]     -                *
                [de]    MIDB address      *
                [hl]     -                *
                [ix/iy]  -                *

        <MIDB address>  used in operating mode 1.1/2.0

         This routine performs the followings.

                . Defines MIDB(780h bytes) and initializes it.
                . Protects basic's PLAY and initializes queue
                 buffer.
                        . loads 00h into MUSIC?(fb3fh).
                        . replaces H.PLAY(ffc5h) with...
                                pop  hl
                                ret
```

END-call: escape from m-bios

```
        <bios 1.1/2.0>  di
                        call    0087h
                        (IRQ disabled % im1 when returned)

        <interface>     <in>            <return>
                [a]      -                *
                <c>      -                *
                [bc]     -                *
                [de]     -                *
                [hl]     -                *
                [ix/iy]  -                *
```

. Do not issue END-call after END-call.

M-monitor: call Music Monitor

M-monitor 1.0 ... DISK function is not available
2.0 ... DISK function is available

<sequence for M-monitor 1.0>

```
execute  I-call (1.0)
execute  S-02   (define EVB)
execute  S-03   (define UVL)
  ...
di
call    00A2h
(IRQ disabled/im1)
  ...
execute  I-call
```

<sequence for M-monitor 2.0>

```
execute  I-call (1.1/2.0)
execute  S-02   (define EVB)
execute  S-03   (define UVL)
  ...
di
call    00AEh
(IRQ disabled/im1)
  ...
execute  I-call
```

<interface>	<in>	<return>
[a]	-	*
<c>	-	*
[bc]	-	*
[de]	-	*
[hl]	-	*
[ix/iy]	-	*
[alt.R]	-	*

<memory map>		
	0000h - 3FFFh	SFG-05
	4000h - 7FFFh	user's area
		... MSIOS maps M-monitor program here
	(bottom)-(himem)	interface area
	(himem)- FFFFh	basic's area
	...(780h8)..	MIDS
		... fixed address (E000h-) in M-monitor 1.0
	...(C20h8)..	UVL
	.....	evl ... more than 2000h8
	.....	stack ... more than 200h8

```
BDOS-call:      execute BDOC-call (for disk access)
-------------------------------------------------------------------
        <bios 1.1>      di
                        call    0084h
                        (IRQ disabled when returned)

        <bios 2.0>      ei
                        call    0084h
                        (IRQ enabled when returned)

        <interface>     <in>            <return>
                [a]     to basic        status
                <c>      -              <0> normal  <1> invalid
                [bc]    to basic        from basic / sub status
                [de]    to basic        from basic
                [hl]    to basic        from basic

        . MsIOS uses [H.HEYI], and restores it after operation.

        <invalid condition>
                . when m-bios is already performing BDOS-call
                . disk does not exist.
                . when m-bios is performing SV-call or disv.

        <status>
                <feh>   disk physical error (see sub-status)
                <other> bdos's status

        <sub-status>
                <01h>    write protected
                <02/03h> drive not ready (R/W)
                <04/05h> CRC error (R/W)
                <06/07h> seek error (R/W)
                <08/09h> record not found (R/W)
                <0a/0bh> write fault (R/W)
                <0c/0dh> other error (R/W)
-------------------------------------------------------------------
```

R-call:

<bios 1.0/1.1>	call 0093h (IRQ disabled when returned)		
<bios 2.0>	ei call 0093h (IRQ enabled when returned)		
<interface>	<in>	<return>	
[a]	op-code#	status#	
<c>	-	<0> normal	<1> error
[bc]	parameter	*	
[de]	parameter	*	
[hl]	-	*	
[ix/iy]	-	o	

<-call:

<bios 1.0/1.1>	di/ei call 0096h (IRQ disabled/enabled when returned)		
<bios 2.0>	ei call 0096h (IRQ enabled when returned)		
<interface>	<in>	<return>	
[a]	op-code#	00h	
<c>	-	<0> normal	<1> busy
[bc]	parameter	*	
[de]	parameter	*	
[hl]	-	*	
[ix/iy]	-	o	

>-call:

<bios 1.0/1.1>	di/ei call 0099h (IRQ disabled/enabled when returned)		
<bios 2.0>	ei call 0099h (IRQ enabled when returned)		
<interface>	<in>	<return>	
[a]	00h	00h	
<c>	-	<0> normal	<1> busy
[bc]	-	*	
[de]	queue map	*	
[hl]	-	*	
[ix/iy]	-	o	

S-call:

```
        <bios 1.0/1.1>  call    0090h
                        (IRQ disabled when returned)

        <bios 2.0>      ei
                        call    0090h
                        (IRQ enabled when returned)

        <interface>     <in>            <return>
                [a]     op-code#        status#
                <c>      -              <0> normal  <1> error
                [bc]    parameter          *
                [de]    parameter          *
                [hl]     -                 *
                [ix/iy]  -                 o

        . Contents of alternative registers are destroyed
         in S-20,S-21,S-22,S-23,S-24 calls.
```

M-call: scan MIDI/in or FIFO/re

```
        <bios 1.0/1.1>  di
                        call    00A5h
                        (IRQ disabled when returned)

        <bios 2.0>      ei
                        call    00A5h
                        (IRQ enabled when returned)

        <interface>     <in>            <return>
                [a]      -                 *
                <c>      -                 *
                [bc]     -                 *
                [de]     -              data/status
                [hl]     -                 *
                [ix/iy]  -                 o

        <status>        ---- ----       <1> framing error
                        ---* ----       <1> overrun,
                                            FIFO/re is overflow
                        ---- ---*       <1> data exist
```

F-call:

<bios 2.0>	ei call 0091h (IRQ enabled when returned)	
<interface>	<in>	<return>
[a]	op-code#	*
<c>	-	*
[bc]	parameter	*
[de]	parameter	*
[hl]	-	*
[ix/iy]	-	o

IRQC:

<bios 1.0/1.1>	di jp/call 009Fh	
<interface>	<in>	
[a]	-	
<c>	-	
[bc]	-	
[de]	-	
[hl]	-	
[ix/iy]	-	

[7]. several SV-calls

R-20 call: send MIDI message

	<in>	<return>
[a]	20h	00h
< c>	-	<0>
[bc]	buffer size	*
[de]	buffer address	*
[hl]	-	*

. You must send one MIDI message by one R-20 call.
. If FIFO/Tr is full, it waits.

R-21 call: send MIDI real time message

	<in>	<return>
[a]	20h	00h
< c>	-	<0>
[bc]	-/data	*
[de]	-	*
[hl]	-	*

<data> 1111 1--- F8h - FFh

. If FIFO/Re full, it waits.

S-05 call: define FIFO/Re, FIFO/Tr

	<in>	<return>
[a]	05h	00h
<c>	-	busy
[bc]	FIFO/Tr address	*
[de]	FIFO/Re address	*
[hl]	-	*

. This SV-call includes the function of S-0C call, and in operating mode 1.1 this SV-call is same as S-0c call.
. The size of FIFO/Re or FIFO/Tr is 100h bytes.
. FIFO/Re or FIFO/Tr is canceled when address value is 0000h.
. When FIFO/Re is defined, RxRDY IRQ is enabled.
. When FIFO/Re is not defined, RxRDY IRQ is disabled.
. When FIFO/Tr is not defined, TxRDY IRQ is disabled.

S-0D call: enable/disable opm IRQ

	<in>	<return>
[a]	0Dh	00h
<c>	-	busy
[bc]	-/mode	*
[de]	-	*
[hl]	-	*

<mode>	0000 00-*	<1> enable clock-A
	0000 00*-	<1> enable clock-B

S-0E call: select IRQ/priority

	<in>	<return>
[a]	0Eh	00h
<c>	-	busy
[bc]	-/mode	*
[de]	-	*
[hl]	-	*

<mode> 0000 000* <0> [MIDI] ---> [a] ---> [b] ---> [vdp]
<1> [MIDI] ---> [c] ---> [a] ---> [vdp]

S-17 call: set stereo l/r

	<in>	<return>
[a]	17h	00h
<c>	-	busy
[bc]	IDB#/data	*
[de]	-	*
[hl]	-	*

<data> *-00 0000 <1> enable /l
-*00 0000 <1> /r

S-18 call: set PMS, AMS

	<in>	<return>
[a]	18h	00h
<c>	-	busy
[bc]	IDB#/data	*
[de]	-	*
[hl]	-	*

<data> 0*** 00-- PMS
0--- 00** AMS

S-19 call: get & load voice (performs S-14 after S-15)

	<in>	<return>
[a]	19h	00h
<c>	-	busy
[bc]	IDB#/voice#	*
[de]	-	*
[hl]	-	*

S-20 call: file driver (basic's format id#=feh)

	<in>	<return>
[a]	20h	error position
<c>	-	error
[oc]	-	-/error status
[de]	buffer address	*
[hl]	-	*

. Contents of alternate registers are destroyed.

<error location>
- <01> pre-procedure
- <02> open file
- <03> read basic's header file
- <04> read file body
- <05> close file
- <06> create file
- <07> write basic header
- <08> write file body

<error status>
- <01> disk i/o error (write protected)
- <02> disk i/o error (drive not ready)
- <03> disk i/o error (crc error)
- <04> disk i/o error (seek error)
- <05> disk i/o error (record not found)
- <06> disk i/o error (write fault)
- <07> i/o error
  - (undefined disk error)
  - (cmt i/o error)
  - (s/ram i/o error)
- <08> bdos error
- <09> invalid parameter error
  - (invalid access code#)
  - (invalid device#)
- <0a> invalid file type error
  - (invalid basic's id#)
  - (invalid yamaha's id)
- <0b> file body size is larger than buffer
- <0c> busy, invalid condition

<buffer / 17 bytes>
- [0] access code#
  - <00h> read file
  - <01h> write file
  - <02h> read UVL
  - <03h> write UVL
- [1] device# <00h> cmt
  - <10h> s/ram
  - <20h> default drive
  - <21h> a:
  - <22h> b:
- [2] buffer address (for access-code#=0)
- [4] buffer size (for access-code#=0)
- [6] file name
  - cmt: 6 chr. (for access-code#=0)
    "VOICE " (fixed for UVL)
  - s/ram: not used
  - disk: 8 chr.
- [14] extention name

```
cmt:    not used
s/ram:  not used
disk:   3 car.
        "V03" (default<J0n> for UVL)
```

F-00 call: clear FIFO/re

	<in>	<return>
[a]	00h	*
<c>	-	*
[bc]	-	*
[de]	-	*
[hl]	-	*

. Clear FIFO/Re, and reset error flag

F-01 call: clear FIFO/tr

	<in>	<return>
[a]	01h	*
<c>	-	*
[bc]	-	*
[de]	-	*
[hl]	-	*

F-02 call: reset error flag

	<in>	<return>
[a]	02h	*
<c>	-	*
[bc]	-	*
[de]	-	*
[hl]	-	*

F-03 call: send MIDI real time message

	<in>	<return>
[a]	03h	*
<c>	-	*
[bc]	-/data	*
[de]	-	*
[hl]	-	*
[ix/iy]	-	*

<data> 1111 1--- F8h - FFh

. If FIFO/Tr is full, it waits.

[8]. Difference between SFG-01 and SFG-05

The following differences are the result of "bug-fixed" made on the MbIOS of SFG-05.

Load Voice:	Clears related IDB's voice area (64 bytes) at loading time of SVL's voice, automatically.
SVL data:	Resets "load enable bit" if ... (PMS*PMD)+(AMS*AMD) = 0 Sets 0 into PMS if PMD=0. Sets 0 into AMS if PMD=0.
Portamento:	Enables portamento even if IDB# is not same as this IDB's channel#.
Assign Channel:	Loads voice into IDB#CSM at the assigning of channels.
Noise:	Bug-fix for noise function.
LFO sync:	Bug-fix for LFO sync function.
MIDI(out):	Enables MIDI(out) for the IDb which has no OPM channels.
Recording:	Enable recording start even if MSB of recording queue# byte (in R-08) is reset.
Recording:	Missing of last event of recorded data is now fixed.
<K-00> call:	Disables IRQ in <K-00> call.
MK scan:	Disables IRQ while scanning MK.
BASIC's HOOK:	Restores HOOK (H.KEYI, H.ERRO) after loading or saving into CMT.
I-call:	Sets OPM output level into -95.25db.
AST:	(caution) You must not issue any SV-call (except for F-call) in AST. When queue has overflowed by BIOS's "set event (R-02)", bios issues "all note off (R-01)" for related queue automatically.

The following differences are the result of "refine of MBIOS specification" of SFG-05.

slot: Enables the BIOS function, even if slot#0 is expanded.

MIDI(out): All-Note-Off message is now changed to work better with DX series.

	(SFG-01)	BXh/7Eh/00h	(mono/on)
		BXh/7Fh/00h	(poly/on)
	(SFG-05)	BXh/40h/00h	(sustain/off)
		BXh/7Bh/00h	(all note/off)

all_note_off	(SFG-01)	<R-00>	damp
		<R-01>	damp
		<S-03>	damp
	(SFG-05)	<R-00>	damp
		<R-01>	release (normal key/off)
		<S-06>	release (normal key/off)
