IBM

PC Music Feature
Technical Reference

First Edition (March 1987)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

Preface

The IBM PC Music Feature Technical Reference describes the hardware design and provides interface information for the IBM PC Music Feature. It also contains information about the basic input/output system and programming support.

The information in this manual is both descriptive and reference-oriented. It is intended for hardware and software designers, programmers, engineers, and other people who need to understand the design and operation of the IBM PC Music Feature. These users should be familiar with the following:

- The IBM system in which the music card is installed

- MIDI data streams and messages

- The generation of synthesized voices

- The concepts of computer architecture and programming.

Suggested Reading

Musical Instrument Digital Interface Specification 1.0, International MIDI Association, 11857 Hartsook Street, North Hollywood, California 91607.

Contents

Chapter 1. Product and Functional Description

This chapter is an overview of the IBM PC Music Feature. It describes music card functions and includes illustrations of the data flow.

Product Description

The IBM PC Music Feature is an eight-voice polyphonic tone generator card for use with the IBM Personal Computer family. The music card employs the FM tone generation method and has 240 preset voices. It stores an additional 96 user-programmed voices. You can freely set the individual voice characteristics and number of voices to be played, thereby selecting a variety of performance configurations.

The music card provides Musical Instrument Digital Interface (MIDI) that allows you to connect devices that comply with MIDI Specification 1.0. MIDI is a standard for transferring control signals between different musical instruments or between musical instruments and a computer.

The following figure is a block diagram of the data flow on the music card adapter.

Figure 1-1. Block Diagram of the Music Card

Product Technical Specifications

Power requirements and other technical information for the music card is described in Appendix B, "Technical Specifications."

Functional Description

This section describes music card modes and functions.

The music card has two primary modes:

- THRU mode

- MUSIC mode.

Following is a description of these modes.

THRU Mode

In THRU mode, the music card operates as a MIDI interface with a first-in, first-out (FIFO) buffer. The musical functions of the card cannot be used in this mode. In THRU mode, all 8-bit binary numbers can be sent and received. The music card sends the data from the MIDI IN terminal to the system. It sends data from the system to the MIDI OUT terminal. The music card does not interpret data. In THRU mode, the system uses the music card as a serial communication device, which can handle all binary numbers from [00] through [FF].
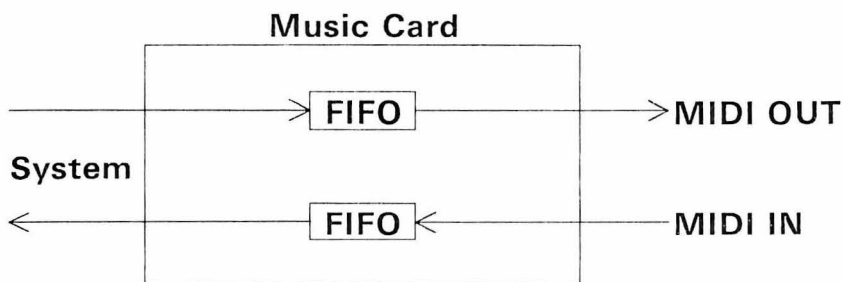
Music Card

Figure   1-2.   Data Flow in THRU Mode

MUSIC Mode

In MUSIC mode, the music card operates as a
sound generator card. MUSIC mode is used
when controlling the card from the system or
from an external MIDI device.
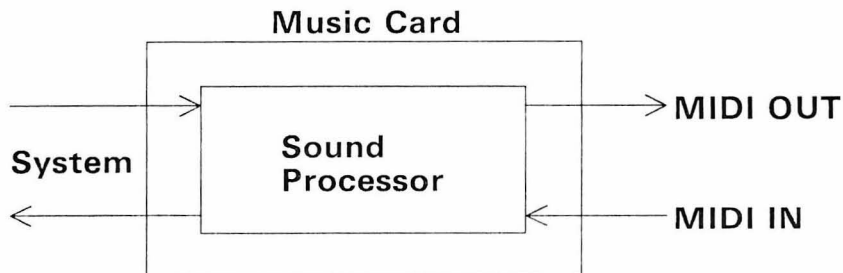
Figure   1-3.   Data Flow in MUSIC Mode

Sound Processor

The sound processor provides the musical
functions.

The internal microprocessor controls the sound
generator and parameter blocks based on the
data sent from the system or MIDI device.

Following is a block diagram of the sound processor.
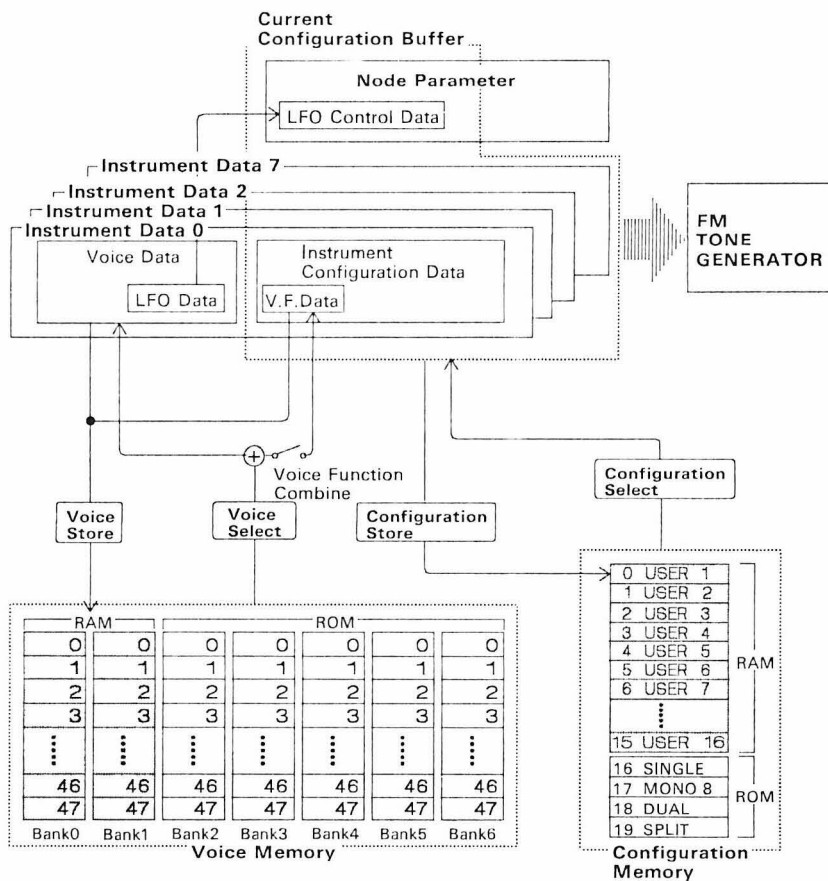
Figure 1-4. Block Diagram of the Sound Processor

Instruments

The music card can play eight voices concurrently. Each voice is called an *instrument*. The parameters for an instrument are defined by instrument voice data and instrument configuration data.

```
┌─Instrument Data 7 ─────────────────────
 ┌─Instrument Data 2 ───────────────────
 ┌─Instrument Data 1 ─────────────────
┌─Instrument Data 0 ───────────────
  ┌─────────────────┐  ┌──────────────────┐
  │ Voice Data      │  │ Configuration Data│
  │ (piano, trumpet,│  │ (volume,          │
  │ clarinet...)    │  │ portamento...)    │
  └─────────────────┘  └──────────────────┘
```
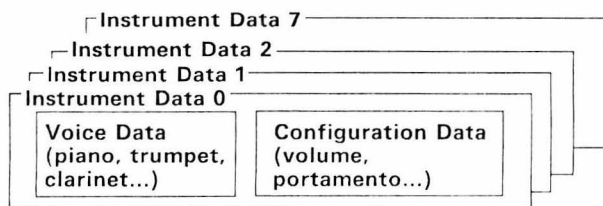
Figure 1-5. Instrument Definition

A MIDI channel can be assigned independently to each instrument, enabling a maximum eight-part performance using a single music card. You can set the number of notes for each instrument, but the maximum number is eight notes per card.

COMBINE Mode

If you select a new voice with COMBINE mode on, both voice data and voice function parameters within the configuration data are updated. When COMBINE mode is off, only the voice data is updated.

Voice Memory

The music card can have 336 types of voice data. Voice data is allocated to banks 0 through 6. Each bank has 48 voices. You can program the voices in banks 0 and 1. Voices in banks 2 through 6 contain preset voice data. See Figure A-5 on page A-6. When you select a voice, the voice data is loaded into the instrument data area.

Configuration Memory

Configuration data consists of instrument parameters and node parameters.

The node parameters apply to the entire music card. These parameters are configuration name, COMBINE mode, LFO speed, AMD, PMD, LFO wave form, and note number reception mode. The instrument parameters control certain aspects of the voice generation and apply only to the selected instrument. The configuration memory stores up to 20 configurations. You can modify the configuration of the music card by selecting a new configuration number.

Configuration Random Access Memory (RAM)

This memory stores the configuration data. Up to 16 configurations can be allocated and stored in configurations 0 through 15. See Figure A-1 on page A-2.

Configuration Read-Only Memory (ROM)

ROM stores configurations 16 through 19. These configurations are preset and cannot be modified. See Figure A-6 on page A-7.

Low-Frequency Oscillator (LFO) Control Block

A low-frequency oscillator (LFO) performs *vibrato* (periodic fluctuations in pitch) and *tremolo* (periodic fluctuations in amplitude) effects. Voice data parameters control the LFO. When you select a voice, the LFO information is loaded into the node parameter area of the current configuration. The LFO control block contains the LFO information from the most recently selected voice, unless LFO load mode of voice data is off. The amount of low-frequency oscillation (depth of vibrato or tremolo) is optimally set for each voice, but can be changed under software control.
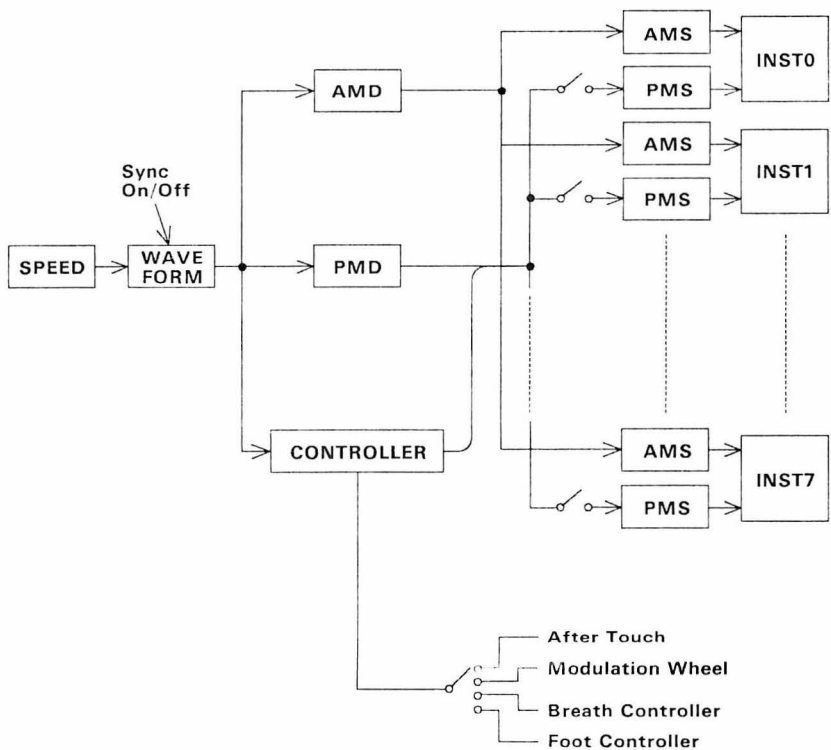
Figure 1-6. Relationship Between LFO Control Block and Instruments

Chapter 2. Input/Output (I/O) Specifications

This chapter describes the interface between the system and the music card.

I/O Register Map

Up to two music cards can be used at the same time. Each card must have a unique address. The switch block (SW1) is on the music card so that the I/O addresses of the music card can be changed. Each music card requires 16 bytes of contiguous I/O addresses, starting with the base address. The base address of the card is assigned as follows:

Card	Position 1	Position 2	Base Address
1	OFF	ON	[2A20]
2	ON	ON	[2A30]

Note: This manual uses decimal, binary and hexadecimal methods. Numerals within brackets [nn] indicate hexadecimal values. Numerals within caret marks < bbbbbbbb > indicate binary values.

When the base address is determined, 16 consecutive addresses, beginning with that base address, are allotted to the music card, as follows.

Offset	Device	Abbrev.
+ [0]	Parallel interface unit port 0	PIU0
+ [1]	Parallel interface unit port 1	PIU1
+ [2]	Parallel interface unit port 2	PIU2
+ [3]	Parallel interface unit command register	PCR
+ [4]	Counter register 0 (timer A)	CNTR0
+ [5]	Counter register 1 (timer B)	CNTR1
+ [6]	Counter register 2 (prescaler to timer B)	CNTR2
+ [7]	Timer control word register	TCWR
+ [8]	Total control register	TCR
+ [9]~[B]	Not used	(Note 1)
+ [C]	Total status register	TSR
+ [D]~[F]	Not used	(Note 2)

Notes:

1. + [9]~[B]: Image of TCR appears.

2. + [D]~[F]: Image of TSR appears.

Parallel Interface Unit (PIU)

The parallel interface unit (PIU) transfers data between the music card and the system. To allow the system to control the card, initialize the PIU by writing [BC] to the PCR.

Port 0 (PIU0)

Data is transferred from the music card to the
system by reading from Port 0. Port 0 (PIU0) is
used in conjunction with bit 7 of PIU2 to transfer
the 9-bit data word from the music card to the
system. The ninth bit (bit 8) is read from PIU2
prior to reading PIU0, as it may be overwritten
when RxRDY is cleared. RxRDY is cleared by
reading PIU0.

Base Address + [0]

Bit	Significance
7	Data bit 7
6	Data bit 6
5	Data bit 5
4	Data bit 4
3	Data bit 3
2	Data bit 2
1	Data bit 1
0	Data bit 0

Sample Program

Following is a sample program for reading data
from the music card.

```
k$read:   cli
          in    al,PIU2
          test  al,00001000b   ;RxRDY?
          jne   r$yes          ;Yes...
          mov   ax,-1           ;No....Not ready
          jmp   r$rtn
;
r$yes:    xor   ah,ah
          sal   ax,1           ;LSB of ah = bit 8 of
                                 9-bit data
          in    al,PIU0        ;ax = receive data from
                                 music card
;
r$rtn:    sti
          ret
```

In this example, music card data is returned in
the AX register. If there is no music card data,
the AX register is -1.

Port 1 (PIU1)

Data is transferred from the system to the music
card by writing to Port 1 (PIU1). Port 1 is used
in conjunction with bit 4 of the TCR to transfer
the 9-bit word from the system to the music card.
When data is written to PIU1, TxRDY is cleared.

Base Address + [1]

Bit	Significance
7	Data bit 7
6	Data bit 6
5	Data bit 5
4	Data bit 4
3	Data bit 3
2	Data bit 2
1	Data bit 1
0	Data bit 0

Sample Program

Following is a sample program for writing data
to the music card from the BX register:

```
k$write:    cli
            in    al,PIU2
            test  al,00000001b   ;TxRDY?
            jne   w$yes          ;Yes...
            mov   ax,-1           ;No...
                                 Music card is busy
            jmp   w$rtn
;
w$yes:      xor   al,al          ;assume bit 8 = 0
            test  bh,1           ;bit 8 = 0?
            jz    w$write        ;Yes...
                                 Good assumption
            mov   al,00010000b   ;else bit 8 = 1
                                 (IRQ is masked)
;
w$write:    out   TCR,al
            mov   al,bl
            out   PIU1,al
;
w$rtn:      sti
            ret
```

Port 2 (PIU2)

Port 2 (PIU2) is a read and write register used to transfer information as described below.

System Write

Base Address + [2]

Bit	Significance
7~5	Not used
4	Read interrupt enable (RIE)
3	Not used
2	Write interrupt enable (WIE)
1~0	Not used

Bit 2 — When set to 1, the TxRDY interrupt is enabled.

Bit 4 — When set to 1, the RxRDY interrupt is enabled.

Bits 2 and 4 of PIU2 are interrupt-enable flags for interrupts to the system from the music card. To enable an interrupt, set the corresponding bit to 1. To inhibit an interrupt, set the corresponding bit to 0. Set bits 2 and 4 indirectly through the Bit Set/Reset command of the PIU Control Register (PCR). You cannot set a bit directly in this register.

System Read

Base Address + [2]

Bit	Significance	
7	Receive data bit 8 (EXR8)	[data]
6	Not used	
5	Not used	
4	Read IRQ enable (RIE)	[status]
3	Receiver ready (RxRDY)	[status]
2	Write IRQ enable (WIE)	[status]
1	Not used	
0	Transmitter ready (TxRDY)	[status]

Bit 0 — The power-on default value is 0. When WIE is 1 and PIU1 assumes Ready-For-Data status, bit 0 is set to 1. When set to 0, data cannot be written to PIU1. When the system writes data to PIU1, bit 0 is set to 0.

Bit 2 — This bit indicates the WIE status. When set to 1, the WIE is in Enable status. When set to 0, the WIE is in Disable status.

Bit 3 — The power-on default value is 0. When RIE is 1, this bit indicates that PIU0 has the data that should be read by the system. When the system reads PIU0, it sets this bit to 0.

Bit 4 — This bit indicates the RIE status. When set to 1, RIE is in Enable status. When set to 0, RIE is in Disable status.

Bit 7 — The bit 8 communication data from the music card is sent to this bit. When set to 1, data from the music card is a special message. When set to 0, data from the music card is the usual 8-bit data.

PIU Command Register

This register is used to initialize the PIU and for Bit Set/Reset operations of PIU2.

Base Address + [3]

Bit	Significance
7~0	Command data

[BC] – Initializes the PIU.

[05] – Sets bit 2 (Write Interrupt Enable) of PIU2.

[04] – Resets bit 2 (~~Read~~ Write Interrupt Enable) of PIU2.

[09] – Sets bit 4 (Read Interrupt Enable) of PIU2.

[08] – Resets bit 4 (Read Interrupt Enable) of PIU2.

Programmable Interval Timer (PIT)

The music card has a Programmable Interval Timer (PIT) that is software-compatible with the 8253 timer. Counter 0 is used as timer A and counters 1 and 2 are used together as timer B. These two timers can be used by application programs.

Counter 0 (CNTR0)

This counter is connected to:

- Bit 0 of the Total Status Register

- The interrupt level specified by the jumper pin on the music card.

Use this counter in the 8253 mode 2. Its countdown resolution is in two-microsecond (μ sec.) units. The setting range is from 2 (4 μ sec.) through 65,535 (131.07 μ sec.).

Base Address + [4]

Bit	Significance
7~0	Count data for timer A

Counter 1 (CNTR1)

This counter is connected to:

- Bit 1 of the Total Status Register

- The interrupt level specified by the jumper pin on the music card.

Use this counter in mode 2. Counter 2 acts as the prescaler for counter 1. The countdown resolution is the product of:

Counter 1 x counter 2 x 0.5 μ sec.

The setting range is from 2 through 65,535.

Base Address + [5]

Bit	Significance
7~0	count data for timer B

Counter 2 (CNTR2)

Counter 2 drives counter 1 and should be used in mode 3. The setting range of counter 2 is from 2 through 65,535.

Base Address + [6]

Bit	Significance
7~0	Prescaler data for timer B

Timer Control Word Register (TCWR)

This register specifies the operating mode of the counters. It is used for counter mode specification and for latch operations of the current count data.

Base Address + [7]

Bit	Significance
7~0	Command

The music card uses the following control words.

Mode Set Command

Before using the timer, issue the following
commands to set the mode of each counter.

[34] — Sets counter 0 for mode 2 and the 16-bit
binary counter.

[74] — Sets counter 1 for mode 2 and the 16-bit
binary counter.

[B6] — Sets counter 2 for mode 3 and the 16-bit
binary counter.

Latch Command

This command reads the current counter data.
When the following commands are issued,
contents of the selected counter are latched to
the counter register.

[00] — Latches the contents of counter 0 to the
register.

[40] — Latches the contents of counter 1 to the
register.

[80] — Latches the contents of counter 2 to the
register.

If the 16-bit read/write mode is specified, the data
is sequentially read from the low-order bits to
high-order bits. Disable interrupts when reading
counter contents.

Total Control Register (TCR)

TCR performs interrupt request (IRQ) control.
Bit 4 is the expansion bit when sending data
from the system to the music card.

Base Address + [8]

Bit	Significance	Function
7	IRQ enable (IBE)	[control]
6	Total IRQ mask (-TMSK)	[control]
5	Not used	
4	Transmit data bit 8 (EXT8)	[data]
3	Timer B enable (TBE)	[control]
2	Timer A enable (TAE)	[control]
1	Timer B clear (-TBC)	[control]
0	Timer A clear (-TAC)	[control]

Bit 0 — The power-on default value is 0. This bit
clears interrupts from timer A. It is
usually set to 1. When clearing an
interrupt from timer A, this bit is set to
0. It is immediately reset to 1.

Bit 1 — The power-on default value is 0. This bit
clears interrupts from timer B. It is
usually set to 1. When clearing an
interrupt from timer B, this bit is set to
0. It is immediately reset to 1.

Bit 2 — The power-on default value is 0. This bit
is the interrupt-enable flag for timer A.
To enable an interrupt, this bit is set to
1. To inhibit an interrupt, it is set to 0.

Bit 3 — The power-on default value is 0. This bit
is the interrupt-enable flag for timer B.
To enable an interrupt, this bit is set to
1. To inhibit an interrupt, it is set to 0.

Bit 4 – This bit is used as bit 8 when sending data to the music card. Data is sent in conjunction with PIU1.

Bit 6 – The power-on default value is 0. When this bit is set to 0, all interrupt requests generated by the music card can be masked. When set to 1, interrupt requests will not be masked.

To process overlapping interrupt requests, the mask function is used as follows for an interrupt service routine:

1. Write 0 to TMSK to suspend (mask) all possible outstanding IRQ sources.

2. Poll one or more IRQ sources and complete the required interrupt handling.

3. Reset the IRQ bits corresponding to the sources serviced in step 2.

4. Issue the EOI command to the system's 8259 (interrupt controller) to re-arm its negative edge sensitivity latch.

5. Set TMSK to 1 to unmask for future IRQ or possible IRQ that occurred during steps 1, 2, and 3.

    Make sure that the systems 8259 (interrupt controller) gets re-armed again for another positive-going edge of IRQ without being disturbed (by possible outstanding un-reset IRQ) during the EOI process.

Bit 7 — The power-on default value is 0. When this bit is set to 1, the IRQ lines of the music card are connected to the IRQ lines of the system bus. When set to 0, this bit is floated from the system bus by a three-state buffer. Set this bit prior to setting TMSK.

Total Status Register (TSR)

TSR reports whether an IRQ is being generated anywhere on the music card.

Base Address + [C]

Bit	Significance
7	Total card IRQ (TCS)
6~2	Not used
1	Timer B status (TBS)
0	Timer A status (TAS)

Bit 0 — Reports the IRQ status of timer A. When set to 1, timer A is generating an IRQ. When the IRQ of timer A is cleared, this bit is also cleared.

Bit 1 — Reports the IRQ status of timer B. When set to 1, timer B is generating an IRQ. When the IRQ of timer B is cleared, this bit is also cleared.

Bit 7 — Reports the IRQ status of the entire music card. When set to 1, one of the IRQ sources on the music card (timer A, timer B, TxRDY, or RxRDY) is generating an IRQ. When none of the sources is generating an IRQ, this bit is 0.

Interrupt Information

The music card contains four IRQ sources:

- Timer A

- Timer B

- TxRDY

- RxRDY.

These IRQs are supplied to the system over a
single interrupt line. The interrupt level can be
specified by connecting the jumper pin on the
music card to one of levels 2 through 7. To send
these IRQs to the system, IRQ Bus Enable and
Total IRQ Mask are set in advance to 1. The
enable flag of each source is then set to 1. This
is illustrated schematically in the following
figure.
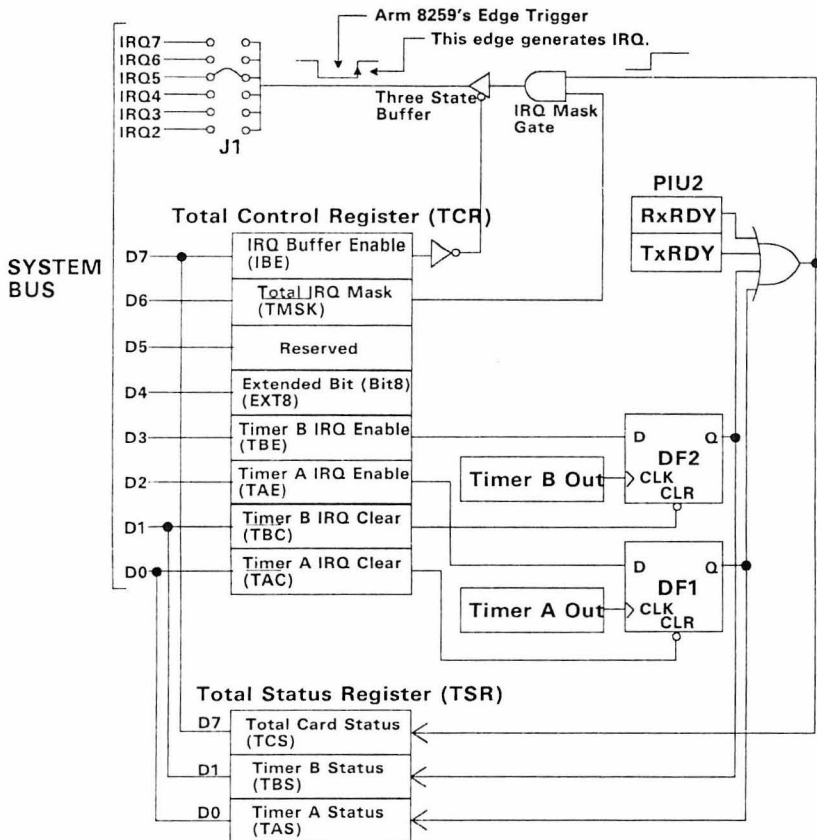
Figure 2-1. Music Card Interrupt System

Chapter 3. Communication Protocol Between the System and Music Card

This chapter describes the communication protocol, including data format, messages, and communication in MUSIC and THRU modes.

Communication between the system and the music card is performed using 9-bit data. To send data from the system to the music card, port 1 (8 bits) of the Parallel Interface Unit (PIU) and bit 4 of the Total Control Register (TCR) are used. When reading data from the music card to the system, port 0 (8 bits) of the PIU and bit 7 of port 2 are used. Bit 4 of TCR and bit 7 of port 2 correspond to bit 8 of the 9-bit data.

When set to 1, bit 8 indicates that the data is a music card message. When set to 0, it indicates MIDI (MUSIC mode) or non-MIDI (THRU mode) data.
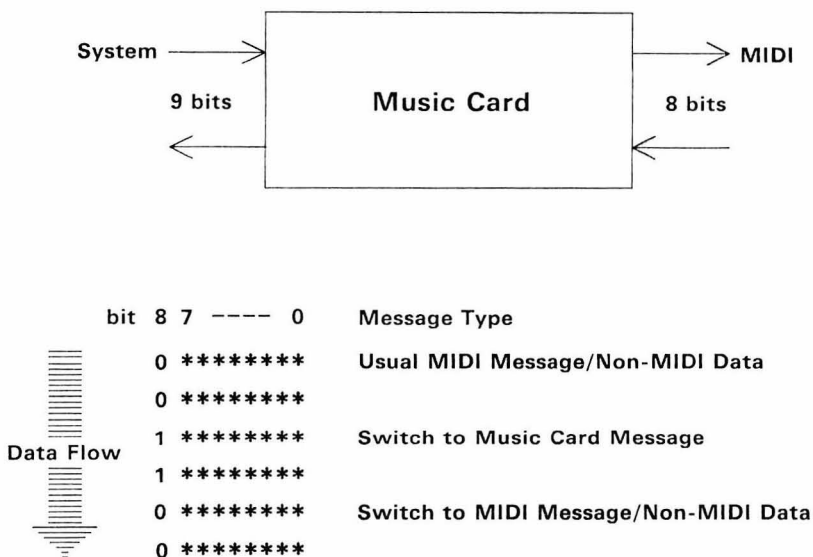
Figure   3-1.   System to MIDI Device Data Flow

A status byte at the beginning of the data is used
for both music card and MIDI messages to
indicate the contents of the message. The status
byte (bit $7 = 1$) defines the amount of data that
follows, or the message meaning. Some messages
consist of only a single status byte.

The status byte may be followed by data bytes.
Bit 7 of the data byte is set to 0 to distinguish it
from the status byte.

When the music card detects a status byte that is
not an error message or a MIDI real-time
message, the music card discards the previous
message and begins interpretation of the new
message. The system should follow the same
procedure.

```
bit  8 7  ----  0        Message Type
        * 1 *******       Status (Top of Message)
        * 0 *******       Data
Data Flow  * 0 *******    Data
        * 1 *******       Status
        * 1 *******       Status
        * 0 *******       Data
```
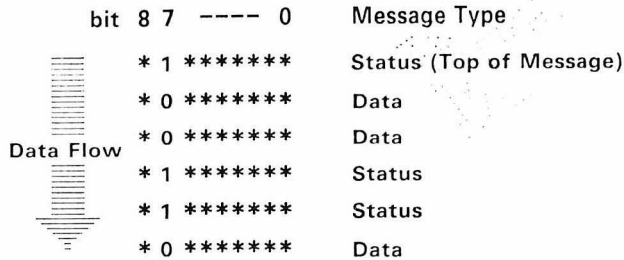
Figure 3-2. Data Flow with Status Byte Control

Music Card Messages

A music card message consists of 9 bits of data
(bits 0 through 8) where bit 8 is 1. The music
card messages can be grouped into three
categories:

- Error report messages

- Command messages

- Status request messages.

Error Messages

This message reports an error detected by the
music card. The error report status must be
enabled for the music card to report an error to
the system. This error report can be freely
enabled or disabled.

An error report message consists of a single status byte. The data format is:

Music Card→System

```
<1 1111ssss> [1fS]        Status byte (command)
```

Command Messages

This message is sent from the system to the music card to set the parameters of the music card. *A command message* consists of a status byte that indicates the command type, followed by data bytes that indicate the contents of the instruction. The length of the data varies with the command type. The data format is:

System→Music Card

```
<1 1110ssss> [1Es]        Status byte (command)
<1 0aaaaaaa>              Data bytes to follow
```

When the music card finishes processing the command sent from the system, it acknowledges the status byte of the processed command message by returning it to the system. For example, when the [1E0][100] command (which sets the music card to MUSIC mode) is sent to the music card, the music card processes the command and sends back the code [1E0] as acknowledgement.

Status Request Messages

This message requests the music card to report its current status to the system. *A status request message* consists only of a status byte. The data format is:

System→Music Card

```
<1 1101ssss> [1Ds]        Status byte (report)
```

The format of the data the music card sends back is identical to the command message for setting that mode, except for the status byte.

Music Card→System

```
<1 1101ssss>  Status byte (report)
<1 0aaaaaaa>  Data bytes to follow
```

MIDI Messages

MIDI is a standard for transferring control signals between different musical instruments or between musical instruments and a computer. In MUSIC mode, the music card processes 9-bit data from the system, where bit 8 equals 0 for a MIDI message. The music card also processes 8-bit MIDI information received from the MIDI IN terminal of the MIDI adapter.

MIDI messages are classified as follows:

- Channel messages

  - Voice messages

  - Mode messages

- System messages

  - Common messages

  - Real-time messages

  - Exclusive messages

Paths

There are five paths over which MIDI data can
flow, as illustrated in the following figure:
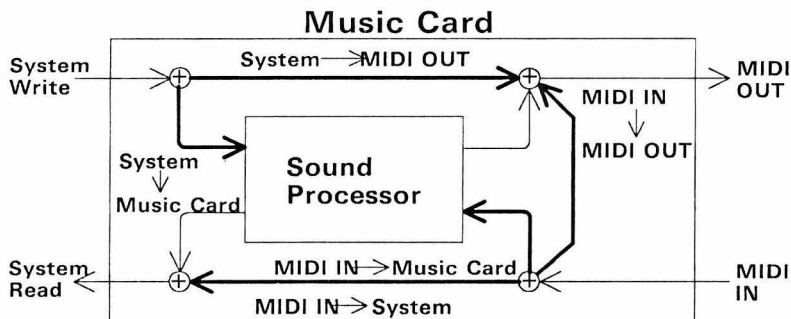
Figure   3-3.   MIDI Data Flow Paths

Each path can be programmed to pass or block
specific MIDI messages. The messages are
categorized in five groups (see "Set Paths" on
page 4-6).

Channel Messages

A *channel message* is sent to a specific musical
channel. *Channel* specifies the message
destination in reference to a MIDI channel. The
receiving channel only processes messages that
are specifically addressed to it. A channel
message consists of a status byte followed by
1 or 2 data bytes.

For channel messages, if the status byte of a
message is identical to the preceding status byte
(including the channel number), the status byte
can be omitted. Data is processed as though it
were sent with the stored status byte. A new
status byte (except for a system real-time
message) replaces the stored status byte. The
music card in MUSIC mode uses this technique
to improve performance.

Channel Voice Messages

The *channel voice message* transfers data directly
related to musical performance. The music card
can receive, but cannot transmit this type of
message. The messages include various control
information, such as ON/OFF of a specific note,
and change of a voice number.

Channel Mode Messages

The *channel mode message* sets the mode that
processes the MIDI data. A channel mode
message includes messages that switch between
polyphonic and monophonic modes. This
message consists of a status byte followed by
2 data bytes.

System Messages

System messages consist of the following:

- Common messages

- Real-time messages

- Exclusive messages.

System Common Messages

The *system common message* sets up the entire system and detects sequence data. A system common message cannot be used with the sound processor.

System Real-Time Messages

In an automatic performance, a *system real-time message* synchronizes the sequencer or rhythm box. A system real-time message includes such messages as Start/Stop performance or Sync Clock. This message consists of a single status byte.

System Exclusive Messages

The *system exclusive message* consists of data related to playing notes, operation control, and so forth. Message contents vary with manufacturers and devices. The length of the message varies, but its format generally is a status byte followed by a data byte describing the instruction contents. Following is a description of five types of system exclusive messages.

Channel Messages: These messages specify the destination of the system exclusive message by the MIDI channel number. This message is similar to the general channel messages of MIDI. The music card functions only when this receiving channel matches a channel assigned to an instrument defined on the music card. The music card changes the parameters related to the voices, instrument functions, voice functions, and so forth.

Instrument Messages: These messages specify their destination by instrument number and node number. It performs instrument-related control, such as changes of voice data, instrument function data, voice function data, and so forth. This message also allows read out of voice data and other data that is set for each instrument.

Node Messages: These messages specify their destination by node number. It changes the system parameters (node parameter changes). This message also allows programming of configuration data or voice data corresponding to 48 voices.

Event List: This message specifies its destination by MIDI channel number. The music card can specify a fine pitch interval in units smaller than half-steps. This is not possible using the Note ON/OFF message. The music card can also specify the note-playing duration using the Note ON message. In these respects, an event list message differs from conventional message systems. Though a system exclusive message switches to the new message system, once the message system has been initiated, data bytes are processed as event lists until another status byte (except a real-time message) is received.

Parameter List: This message specifies ι.. destination by instrument number and node number and allows continuous transmission of parameter change data. After this mode is initiated using a system exclusive message, the system can continuously transmit only the data required to make parameter changes. This eliminates the overhead of transmitting a separate system exclusive message for each parameter change. The data used in this message is called a *parameter list.* Once the mode has been switched, data bytes are processed as parameter lists until a new status byte (except a real-time message) is received.

Undefined Messages

Any status bytes (and successive data bytes) not defined by MIDI ([F4], [F5], [F9], [FD], [FF]) are always blocked.

Active Sensing Code

An active sensing [FE] code received from MIDI IN is intercepted by the music card. If data is not received by the MIDI IN terminal within 300 ms after the music card receives a single [FE], an off-line error message is generated.

The music card automatically generates the active sensing code every 150 ms and transmits it on MIDI OUT.

Processing System Exclusive Messages

This section describes processing system exclusive messages with multiple paths set and with non-terminated messages.

Multiple Paths: When the two paths, System→MIDI and MIDI→System, have been set, special attention is required to send system exclusive messages over both paths at the same time. With the exception of event lists and parameter lists, the music card can interpret only one system exclusive message at a time. When a long system exclusive message, such as a bulk data transfer, is sent on both paths at the same time, the music card dedicates itself to interpreting the data that was sent first. During this period, the other system exclusive message waits in the FIFO without being processed. As a result, the FIFO can assume FULL status and may overflow.

You can send short exclusive messages concurrently over both paths. While the first message is being interpreted, the other is stored in the FIFO. If the FIFO does not overflow, the second message is processed immediately after the first.

Non-terminated Messages: If a [F7] code (message terminator) or other status byte, except a real-time message, is not received within approximately two seconds, a time-out error message is generated to the system.

CHAIN Mode

CHAIN mode allows you to connect multiple music cards to play more than eight notes simultaneously. The additional notes are passed to the second card when the first card is in CHAIN mode. All other messages are received by both cards. To chain the cards together, a MIDI cable connects MIDI OUT of card 1 to MIDI IN of card 2. CHAIN mode of card 1 is set ON and the CHAIN mode for card 2 is set OFF.

Note: The paths, System→MIDI OUT and MIDI IN→MIDI OUT, are automatically closed when CHAIN mode is set. When CHAIN mode is disabled, the paths return to their previous states.

First-In, First-Out (FIFO) Buffer

The music card has a FIFO buffer for the MIDI and system I/O operations. When a FIFO buffer becomes full, the music card responds according to the following descriptions.

System→Music Card

While FIFO is full, bit 0 of port 2 of the Parallel Interface Unit remains 0.

Music Card→System

Unless the system reads data during the 30 ms interval following the FIFO buffer FULL status, error processing is performed.

Music Card→MIDI

When FIFO becomes empty, data is written.

MIDI→Music Card

As soon as the buffer FULL status is achieved, error processing is done for messages received while the buffer is full.

For descriptions of error processing, refer to "Error Report Messages" on page 4-1.

THRU Mode

In THRU mode, the music card is used as a MIDI communication device, and no data is interpreted as MIDI commands. Regardless of the contents, the data sent from the system is directly output to MIDI OUT, and the data received at MIDI IN is directly output to the system.

In THRU mode, switching between MIDI messages and music card messages is as follows:

Music Card→System

If the music card must send a message while non-MIDI data is being sent, it sends the message immediately. The non-MIDI data message is completed after the music card message has been processed. An error report message ([1F0] to [1FF]) is sent at the moment it is detected without awaiting the completion of other messages.

System→Music Card

When a music card message with multiple bytes of data is being sent, non-MIDI data cannot be sent.

MUSIC Mode

In MUSIC mode, the music card functions as a sound generator card. Both MIDI messages and music card messages can have 1 or more bytes of data. When switching between music card and MIDI messages, the current message completes before the new one begins.

Music Card→System

If the music card must send a message while a MIDI message is in progress, it waits for the MIDI message to complete. If the music card must send a real-time or error report message, it sends the message immediately, regardless of the type of message currently being sent. Real-time messages and error report messages (if enabled) can interrupt the flow of MIDI messages or music card messages. The system must complete processing of interrupted messages.

System→Music Card

When data is sent to the music card from the system, message switching must be performed after each message has been completed. Real-time messages can be sent at any time.

Chapter 4. Music Card Messages

A *music card message* is 9-bit data (bits 0 through 8), where bit 8 is ON (1), and is used only between the system and the music card. There are three types of music card messages:

- Error reports from the music card to the system

- Commands to the music card from the system

- Status requests from the system and reports from the music card.

Error Report Messages

An *error report message* is sent immediately to the system when the music card detects an error. An error report message may interrupt a normal data stream. The validity of the data received prior to the error report message is determined by the type of error detected. The system processes the data in accordance with the error type. If a serious error occurs, such as an overflow of the FIFO buffer from the music card to the system, the interrupted message is discarded. The card waits for the next valid status byte and processing resumes. If the error is not directly related to the message being processed, the system should discard the error report message and continue processing the interrupted message.

Following is a list of error codes for the music card.

THRU and MUSIC Mode:

[1F0] FIFO (Music Card→System) overflow error
[1F1] FIFO (MIDI→Music Card) overflow error
[1F2] MIDI reception error

MUSIC Mode Only:

[1F3] MIDI off-line error
[1F4] Time-out error (MIDI→Music Card)
[1F5] Time-out error (System→Music Card)

When an error occurs in MUSIC mode, the music card:

- Clears the FIFO where the error occurred

- Switches OFF any note being sounded

- Switches OFF sustain

- Switches OFF sostenuto

- MIDI OUT suspends active sensing signal

- Waits for the next valid status byte

- Stops waiting for real-time clock.

FIFO Overflow Errors

FIFO Overflow Error (Music Card→System)

Reports an overflow in the FIFO buffer that sends data from the music card to the system.

FIFO Overflow Error (MIDI→Music Card)

Reports an overflow in the MIDI reception buffer (FIFO).

MIDI Errors

MIDI Reception Error

Reports the occurrence of a MIDI reception error. This reception error is either a framing error detected by the Serial Control Unit or an overrun error.

MIDI Off-Line Error

Reports that an off-line error has occurred in MUSIC mode. See "Active Sensing Code" on page 3-10.

Time-Out Errors

Time-Out Error (MIDI→Music Card)

Reports that a time-out error has occurred while
the music card is receiving a system exclusive
message from MIDI IN in MUSIC mode.

Time-Out Error (System→Music Card)

Reports that a time-out error has occurred while
the music card is receiving a system exclusive
message from the system in MUSIC mode.

Command Messages

Following are the five commands issued from the
system to the music card:

[1E0] Select Music Card Mode

[1E1] Select Error Report Mode

[1E2] Set Path

[1E5] Reboot

[1E3] Set Node Parameters

When a path is set to pass a message group, the
path passes the status byte and the event list for
that group.

Select Music Card Mode

This command selects the music card function mode, either THRU mode or MUSIC mode. The default value is MUSIC mode. The format is:

```
<1 11100000>  [1E0]   Status byte
<1 0000000n>  [10n]   n=0: MUSIC mode
                      n=1: THRU mode
```

Select Error Reporting Mode

This command specifies whether the music card should send an error report to the system. The default value is error report disabled. The format is:

```
<1 11100001>  [1E1]   Status byte
<1 0000000n>  [10n]   n=0: Disable error report
                      n=1: Enable error report
```

Set Paths

This command specifies the music card paths.
Each path can be specified independently. The
five bytes of data following the status byte are
assigned to each path as shown below.

```
<1 11100010> [1E2]  Status byte
<1 000edcba> [1##]  Pass-thru pattern
                    (MIDI IN→System)
<1 000edcba> [1##]  Pass-thru pattern
                    (System→MIDI OUT)
<1 000edcba> [1##]  Accept pattern
                    (MIDI IN→SP)
                    SP=sound processor
<1 000edcba> [1##]  Accept pattern
                    (System→SP)
<1 000edcba> [1##]  Pass-thru pattern
                    (MIDI IN→MIDI OUT)
```

There are five groups of MIDI messages. The
path specification can be set for each group by
using the low-order five bits of a data byte. The
bit corresponding to the classifier should be set
to 1 to enable message passing of that group.

Bit	Affected Message	Status Byte	Status Byte (Event List)
a	Note ON/OFF	[80] ~ [9F]	[00] ~ [2F]
b	After Touch and Pitchbend	[A0] ~ [AF], [D0] ~ [DF] [E0] ~ [EF]	[50] ~ [5F] [60] ~ [6F]
c	Control Change and Program Change	[B0] ~ [BF] [C0] ~ [CF]	[30] ~ [3F] [40] ~ [4F]
d	System Exclusive System Common	[F0], [F7] [F1], [F2], [F3], [F6]	[70] ~ [7F]
e	System Real-Time	[F8], [FA] ~ [FC]	

Note: Undefined MIDI status bytes ([F4], [F5], [F9], [FD], [FF]) and the active sensing code ([FE]) not listed in the preceding table are blocked. Succeeding data bytes are also blocked.

Path Definitions

MIDI IN→System: This is the path from MIDI IN to the system. This path is enabled when the data received at MIDI IN is to be processed by the system.

System→MIDI OUT: This is the path from the system to MIDI OUT. This path is used when sending a message from the system to an external MIDI device. If this path is enabled, it is automatically closed when CHAIN mode is set. When CHAIN mode is cancelled, the path returns to its previous state. Even if this path has been closed by entry into CHAIN mode, status reports from the music card will reflect its programmed state.

MIDI IN→Sound Processor: This is the path from MIDI IN to the sound processor. This path is set so that the music card can be controlled by an external MIDI device.

System→Sound Processor: This is the route from the system to the sound processor. This path is set so that the music card can be controlled by the processor.

> **Note:** If you try to set both the MIDI IN→Sound Processor and System→Sound Processor paths to pass system real-time messages, only the MIDI IN→Sound Processor path is set. The other path is ignored with no error report message.

MIDI IN→MIDI OUT: This is the path from MIDI IN to MIDI OUT. When this path is set, the MIDI OUT terminal functions in essentially the same manner as the MIDI THRU terminal. When the System→MIDI OUT path is concurrently set, the data from the system and the data input to MIDI IN can be merged on MIDI OUT. When this path is set, the program request from a MIDI line is not valid, and a MIDI handshaking message cannot be output to a MIDI line.

This path automatically closes when CHAIN mode is set. When CHAIN mode is cancelled, this path returns to its previous state. Even if this path has been closed by entry into CHAIN mode, status reports from the music card reflect its programmed state.

Reboot

This command reboots the music card. When this command is issued, the entire music card excluding the PIU returns to its default status. The PIU mode is not reset. The format is:

<1 11100101> [1E5]   Status byte

At power-on, the music card is set as follows:

Parameter Name	Description	Value
Card mode setting	MUSIC mode	0
Error reporting	No	0
Node setting		
Node #	0	0
Memory protection	Off	0
Configuration #	Single	16
Master tuning	0	0
Master volume level	Maximum	127
CHAIN mode	Disabled	0
Path parameters setting		
MIDI IN→System	All blocked	< 00000000 >
System→MIDI OUT	All blocked	< 00000000 >
MIDI IN→SP	Accept all MIDI messages	< 00011111 >
System→SP	All blocked	< 00000000 >
MIDI IN→MIDI OUT	All blocked	< 00000000 >

Set Node Parameters

This command sets the most fundamental parameters of the music card, beginning with the node number. The node number is used to identify a music card. This allows selection of a particular music card among multiple music cards when sending voice data or changing parameters.

The format is:

```
<1 11100011>  [1E3] Status byte
<1 0000nnnn>  [10n] Node number, n=0~15
<1 0000000n>  [10n] Memory protect
                    n=1: protect
                    n=0: non-protect
                         (power-on default)
<1 000nnnnn>  [10n] Configuration number, n=0~19
<1 0vvvvvvv>  [1##] Master tune (-64 through 63 in
                    7-bit 2's complement notation)
<1 01111111>  [1##] Master output level
                    (0 through 127)
<1 0000000n>  [10n] Chain mode
                    n=1: enable CHAIN mode
                    n=0: disable CHAIN mode
<1 00000000>  [100] Reserved (always 0)
<1 00000000>  [100] Reserved (always 0)
```

Status Request Message

The status request message consists of the following four message types:

- [1D0] Card mode status

- [1D1] Error report mode status

- [1D2] Path setting

- [1D3] Node parameter setting.

Card Mode Status

This message reports whether the music card is in MUSIC mode or in THRU mode. The format is:

```
<1 11010000> [1D0]   Status byte
```

The response format is:

```
<1 11010000> [1D0]
<1 0000000n> [10n]  n=0: MUSIC mode
                    n=1: THRU mode
```

Error Report Status

This message confirms that error messages will be sent to the system. The format is:

```
<1 11010001> [1D1]   Status byte
```

The response format is:

```
<1 11010001> [1D1]
<1 0000000n> [10n]  n=0: disable Error Report
                    n=1: enable Error Report
```

Path Parameter Status

This message checks the path settings within the music card. The format is:

```
<1 11010010> [1D2]   Status byte
```

The response format is:

```
<1 11100010> [1D2]
<1 000edcba> [1##]  Pass-thru pattern
                    (MIDI IN→System)
<1 000edcba> [1##]  Pass-thru pattern
                    (System→MIDI OUT)
<1 000edcba> [1##]  Accept pattern
                    (MIDI IN→SP)
<1 000edcba> [1##]  Accept pattern
                    (System→SP)
<1 000edcba> [1##]  Pass-thru pattern
                    (MIDI IN→MIDI OUT)
```

See "Set Paths" on page 4-6 for pattern bit definitions.

Node Parameter Status

This message reports the status of the
fundamental parameters of the music card,
beginning with the node number. The format is:

```
<1 11010011> [1D3]                    Status byte
```

The response format is:

```
<1 11010011>  [1D3]
<1 0000nnnn>  [10n] Node number, n=0~15
<1 0000000n>  [10n] Memory protect
                    n=1: protect
                    n=0: non-protect
                         (power-on default)
<1 000nnnnn>  [10n] Configuration number, n=0~19
<1 0vvvvvvv>  [1##] Master tune (-64~63 in
                    7-bit 2's complement notation)
<1 01111111>  [1##] Master output level (0~127)
<1 0000000n>  [10n] CHAIN mode
                    n=1: enable CHAIN mode
                    n=0: disable CHAIN mode
<1 00000000>  [100] Reserved (always 0)
<1 00000000>  [100] Reserved (always 0)
```

Chapter 5. MIDI Messages

The Musical Instrument Digital Interface (MIDI)
is a standard developed to exchange information
between different musical instruments or between
musical instruments and a computer. In MUSIC
mode, data that conforms to the MIDI standard
controls the music card. The messages described
in this section, whether sent from the system or
from an external MIDI device, function in an
identical manner. However, you must set the
appropriate paths in advance before you can use
the MIDI commands.

Channel Voice Messages

Channel voice messages transfer data directly
related to musical performance.

Note OFF Message

This message stops the generation of the note
indicated by the note number. The format is:

```
<1000nnnn> [8n]     Status byte
                    n=channel number (0~15)
<0kkkkkkk>          Note number
                    k=0 (C-2)~127 (G8)
<0vvvvvvv>          Key velocity (ignored)
```

Note ON/OFF Message

This message starts the generation of the note indicated by the note number. The output level of each operator varies with the key velocity value at the time of reception. The velocity setting specified by each set of voice data determines the variance. The attack rate also varies for voices that have specified the attack rate velocity setting.

The playing range of the music card is from D#-1 (KC #15) through D7 (KC #110). The music card can accept MIDI key codes outside this range. When Multi = 1 and the operator is used as a carrier (versus modulator), the notes are transposed into the music card playing range. For example, KC #14 is transposed to KC #26, and KC #111 is transposed to KC #99. When the key velocity is 0, the Note OFF function is invoked. The format is:

```
<1001nnnn> [9n]    Status byte
                   n=channel number (0~15)
<0kkkkkkk>         Note number
                   k=0 (C-2)~127 (G8)
<0vvvvvvv>         Key velocity
                   v=1~127:Note ON
                   v=0     :Note OFF
```

Control Change Message

Control change is the data sent in response to operations performed by various external controllers, such as modulation wheels and pedals.

The following codes are received at the music card.

```
<1011nnnn> [Bn]    Status byte
                   n=Channel number (0~15)
<0ccccccc>         Control number
<0vvvvvvv>         Control value
```

Following are the control numbers and corresponding values for the codes received at the music card.

Control Number	Control Name	Control Value
c = 1	Modulation wheel	v = 0~127
c = 2	Breath controller	v = 0~127
c = 4	Foot controller	v = 0~127
c = 5	Portamento time	v = 0~127
c = 7	Volume	v = 0~127
c = 10	Pan control	v = 0 (L) v = 64 (L + R) v = 127 (R)
c = 64	Sustain ON/OFF	v = 0 (OFF), 127 (ON)
c = 65	Portamento ON/OFF	v = 0 (OFF), 127 (ON)
c = 66	Sostenuto ON/OFF	v = 0 (OFF), 127 (ON)

The default values are as follows:

Volume = 127
Sustain ON/OFF = 0 (OFF)
Portamento ON/OFF = 127 (ON)
Sostenuto ON/OFF = 0 (OFF)

Voice Change Message

This message switches to a voice in the music card memory that is defined by a voice number. The music card automatically sets Note OFF, Sustain OFF, and Sostenuto OFF when a voice change is received. The format is:

```
<1100nnnn> [Cn]    Status byte
                   n=Channel number (0~15)
<0vvvvvvv>         v=Voice number (0~47)
```

After-Touch Message

This message is received when TOUCH is specified from the controller used for pitch modulation. The format is:

```
<1101nnnn> [Dn]    Status byte
                   n=Channel number (0~15)
<0vvvvvvv>         Value
```

Pitchbender Message

This message changes the pitch interval by the specified value. The value is determined by a previously given range. The format is:

```
<1110nnnn> [En]    Status byte
                   n=Channel number (0~15)
<0uuuuuuu>         Value (LSB)
<0vvvvvvv>         Value (MSB)
```

Channel Mode Messages

Channel mode messages set the mode used to process MIDI data. They include All Notes OFF, Mono Mode, and Poly Mode messages. The format of Channel Mode messages is:

```
<1011nnnn> [Bn]      Status byte
                     n=Channel number (0-15)
<0ccccccc>
<0vvvvvvv>
```

Notes:

1. When $c = 123$ and $v = 0$, all notes are OFF.

2. When $c = 126$ and $v = 1$, MONO mode is ON. (MONO mode operates only when PLAY mode is polyphonic.)

3. When $c = 127$ and $v = 0$, POLY mode is ON. (POLY mode operates only when PLAY mode is monophonic.)

System Real-Time Messages

System real-time messages are single-byte messages that convey the immediate status of the music card and attached MIDI devices.

Active Sensing

The active sensing code allows the music card to determine if the MIDI interface is operational. The MIDI reception buffer is cleared, and the note being played is switched off if the following error occurs:

- The active sensing code has been received from MIDI IN, and

- Nothing is received within a period of 300 ms.

The format is:

```
<11111110> [FE]        Status byte
```

Real-Time Clock Message

The music card uses the duration count of the MIDI real-time clock (sync clock) when processing the system exclusive message event list. The format is:

```
<11111000> [F8]        Status byte
```

Start Message

When the music card receives a start message (FA), the MIDI real-time clock resets and begins to count. The format is:

```
<11111010> [FA]        Status byte
```

Continue Message

When the music card receives a continue message (FB), the MIDI real-time clock resumes counting. The format is:

<11111011> [FB]   Status byte

Stop Message

When the music card receives a stop message (FC), the MIDI real-time clock stops counting and stops playing notes started by an event list. The format is:

<11111100> [FC]   Status byte

Chapter 6. MIDI System Exclusive Messages

MDI system exclusive messages control music card operation, bulk data transfer, and handshaking.

Music Card Operational Controls

The music card operation is controlled through parameter change requests, event lists, and store requests.

Parameter Change Requests

Parameter change requests update music card voice and configuration data through instrument or node parameter changes.

Instrument Parameter Change

Two data formats, type A and type B, are used to define instrument parameter changes. The parameters can be changed individually or in groups. Group parameter modification is described in the parameter list format. The destination of individual parameters can be addressed by MIDI channel or node and instrument number.

A type A message consists of 1 byte and is used to change a 7-bit parameter. A type B message consists of 2 bytes and is used to change an 8-bit parameter.

Type A: This message modifies 7-bit parameters within the range from [00] through [3F] and is the message type most typically used to modify these parameters. Parameter numbers [00] through [0F] represent instrument configuration data. Parameter numbers [10] through [17] are used to change instrument voice data contents.

There are two kinds of type A messages. One kind of message specifies its destination by MIDI channel. The other specifies its destination by instrument and node number. Note that the voice (LFO) parameters are loaded directly into the LSI control chip without reloading the entire set of voice parameters.

The format for type A data is shown in the table below.

(1) By MIDI channel number

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<0001nnnn>    [1n]    Sub-status
                     n=Channel number (0~15)
<00010101>    [15]    Parameter group
<00pppppp>           *Parameter number
                     p=[00]~[3F]
<0ddddddd>           *Data
<11110111>    [F7]    End-of-command
```

(2) By instrument and node number

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                     s=0~15
<00011iii>           Message number
                     i=Instrument number
<00pppppp>           *Parameter number
                     p=[00]~[3F]
<0ddddddd>           *Data
<11110111>    [F7]    End-of-command
```

* Defined in Figure 6-1 on page 6-8.

Type B: This message changes an 8-bit
parameter range. The data section for this
message contains 2 bytes. The low-order 4 bits
and the high-order 4 bits of the change data are
divided into 2 bytes. The bits are then sent in a
low-to-high sequence. The parameter numbers
range from [40] to [7F] and are derived by adding
[40] to the parameter number in the Instrument
Parameter list. See Figure 6-1 on page 6-8.

This message type is typically used by voice
editing programs to update a single byte of data
and to reduce the traffic on the MIDI channel.

> **Note:** A byte may contain more than one
> parameter.

The voice editing program edits an 8-bit byte in
the system memory and sends that byte directly
to the music card. The music card updates all 64
bytes of voice data to the LSI control chip,
thereby reducing the traffic on the MIDI
channel. This message type also allows the voice
editing program to send an 8-bit byte (versus a
7-bit byte) to the music card.

There are two kinds of type B messages. One
kind specifies its destination by MIDI channel.
The other specifies its destination by instrument
and node number.

Following is the format for type B data:

(1) By MIDI channel number

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<0001nnnn>    [1n]    Sub-status
                      n=Channel number (0~15)
<00010101>    [15]    Parameter group
<01pppppp>            *Parameter number
                      p=[00]~[3F]
<0000dddd>            *Data low
<0000dddd>            *Data high
<11110111>    [F7]    End-of-command
```

(2) By instrument and node number

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00011iii>            Message number
                      i=Instrument number
<01pppppp>            *Parameter number
                      p=[00]~[3F]
<0000dddd>            *Data low
<0000dddd>            *Data high
<11110111>    [F7]    End-of-command
```

* Defined in Figure 6-1 on page 6-8.

Parameter List

Parameter list is one type of system exclusive
message used to change a group of parameters.
When the 4-byte data consisting of [F0], [43],
[75], and [71] is sent, the data required for
successive parameter changes can be
continuously sent. The continuously sent data is
referred to as a *parameter list*. The use of a
parameter list results in a more efficient
transmission of data when a large quantity of
parameters are to be changed. The parameter
list messages are continuously received and
executed until a MIDI status byte (other than a
real-time message) is received. Note that the
instrument and node numbers define the
destination specification. If destination
specification by MIDI channel number is desired,
use the commands described in "Bulk Data
Transfers" on page 6-16.

The format for a parameter list is shown below.

```
<11110000>    [F0]   Status byte
<01000011>    [43]   Mfg ID
<01110101>    [75]   Sub-status
<01110001>    [71]   Message ID
<0ddddddd>           Parameter change
                     data list
<0ddddddd>
<11110111>    [F7]   End-of-command
```

Parameter lists also allow the type A and type B data formats shown below.

Type A

<0iiissss>	[is]	Instrument/Node number i=Instrument number (0~7) s=Node number (0~15)
<00pppppp>		Parameter number p=[00]~[3F]
<0ddddddd>		Data

Type B

<0iiissss>	[is]	Instrument/Node number i=Instrument number (0~7) s=Node number (0~15)
<0ppppppp>		Parameter number p=[40]~[7F]
<0000dddd>		Data low
<0000dddd>		Data high

Parameter Number	Parameter Name	Value
[00]	Number of notes	0~8
[01]	MIDI channel number	0~15
[02]	Note number limit high	0~127
[03]	Note number limit low	0~127
[04]	Voice bank number	0~6
[05]	Voice number	0~47
[06]	Detune	-64~63 (2's complement)
[07]	Octave transpose	0~4 (2 = Center)
[08]	Output level	0~127
[09]	Pan	0 = L, 64 = L + R, 127 = R
[0A]	LFO enable	0, 1(OFF)
[0B]	Portamento time	0~127
[0C]	Pitchbender range	0~12
[0D]	MONO/POLY mode	0,1 (MONO mode)
[0E]	PMD controller	0 = OFF 1 = Touch 2 = Wheel 3 = Breath 4 = Foot
[0F]	Reserved area	
[10]	LFO speed	0~127(Max)
[11]	Amplitude modulation depth	0~127
[12]	Pitch modulation depth	0~127
[13]	LFO Wave form	0~3
[14]	LFO load enable	0,1 (ON)
[15]	LFO sync	0,1 (ON)
[16]	Amplitude modulation Sensitivity	0~3
[17]	Pitch modulation Sensitivity	0~7
[18]~[3F]	Reserved area	

Figure 6-1. Instrument Parameter List

Node Parameter Change Message

This message changes the configuration data and system common parameters (node parameters). The format is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00010000>    [10]    Message number
<0ppppppp>            Parameter number
<0ddddddd>            Data
<11110111>    [F7]    End-of-command
```

Parameters that can be changed using the Node Parameter command are listed below.

Parameter Number	Parameter Name	Value
[00]~[07]	Name	
[08]	COMBINE mode	0,1 (ON)
[09]	LFO speed	0~127
[0A]	Amplitude modulation depth	0~127
[0B]	Pitch modulation depth	0~127
[0C]	LFO wave form	0~3
[0D]	Note number reception mode	0 (ALL) 1 (EVEN) 2 (ODD)
[0E]~[1F]	Reserved area	
[20]	Node number	0~15
[21]	Memory protect mode	0,1 (ON)
[22]	Configuration number	0~19
[23]	Master tune	-64~63*
[24]	Master output level	0~127
[25]	CHAIN mode	0,1 (Enable)
[26]~[7F]	Reserved area	

* 2's Complement (7 Bits)   ±100 cents range

Event List

The music card can specify fine intervals in units of 1/64th of a half-step. This resolution using standard MIDI Note ON/OFF messages is not possible. When the music card receives the 4 bytes, [F0], [43], [75], and [70], the following occur:

- Fraction data can be appended to the Note ON messages

- The music card can receive the event list messages.

Event list messages are continuously received until a MIDI status byte, excluding an [F7]

(end-of-command) or a real-time message is received. The format is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<01110000>    [70]    Node number
<0eeeeeee>            Event list
  . . . . . .
<0eeeeeee>
<11110111>    [F7]    End-of-command
```

Each event begins with an event status byte and has a length that ranges from 2 to 6 bytes, determined by the event type. The event status byte indicates the status section within the event list. The event status byte is referred to hereafter as a status byte, although the event status byte differs from the MIDI status byte. Following is a summary of event status bytes.

Event Status	Description
[0n]	Note OFF
[1n]	Note ON/OFF
[2n]	Note ON/OFF with duration
[3n]	Control change
[4n]	Program change
[5n]	After-touch
[6n]	Pitchbender range
[7n]	Parameter change

Format of Event List

The following tables show the format for the
Note OFF and Note ON/OFF event list
commands.

```
Note OFF
 <0000nnnn>    [0n]   Status byte
                      n=Channel number (0~15)
 <0kkkkkkk>           Note number
                      k=0 (C-2)~127 (G8)
 <0fffffff>           Fraction
                      f=0~127 (+100 cents)
Note ON/OFF
 <0001nnnn>    [1n]   Status byte
                      n=Channel number (0~15)
 <0kkkkkkk>           Note number
                      k=0 (C-2)~127 (G8)
 <0fffffff>           Fraction
                      f=0~127 (+100 cents)
 <0vvvvvvv>           Key velocity
                      v=1~127: Note ON
                      v=0: Note OFF
```

Note ON/OFF with Duration

When a non-zero value is specified for the
duration, the interval is counted by the MIDI
clock sent from the receiving instrument. The
note is played for the specified interval. The
receiving instrument does not acknowledge a
Note OFF message. Counting by MIDI clock
begins when the MIDI start message [FA] or
continue message [FB] is received. The MIDI
clock stops when the stop message [FC] is
received. The format is:

```
<0010nnnn>    [2n]    Status byte
                     n=Channel number (0~15)
<0kkkkkkk>            Note number
                     k=0 (C-2)~127 (G8)
<0fffffff>           Fraction
                     f=0~127 (+100 cents)
<0vvvvvvv>           Key velocity
                     v=1~127: Note ON
                     v=0: Note OFF
<0ddddddd>           Duration LSB
<0ddddddd>           Duration MSB
                     d=0: Note ON/OFF Only
```

Type A Parameter Change

This message changes the parameters of the
various instruments. The format is:

```
<0111nnnn>    [7n]    Status byte
                     n=Channel number (0~15)
<00pppppp>           Parameter number [0]~[3F]
<0ddddddd>           Data
```

Type B Parameter Change

This message changes the parameters of the
various instruments. Its data section consists of
2 bytes: a low-order 4-bit byte and a high-order
4-bit byte. The bytes are sent in a
lower-to-higher sequence. The format is:

```
<0111nnnn>    [7n]  Status byte
                    n=Channel number (0~15)
<01pppppp>          Parameter number [0]~[3F]
<0000dddd>          Data low
<0000dddd>          Data high
```

Control Change, After-Touch, and Pitchbend

The data ranging from control change to pitchbend has the same format as channel messages, with the exception of the status byte.

Store Request Messages

The store request message stores the current configuration data and voice data into the music card memory. Immediately after the [F7] (End-of-command) code is sent, and the operation is complete, the music card sends a handshaking message (ACK, NAK, or CANCEL) to the requester.

Store Voice Data

The store voice data message stores the voice
programmed at each instrument into memory.
The format is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00101iii>    [2x]    Message number
                      i=Instrument number
<01000000>    [40]    Format number
<0ddddddd>            Voice number
                      d=0~47 (Bank number 0)
                      d=48~95 (Bank number 1)
<11110111>    [F7]    End-of-command
```

Store Configuration Data

The store configuration data message stores the
currently programmed configuration data into
memory. The format is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00100000>    [20]    Message number
<01000000>    [40]    Format number
<0000dddd>            Configuration number
                      d=0~15
<11110111>    [F7]    End-of-command
```

Bulk Data Transfers

The bulk data transfer commands transfer large amounts of data, such as voice or configuration data, as quickly as possible.

The data is transferred in one or more blocks, called *packets*, and can be sent or received by the music card.

Packets

The data that is transferred by bulk data commands is contained in variable-length blocks, called packets. Each packet consists of a 2-byte data-length count, the data, and a checksum. The maximum amount of data that can be transferred in one packet is 4,096 bytes. However, several packets can be included in one bulk data transfer.

> **Note:** There must be a delay of at least 10 ms between packets in each bulk data transfer operation.

There are two types of packets, type A and type B. Type A sends 8-bit data. Type B sends 7-bit data. The formats for the two packet types are shown in the following figures.

Offset	Content	Description
0	<000hhhhh>	Byte count (high)
1	<01111110>	Byte count (low)
2	<0000dddd>	First data byte, low half
3	<0000dddd>	First data byte, high half
.	.	
.	.	
n-2	<0000dddd>	Last data byte, low half
n-1	<0000dddd>	Last data byte, high half
n	<0ccccccc>	Checksum

Figure 6-2. Packet Type A (for 8-bit data)

Offset	Content	Description
0	<000hhhhh>	Byte count (high)
1	<01111111>	Byte count (low)
2	<0ddddddd>	First data byte
.	.	
.	.	
n-1	<0ddddddd>	Last data byte
n	<0ccccccc>	Checksum

Figure   6-3.   Packet Type B (for 7-bit data)

In the formats shown, byte count refers to the length of the data section (n-2). Values can range from 1 through 4,096 ([001] through [1000]). The checksum is the low-order 7 bits of the 2's complement of the arithmetic sum of the data section (offset 2 through n-1). The following examples illustrate packets for transferring the 4 bytes, [12], [34], [56], and [78], in two 2-byte packets.

[00] [04] [02] [01] [04] [03] [76] (1st packet)
[00] [04] [06] [05] [08] [07] [66] (2nd packet)

Figure   6-4.   Type A Packets

```
[00] [02] [12] [34] [3A]        (1st packet)
[00] [02] [56] [78] [32]        (2nd packet)
```

Figure 6-5.   Type B Packets

The checksum in the first type A packet ([76]) is
calculated as follows:

- The sum of the data section is [OA]

- The 2's complement of this sum is [F6]

- The low-order 7 bits of this value is the
  checksum value of [76].

The checksums in the other packets are
calculated in the same way.

Bulk Transfers to the Music Card

There are two types of data transfers: node and
instrument. These transfers are described in this
section. A voice bank data packet is used as an
example. The detailed format for voice data is
described in Figure A-3 on page A-4 and
Figure A-4 on page A-5, but a summary is given
below.

The voice bank data consists of a header
followed by data for all 48 voices contained in
the bank.

The format of the voice bank data is:

Offset	Content
[000]..[007]	ASCII name parameter (part 1 of header)
[008]..[01F]	Not used, must be [00] (part 2 of header)
	Voice data (48 voices, 64 bytes/voice):
[020]..[03F]	Voice 0 data
[040]..[04F]	Voice 1 data
. .	.
. .	.
. .	.
[C20]..[C3F]	Voice 47 data

Node Messages

This message can transfer and specify the destination of five types of data. Data type is indicated through the format number command byte, and data destination is indicated in the destination number command byte. The format is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00000000>    [00]    Message number
<0fffffff>            Format number
<0ddddddd>            Destination number
  ......      ....    Packet(s)
  ......      ....
<11110111>    [F7]    End-of-command
```

The meanings of the format and destination numbers are:

Format Number	Data Type	Destination Number	Pac- ket
0	Voice bank data	Bank number (0 or 1)	A
1	Configuration-1	Must be 0	B
2	Configuration memory	Configuration (0~15)	B
3	Configuration RAM	Must be 0	B
6	Configuration-2	Must be 0	B

Configuration-1, configuration-2, and configuration memory are 160-byte data blocks described by the Configuration Data Format table in Appendix A. The configuration RAM refers to all 16 user-programmable configurations (16 x 160 bytes each = 2,560 bytes). Configuration-1 and configuration-2 differ in that configuration-1 updates the configuration data, including the voice function data and the LFO, while configuration-2 does not affect the voice function data if COMBINE mode is off and will not affect the LFO if LFO load mode (of voice data) is off. Configuration memory refers to the content of one of the 16 programmable partitions in the configuration RAM.

Instrument Messages

Instrument messages allow the replacement of voice data or configuration data for one of the eight possible instruments. There are three types of instrument messages:

- Instrument voice

- Instrument configuration-1

- Instrument configuration-2.

The format of the message is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00001iii>    [0x]    Message number
                      i=Instrument number
<000000ff>    [0f]    Format number
                      f=0~2
<00000000>    [00]    Destination number
......        ....    Packet(s)
......        ....
<11110111>    [F7]    End-of-command
```

The meaning of the format number is:

Format Number	Data Type	Packet
0	Instrument voice	A
1	Instrument configuration-1	B
2	Instrument configuration-2	B

Instrument voice refers to the 64 bytes of voice data for one of the eight instruments, and instrument configuration data refers to the 16 bytes of configuration data for an instrument. Note that format number 1 always updates the voice function data of the instrument, while format number 2 updates only the voice function data if COMBINE mode is ON.

Bulk Transfers from the Music Card

The content of voice or configuration data may
be read from the music card using the dump
request commands. Immediately after receiving
the dump request, the music card sends the
requested data in packet format, or returns a
CANCEL handshake if the dump request
contained an error. Dump requests can be node
messages or instrument messages. A special node
dump request message is provided to return the
ID of the music card.

Node Dump Request Message

This message requests a dump of the data from
one of six sources. The format of the node dump
request message is:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00100000>    [20]    Message number
<00000xxx>    [0x]    Source number
                      x=0~4, 6
<000yyyyy>    [yy]    Source ID
<11110111>    [F7]    End-of-command
```

The source number can be one of the following
six sources:

Source 0 (Voice Memory Bank): This is one
of the on-card voice memory banks. The bank to
be transferred is selected with the source ID byte
(0~6). The music card responds with a header
packet followed by 48 voice data packets in the
following format (all packets are type A):

[F0] [43] [75] [0s] [00] [00] [bank number]
      (packet #1)...(packet #49) [F7]

Sources 1 and 6 (Configuration Buffer):
These source numbers operate identically, and
specify the current content of the configuration
buffer. The source ID byte should be 0.

The 160 bytes of data are returned in a type B
packet as follows:

Source 1:
    [F0] [43] [75] [0s] [00] [01] [00] (packet) [F7]
Source 6:
    [F0] [43] [75] [0s] [00] [06] [00] (packet) [F7]

Source 2 (Individual Configuration): This
source is a particular configuration within the 16
programmable configurations in card memory.
The configuration number to be sent is specified
with the source ID byte (0 through 19). The 160
bytes of data are returned in a type B packet as
follows:

[F0] [43] [75] [0s] [00] [02] [configuration number]
            (packet) [F7]

Source 3 (Configuration RAM): This source
is the entire content of the programmable
configuration RAM. The source ID byte should
be 0. Sixteen packets of 160 bytes each are
returned according to the following format:

[F0] [43] [75] [0s] [00] [03] [00] (packet #1) ...
            (packet #16) [F7]

Source 4 (Music Card ID): A type B packet of 16 bytes is returned, containing an ASCII string. The source ID byte should be 0. For the music card, the string is "YAMAHA IBM MUSIC" and is returned in the following format:

[F0] [43] [75] [0s] [00] [04] [00] (packet) [F7]

Instrument Dump Request Messages

The instrument dump request message causes the music card to return the content of voice or configuration data for a particular instrument. The format of the message is as follows:

```
<11110000>    [F0]    Status byte
<01000011>    [43]    Mfg ID
<01110101>    [75]    Sub-status
<0000ssss>    [0s]    Node number
                      s=0~15
<00101iii>    [2x]    Message number
                      i=instrument (0~7)
<000000yy>    [0y]    Source ID
<00000000>    [00]
<11110111>    [F7]    End-of-command
```

Source ID 0 is the voice data for one of the instruments in the current configuration. Source IDs 1 and 2 are equivalent and specify the configuration data for one of the instruments in the current configuration.

The 64 bytes of instrument voice data are returned in a type A packet in the following format:

[F0] [43] [75] [0s] [0x] [00] [00] (packet) [F7]

The 16 bytes of instrument configuration data are returned in a type B packet in the following formats:

[F0] [43] [75] [0s] [0x] [01] [00] (packet) [F7]
[F0] [43] [75] [0s] [0x] [02] [00] (packet) [F7]

Handshaking Messages

If a dump message is received by the music card,
or if an operation message is sent in (for
example, a dump request or store command), this
message is used to inform the requesting side
whether an error has occurred. The format is:

```
<11110000>    [F0]   Status byte
<01000011>    [43]   Mfg ID
<0110ssss>    [6s]   Sub-status
                     s=Node number (0~15)
<00000mmm>           Message number*
<11110111>    [F7]   End-of-command
```

* See the following Message Number table.

Message Number	Message	Description
m = 2	ACK	Completed
m = 3	NAK	During data reception, one of the following errors has occurred: • Checksum error • FIFO overflow error • MIDI reception error • MIDI off-line error • Time-out error.
m = 4	Cancel	Execution not possible due to one of the following reasons: • Specified argument out of range • Write attempt (store/dump) was made to ROM area. • Write attempt (store/dump) was made to memory-protected RAM.

The following table summarizes the possibility of talk-back returned from the music card for the various system exclusive messages through the PC bus or MIDI IN.

Type of Operations	Talk Back from the Card			
	ACK	NAK	Cancel	Dump
Instrument parameter change	no	no	no	no
Node parameter change	no	no	no	no
Event list	no	no	no	no
Parameter list	no	no	no	no
Bulk dump request	no	no	yes	yes
Store request	yes	yes	yes	no
Bulk dump to music card	yes	yes	yes	no
(The following are illegal conditions)				
Node number mismatch	no	no	no	no
Undefined request number (operation/dump)	no	no	no	no

Summary of System Exclusive Messages

Following is a summary of the formats.

Channel Messages

Parameter change
[F0] [43] [1n] [15] ... [F7]

n = Channel number (0~15).

Instrument Messages

- Bulk dump to/from music card
  [F0] [43] [75] [0s] [0x] [0f] [00]...[F7]

- Parameter change
  [F0] [43] [75] [0s] [1x]... [F7]

- Bulk dump or store request to music card
  [F0] [43] [75] [0s] [2x] [0y] [00]...[F7]

Notes:

1. x = Instrument number + 8. (For instrument number 0, x = 8; for instrument number 7, x = 15.)

2. s = Node number (0~15).

3. For formats of [0f] and [0y] bytes, see the individual command descriptions.

Node Messages

- Talk-back (handshaking message) from music card
  [F0] [43] [6s]...[F7]

- Bulk dump data (packet) transfer to music card
  [F0] [43] [75] [0s] [00] [0f]...[F7]

- Parameter change
  [F0] [43] [75] [0s] [1x]...[F7]

- Bulk dump or store request to music card
  [F0] [43] [75] [0s] [20] [0x] [0y]...[F7]

  Note: s = Node number (0~15).

Event List Transfer to Music Card

[F0] [43] [75] [70]...[F7]

Parameter List Transfer to Music Card

[F0] [43] [75] [71]...[F7]

Chapter 7. Definitions of Configuration Data

This chapter provides descriptive information and hex values for node parameters, instrument parameters, and voice function parameters. Preset configuration data is summarized in tables in Appendix A.

Node Parameters

In this chapter, the numbers following the parameter names refer to the numbers used to change the parameters in the node parameter change command. See "Node Parameter Change Message" on page 6-9.

Configuration Name

[00] through [07]

A configuration name can be specified using a maximum of eight alpha or numeric characters.

COMBINE Mode

[08]

When switching voices, both the voice function data and the voice data that are stored together are loaded into the configuration buffer. When COMBINE mode is set to OFF (O), however, the voice function data is not loaded. Therefore, the voice function data, such as the pitchbender range, can be constantly fixed for each instrument.

LFO Control Block Data

[09] Through [0C]

See "Voice Function Parameters" on page 7-11.

Note Number Reception Mode

[0D]

The note number reception mode selects the reception mode of Note ON/OFF messages. The three reception modes are as follows:

0 (All) All note numbers are accepted. (Default)

1 (Even) Only even note numbers are accepted.

2 (Odd) Only odd note numbers are accepted.

Node Number

[20]

The node number distinguishes multiple music cards. When connecting and using multiple music cards, you can transmit bulk data to a specific music card by giving each music card a different node number. The node number is specified when sending or receiving either a system or an instrument message within a system exclusive message. The specified node number can range from 0 through 15.

Memory Protect Mode

[21]

The voice RAM and configuration RAM can be write-protected by using a memory protect function. When memory protect mode is ON, the voice RAM or configuration RAM cannot be written. When the memory protect mode is OFF, memory protection is cancelled. The power-on default is Protect OFF. To set memory protect mode, set 0 for Protect OFF, or 1 for Protect ON.

Configuration Number

[22]

The configuration to be used is specified with respect to the configuration memory by the configuration number. The configuration number can be within the range from 0 through 19.

Master Tune

[23]

The master tune is used to tune the music card to another instrument. The pitch can be changed within a half-step above or below the center value of zero. The setting values range from -64 through +63. The value -64 executes a half-step drop and +63 executes a half-step rise. The value is represented in 7-bit 2's complement.

Master Output Level

[24]

The master output level adjusts the output level of the entire music card. The values range from a minimum of 0 through a maximum of 127.

CHAIN Mode

[25]

When the music card is set with CHAIN mode ON, any Note ON messages that overflow the programmed note limit are output to the MIDI OUT terminal. Any messages other than a Note ON message are subjected to THRU processing. To set CHAIN mode, 0 sets CHAIN Mode OFF and 1 sets CHAIN Mode ON.

Instrument Parameters

Note: The numeric value following the parameter names in this section refer to the numbers used to change the parameter in the instrument parameter change command. (See Figure 6-1 on page 6-8.)

The music card is a sound generator system capable of playing eight notes concurrently. Each note can be played using a different voice by assigning separate voice data to the eight sound generators. The unit that produces one voice is called an instrument. The sound generator produces notes on the basis of data programmed for this instrument.

Number of Notes

[00]

The number of notes sets the maximum number of notes to be played by each instrument, from 0 through 8. If the number of notes to be played by the entire music card is already set to eight notes, an attempt to increase the number of notes for another instrument decreases the number of notes for the instrument with the largest number. Special attention is required, therefore, when changing the distribution of the number of notes.

Note: If the number of notes is 1, the monophonic mode is assumed. This differs slightly from specifying MONO using the POLY/MONO mode parameter. For a description of the differences, see "Channel Mode Messages" on page 5-5.

MIDI Channel Number

[01]

MIDI signals, such as Note ON/OFF, also include a channel number designed to operate only when the channel numbers of the sending and receiving sides match. This channel operation permits multiple-part performance using a single MIDI cable. A channel number can range from 0 through 15. Usually, a separate channel is specified for each instrument and each part is performed concurrently, but independently. When multiple instruments are set to the same channel, it is possible to execute the same performance using multiple voices, or to divide the scale and produce different voices with the treble and bass sections of the keyboard.

Note: If multiple instruments are set to the same MIDI channel number, the music card performs sequential processing from the smallest instrument number.

Note Number Limit High

[02]

The note number limit high sets the upper limit of the scale to be played at each instrument in a range from 0 (C-2) through 127 (G8) The note number limit high is usually set to 127.

Note Number Limit Low

[03]

The note number limit low sets the lower limit of
the scale to be played at each instrument in a
range from 0 (C-2) through 127 (G8). The note
number limit low is usually set to 0.

The correspondence between these high and low
limits and the keyboard positions are illustrated
in the following figure.
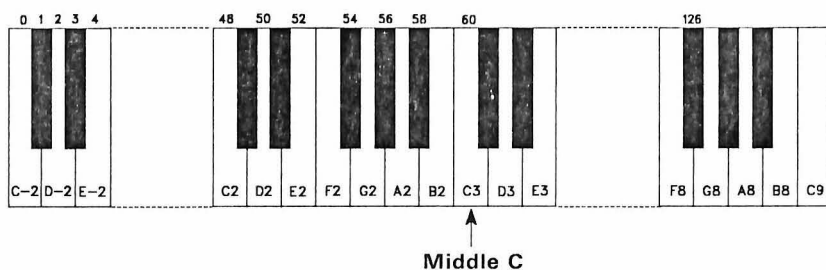
Figure 7-1. Key-to-Note-Number Relationship

Voice Bank Number

[04]

The music card is capable of storing up to 336
voices at once. However, only 128 voice types
can be selected using MIDI. Therefore, the
music card groups the voice data into seven
banks of 48 voices each. The voice bank number
specifies the bank to which the voice is assigned
and can be set in a range from 0 through 6. The
voice is determined by this voice bank number
parameter and the voice number parameter
described below.

Note: Although the programmed voice number remains unchanged when the bank number is switched, the voice itself changes to the corresponding voice within the new bank number.

Voice Number

[05]

The voice number selects the voice in the active voice bank for each instrument and can be set in a range from 0 through 47.

Detune

[06]

The detune performs a minute change in the pitch of each instrument. When two instruments are set to the same MIDI channel, the instruments are played concurrently by one KEY ON code. By minutely varying the pitch of the two instruments, a richer, more expansive sound can be achieved. The pitch can be changed within a range of a half-step above or a half-step below the center value of 0, using a value ranging from -64 through +63.

Octave Transpose

[07]

The octave transpose changes the octave of each instrument by one or two octaves. If two instruments are set to the same voice and the octave of one instrument is transposed, playing both instruments simultaneously produces a full, thick sound.

Value	Function
0	2 octaves down
1	1 octave down
2	Center
3	1 octave up
4	2 octaves up

Output Level

[08]

The output level adjusts the output level of each instrument. The output level can be set in a range from 0 through 127 (127 indicates the maximum output). The output level is used to balance the volume among multiple instruments.

Pan

[09]

The music card has two audio output terminals, L and R. The output for each instrument can be directed to L, R, or L + R. The setting values are as follows:

Value	Pan
0	L
64	L + R
127	R

LFO Enable Mode

[0A]

The music card can generate a maximum of eight notes at the same time. However, the vibrato and tremolo effects are controlled by one LFO. Therefore, when a deep pitch modulation is applied to an instrument, certain voices can be subjected to an extremely undesirable modulation. By setting the LFO Enable mode to OFF (1), no modulation is applied to that instrument. The value 1 sets the LFO Enable mode to OFF and 0 sets the mode to ON.

Voice Function Parameters

Note: The numeric values following the parameter names in this section refer to the numbers used to change the parameters in the instrument parameter change command. (See Chapter 6, "MIDI System Exclusive Messages" on page 6-1.)

The music card can set, in addition to voice data, functions such as pitchbender range or portamento time independently for each instrument. These functions change when the voice is changed. By setting the COMBINE mode to OFF, the set values remain unchanged, despite any voice changes.

Portamento Time

[0B]

Portamento is an effect by which the pitch slides smoothly from note to note. The music card allows the speed of portamento (the speed of the pitch slide) to be set in a range from 0 through 127 (0 corresponds to Portamento OFF). The higher the numeric value, the slower the portamento speed.

Notes:

1. In POLY mode, portamento is constantly applied regardless of the playing method.

2. In MONO mode, portamento is applied only when a note is switched ON while a preceding note is ON.

Pitchbender Range

[0C]

The pitchbender is a type of wheel controller built into a device, such as a synthesizer, used to modify pitch. The pitchbender range sets the width of pitch fluctuation generated by the pitchbend wheel. The value range is from 0 through 12. When the pitchbender range is set to 0, no pitch fluctuation occurs. Each time the pitchbender parameter is incremented by one, the variable width expands both a half-step higher and lower. When the pitchbender parameter is set to 12, the pitchbend is altered plus or minus one octave.

POLY/MONO Mode

[0D]

The POLY/MONO mode sets the performance mode to polyphonic or monophonic. The value 1 is the MONO mode and 0 is the POLY mode.

Note: If the number of notes parameter is set to 1, that instrument plays a monophonic performance. The differences between this example and the MONO mode specified by the POLY/ MONO mode parameter is described in the following sections.

POLY Mode

- An attack is always applied to the voice regardless of the playing method used.

- The last note specified is played first when last-note priority is used.

- Portamento is constantly applied when the status is Portamento ON.

MONO Mode

- The attack is not applied when legato style is used. (Legato style consists of pressing a keyboard key to switch the note ON before the key for the previous note is released to switch that note OFF.) A slurred sensation can be produced when the legato style is used.

- Last-note priority applies. However, when two preceding notes are memorized and a note is pressed and released while another is pressed continuously, the second note plays as a continuously-pressed note.

- Portamento is applied only when playing in legato style.

PMD Controller

[0E]

Vibrato is an effect that creates periodic fluctuations in pitch and is effective in enriching the sound of a voice. The vibrato effect can be produced with the music card by using the pitch modulation depth (PMD) function. The depth of pitch modulation is set to the optimum value for each voice. However, the pitch modulation can

be changed in real-time by using a controller attached to such devices as synthesizers. MIDI allows the four types of controllers specified, as follows.

Controller Name	Value
None	0
After-touch	1
Modulation wheel	2
Breath controller	3
Foot controller	4

The music card applies modulation using one LFO. Therefore, if modulation is applied on one instrument when multiple instruments are in use, modulation may also be applied to other instruments, depending on their voices. Switch LFO Enable to OFF for the instruments to which modulation is not desired.

Chapter 8. Voice Parameter Functions

This chapter presents descriptive information and hex values for overall parameters, LFO parameters, operator parameters, and envelope parameters. In this chapter, the numbers following the parameter names are the offsets used to change the parameters in the voice parameter change command. See Figure A-3 on page A-4 and Figure A-4 on page A-5.

Overall Parameters

Voice Name

[00] through [06]

A voice name can be a maximum of seven alpha or numeric characters.

Transpose

[0F]

This parameter performs transposition in half-step units. With a setting range from -128 through +127, it is represented in 8-bit 2's complement. When transpose causes the height of a note to exceed the sound generator's range, the music card adjusts the note higher or lower in octave units.

Algorithm and Modulator/Carrier Flags

[0C] (Algorithm)
[15] (Modulator/Carrier)

An algorithm refers to a combination of operators. With the music card, selection can be made from eight types of algorithmic patterns. The smaller the number of carriers and the greater the number of modulators, the more complex the voice changes, and the easier it is to produce noise components. With a setting range from 0 through 7, the corresponding algorithm patterns are as follows:
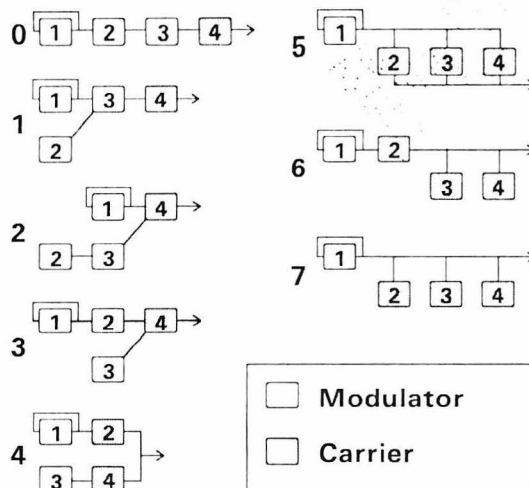
Figure 8-1. Algorithm Patterns

An operator is unconditionally determined by the algorithm to be a modulator or carrier, but the decision must be set in the Modulator/Carrier Flag. The value 0 specifies a modulator and 1 specifies a carrier. Be careful to avoid contradiction between this setting and the algorithm being selected.

Feedback Level

[0C]

Feedback is a function for feeding back frequency modulation to the input. The higher the amount of feedback, the greater the harmonic component and the larger the fluctuation in timbre. The setting range is from 0 through 7. Note that this function only operates with operator 1.

Low-Frequency Oscillator (LFO) Parameters

The LFO is used to modulate pitch or amplitude of a voice.

Note: If AMD or AMS is set to 0, LFO is not applied to amplitude. If PMD or PMS is set to 0, LFO is not applied to pitch.

LFO Wave Form

[0E]

This parameter selects the LFO waveform, which affects the way the amplitude or pitch is modulated. With a setting range of 0 through 3, the corresponding numeric values conform to the following figure:

0	ramp	
1	square	
2	triangle	
3	sample/hold	

Figure 8-2. LFO Waveform Values 0 through 3

LFO Speed

[08]

This sets the LFO speed (frequency). The frequency can be set from approximately 0.009 Hz through 59 Hz. Its programming range is 0 through 255. The larger the numeric value, the

higher the frequency, and the faster the speed of fluctuation.

Amplitude Modulation Depth (AMD)

[09]

This parameter sets the LFO output level with respect to the carrier level. The setting range is from 0 through 127. The larger the numeric value, the deeper the amplitude modulation.

Pitch Modulation Depth (PMD)

[0A]

This parameter sets the LFO output level with respect to the pitch. The setting range is from 0 through 127. The larger the numeric value, the deeper the pitch modulation.

LFO Sync Mode

[0A]

During KEY ON (the time a key is being pressed), the LFO can be synchronized. When LFO sync mode is ON, the leading edge of its waveform is synchronized to key on, as shown in the following figure. This parameter can be set to 1 (LFO sync mode ON) or 0 (LFO sync mode OFF).

LFO sync mode ON

KEY ON   KEY KEY
          OFF ON

LFO sync mode OFF

KEY ON   KEY KEY
          OFF ON
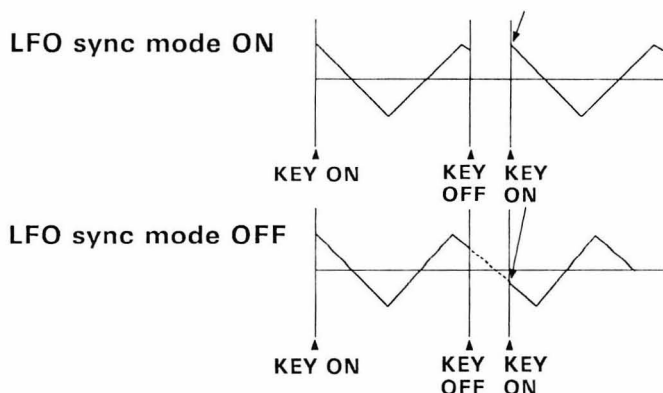
Figure 8-3. Wave Form Synchronization to KEY ON

LFO Load Mode

[09]

The LFO load mode parameter determines
whether LFO data stored in a voice will replace
current LFO data when the voice is selected.
Setting LFO load mode to 0 (OFF) prevents
current LFO data from being replaced if this
voice is selected.

Amplitude Modulation Sensitivity (AMS)

[0D]

When generating different voices at the same
time, the LFO effect can be enabled separately
for each voice. AMS sets the LFO sensitivity
with respect to amplitude for each sound
generator. The values range from 0 through 3 (3
indicates maximum sensitivity).

Pitch Modulation Sensitivity (PMS)

[0D]

Similar to AMS, this parameter can be programmed for each sound generator. It sets the LFO sensitivity with respect to pitch. The setting range is from 0 through 7 (7 indicates maximum sensitivity).

> **Note:** If AMD or AMS is set to 0, LFO is not applied to amplitude. If PMD or PMS is set to 0, LFO is not applied to pitch.

Operator Parameters

These parameters are programmable for each of the four operators.

Operator Enable

[0B]

Individual operators can be switched off. When switched OFF, the operator assumes a status identical to that when its output level is set to 0. Using the four bits of bits 6 through 3, the ON/OFF status of each operator can be set. The value of 1 enables the operator and 0 disables the operator.

Bit	Operator
7	0
6	op4
5	op3
4	op2
3	op1
2	0
1	0
0	0

Pitch Parameters

Pitch parameters affect the frequency of individual operators.

Multiple

[13]

This parameter sets the frequency of the operator to a ratio of operator pitch to keyboard standard pitch. Following is a description of the algorithm that is used:

For a Multiple value of 0, the operator pitch is one-half the keyboard standard pitch. For Multiple values of 1 through 15, the operator pitch is the product of the multiple value and the keyboard standard pitch.

Inharmonic

[16]

Similar to the Multiple parameter (described above), this parameter affects the frequency of each operator. Inharmonic is used, however, to allow non-integer multiplication of the frequency ratio. The values range from 0 through 3, and correspond to the multipliers listed below. For

example, if Multiple = 2 and Inharmonic = 3, the frequency ratio becomes:

2 x 1.73 = 3.46.

Set Value	Frequency Ratio
0	1.00X
1	1.41X
2	1.57X
3	1.73X

Detune (DT)

[13]

This function minutely detunes the pitch of each operator and can be performed independently for each operator. When the pitch of the carrier and modulator are minutely detuned, an effect such as phasor can be created. Using an algorithmic pattern with multiple carriers, a chorus effect can also be created by minutely detuning each carrier. The values range from -3 through +3 and are coded as follows:

Value	Pattern
-3	< 111 >
-2	< 110 >
-1	< 101 >
0	< 000 >
1	< 001 >
2	< 010 >
3	< 011 >

Envelope Parameters

The envelope plays a vital role in the creation of sound. For each operator, the music card has a built-in Envelope Generator (EG). Each EG is applied to its corresponding operator output to control temporal fluctuation in amplitude or timbre.

The envelope is programmed with the five parameters illustrated below.
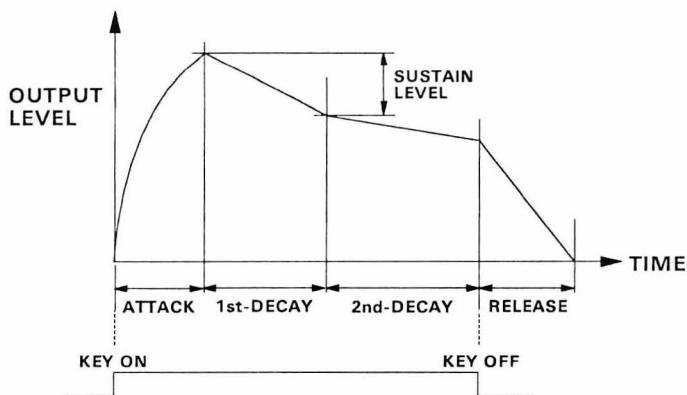
Figure 8-4. Parameters Programmed for the Envelope

The EG of an operator programmed as a carrier controls temporal fluctuations of amplitude. The EG of an operator programmed as a modulator controls temporal fluctuations of timbre.

Attack Rate (AR)

[14]

This parameter sets the rate of increase from the moment a key is pressed until the output level of EG reaches its maximum value. The values range from 0 through 31. (The value 31 indicates the maximum fluctuation speed, and 0 indicates no fluctuation.)

Decay 1 Rate (D1R)

[15]

This parameter sets the decay rate from the
moment the EG output level reaches its
maximum value until it reaches the level set by
sustain level. The values range from 0 through
31. (The value 31 indicates the maximum
fluctuation speed, and 0 indicates no decay 1 is
applied.)

Sustain Level (SL)

[17]

This parameter sets the attenuation level
between the first decay rate and the second
decay rate. The values range from 0 through 15
(0 indicates no attenuation).

Decay 2 Rate (D2R)

[16]

This sets the decay from the moment the EG
output level reaches the level set by sustain level
until it becomes 0. The values range from 0
through 31. (The value 31 indicates the shortest
decay, and 0 indicates no decay.)

Release Rate (RR)

[17]

This parameter sets the decay from the moment a
key is released until the output level becomes 0.
The envelope set by AR through D2R is produced
while the key is being pressed. However, when
the key is released, the output level drops from
the current output level towards 0. The values

range from 0 through 15 (15 indicates the fastest decay).

Total Level (TL)

[10]

This parameter sets the output level of each operator. According to this output level, the FM sound generator controls not only the amplitude, but also the basic timbre. Changing the carrier output level changes the amplitude, and changing the modulator output level changes the timbre. The values range from 0 through 127 (0 indicates the maximum output level).

Keyboard Rate Scaling

[14]

This parameter affects the rates (AR, D1R, D2R, and RR) of the envelope. The keyboard rate scaling is carried out by hardware internal to the FM sound generator chip. The purpose of keyboard rate scaling is to better simulate natural sounds. The envelope of a sound from a piano, for example, varies according to its pitch. Sharper attack and decay are typical for higher-pitched notes. The affect is controlled by the keyboard rate scaling depth (KRSD) of each operator. The more depth, the faster the rates and the sharper the envelope shape as the key code number increases.

Keyboard Level Scaling

[11] Bit 0
[13] Bit 1
[12] Scaling Depth

The purpose of keyboard level scaling stems from the characteristics of most existing sounds, where the loudness (level of carrier) and spectra

(level of modulator) varies as the pitch of the sound varies.

Keyboard scaling and scaling depth affect the level of each operator output according to the keycode number (KC#).

When a scaling value corresponding the KC# is taken out of the curve, it is multiplied by the keyboard level scaling depth to yield the key-scale dependent offset (adjusted by depth of effect). Depth serves as a sensitivity adjustment for level scaling (up to -24 dB or -48 dB).

Bit 1	Bit 2	Type of Scaling Curve
0	0	Linear attenuation with increasing KC#; maximum of -24 dB per 8 ocatves.
0	1	Reverse S-shaped attenuation with decreasing KC#; maximum of -24 dB per 8 octaves.
1	0	Non-linear attenuation with increasing KC#; maximum of -48 dB per 8 octaves.
1	1	Reverse S-shaped attenuation with decreasing KC#; maximum of -48 dB per 8 octaves.

Velocity Parameters

The amplitude and timbre of a piano vary according to the pressure with which the keys are pressed. Certain synthesizers have the capability to express this feature through MIDI. A touch response function, for example, detects the velocity with which the keys are pressed and controls the amplitude and timbre in response to that velocity data. The music card also controls the amplitude and timbre according to velocity data. It has an additional function for controlling the rise time of notes.

Velocity Sensitivity to Total Level (VSTL)

[11]

This parameter changes the output level of an operator with respect to velocity data. Because control of individual operators is possible, raising the VSTL of a carrier changes the amplitude and raising the VSTL of a modulator changes the timbre. The values range from 0 through 7 (7 indicates maximum sensitivity).

Velocity Sensitivity to Attack Rate (VSAR)

[15]

This parameter changes the attack rate of an envelope generator with respect to velocity data. The values range from 0 through 3. At 3, the sensitivity to keyboard pressure is maximum.

Voice Functions

These functions are defined in "Voice Function Parameters" on page 7-11.

Chapter 9. Multiple Card Usage

This chapter illustrates a typical configuration and usage of multiple music cards in a system.

Multiple Card Configuration

The following figure is an example of complex connections of multiple music cards and external MIDI devices. The connection between the PC and the music card is through the PC parallel bus. All other connections are made through MIDI. It is important to note that logical communication is accomplished through MIDI protocol.
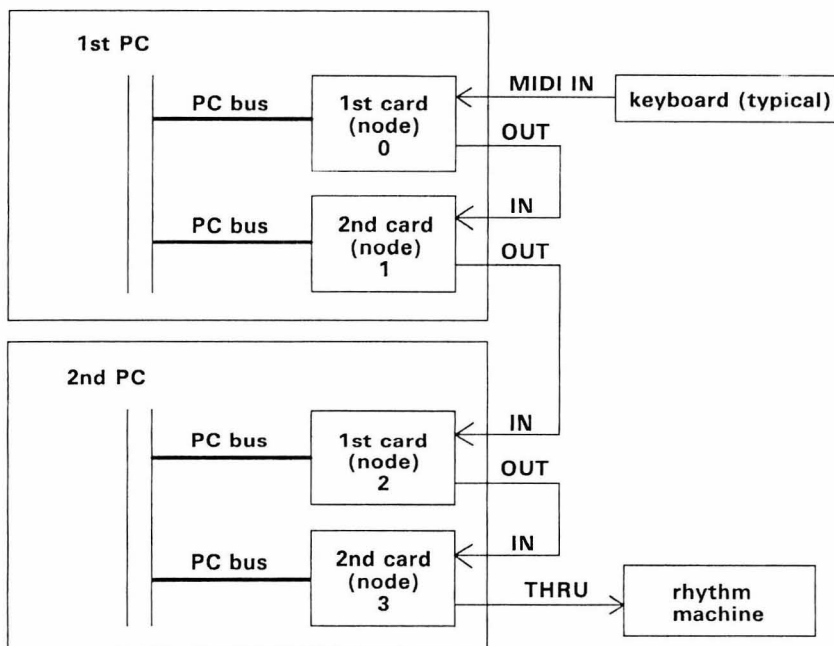
Figure 9-1. Connections of Multiple Cards and External MIDI Devices

Interrupt Level Setting with Multiple Music Cards

The following figure shows how interrupts are used in a multiple music card arrangement.
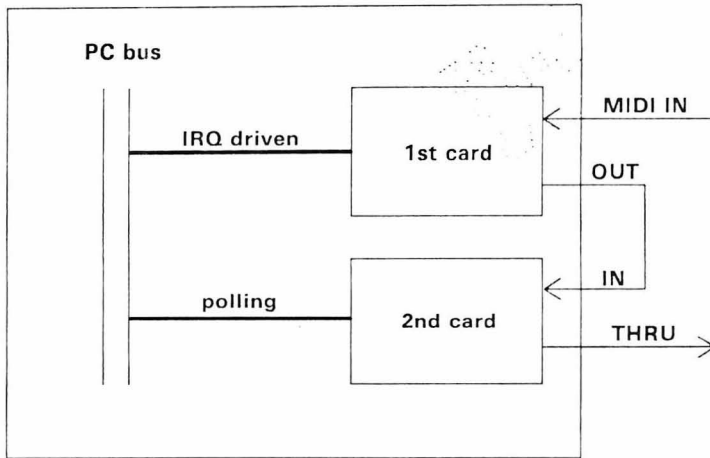
Figure 9-2. Interrupt Capability in a Multiple Music Card Arrangement

Since the music card does not support hardware interrupt sharing, the interrupt level must be unique to the music card. The first music card can be set anywhere between interrupt level 2 and 7 as long as it does not conflict with another adapter card. The second music card can be set to any other unused interrupt request level.

If there are no available Interrupt Request Levels (IRQ) left, remove the interrupt jumper on the music card. This disconnects the interrupt request line.

Multiple music cards can share the same interrupt level if the application or software driver assures that the conflicting music cards are not enabled simultaneously.

Appendix A. Data Format Tables

Offset	Parameter Name	Data
[00]~[07]	Configuration name	ASCII
[08]	Combine mode	0,1 (Combine ON)
[09]	LFO speed	0~127
[0A]	Amplitude modulation depth	0~127
[0B]	Pitch modulation depth	0~127
[0C]	LFO wave form	0~3
[0D]	Note number reception mode	0 = All, 1 = Even, 2 = Odd
[0E]~[1F]	Reserved	0
	Instrument 0	
[20]	Number of notes	0~8
[21]	MIDI channel number	0~15
[22]	Note number limit/H	0~127
[23]	Note number limit/L	0~127
[24]	Voice bank number	0~6
[25]	Voice number	0~47
[26]	Detune	-64~63 (2's complement)
[27]	Octave transpose	0~4 (2 = Center)
[28]	Output level	0~127
[29]	Pan	0 = L, 64 = L + R, 127 = R
[2A]	LFO enable mode	0,1 (OFF)
[2B]	Portamento time	0~127
[2C]	Pitchbender range	0~12
[2D]	MONO/POLY mode	0,1 (MONO mode)
[2E]	PMD controller	0 = OFF, 1 = Touch, 2 = Wheel, 3 = Breath, 4 = Foot
[2F]	Reserved	
[30]~[3F]	Instrument 1	
....	....	
[90]~[9F]	Instrument 7	

Figure A-1. Configuration Data Format

Offset	Content
[000]...[007]	ASCII name parameter
[008]...[01F]	Not used, must be < 0 >
[020]...[03F]	Voice 0 data*
[040]...[05F]	Voice 1 data
.	.
.	.
.	.
.	.
[C20]...[C3F]	Voice 47 data

Figure   A-2.   Voice Bank Data Format

* See Figure A-3 on page A-4 and Figure A-4 on page A-5.

Offset	Parameter		Value
[00]~		Voice name	ASCII
[06]			
[07]		Reserved	
[08]	<********>	LFO speed	0~255
[09]	<a*******>	<a>LFO load mode	0,1 (Enable)
		<*>Amplitude	0~127
		modulation	
		depth (AMD)	
[0A]	<a*******>	<a>LFO sync mode	0,1 (Sync ON)
		<*>Pitch	0~127
		modulation	
		depth (PMD)	
[0B]	<0abcd000>	<*>Operator	0,1 (ON)
		enable	
		a=operator #4	
		....	
		d=operator #1	
[0C]	<11aaabbb>	<a>Feedback	0~7
		level	
		<b>Algorithm	0~7
[0D]	<0aaa00bb>	<a>Pitch	0~7
		modulation	
		sensitivity	
		(PMS)	
		<b>AMS	0~3
[0E]	<0**00000>	<*>LFO wave form	0~3
[0F]	<********>	Transpose	-128~127
	Operator 1		
[10]	<0*******>	Total level	0(Max)~127
		(TL)	
[11]	<a***0000>	<a>Keyboard	0,1
		level scaling	
		type	
		(KLS Bit0)	

Figure   A-3.   Voice Data Format (Part 1 of 2)

Offset	Parameter		Value
		<*>Velocity sensitivity to total level (VSTL)	0~7
[12]	<aaaa0000>	<a>Keyboard level scaling depth (KLSD)	0~15
[13]	<abbbcccc>	<a>KLS bit 1	
		<b>Detune (DT)	0~7
		<c>Multiple	0~15
[14]	<aa0*****>	<a>Keyboard rate scaling depth (KRSD)	0~3
		<*>Attack rate (AR)	0~31
[15]	<abb*****>	<a>Modulator/ Carrier	0=modulator 1=carrier
		<b>Velocity sensitivity to attack rate (VSAR)	0~3
		<*>Decay 1 rate (D1R)	0~31
[16]	<aa0*****>	<a>Inharmonic	0~3
		<*>Decay 2 rate (D2R)	0~31
[17]	<aaaabbbb>	<a>Sustain level (SL)	0(Max)~15
		<b>Release rate (RR)	0~15
[18~1F]	Operator 2		
[20~27]	Operator 3		
[28~2F]	Operator 4		
[30~39]	Reserved		

Figure   A-4.   Voice Data Format (Part 2 of 2)

	ROM1 (Bank2)	ROM2 (Bank3)	ROM3 (Bank4)	ROM4 (Bank5)	ROM5 (Bank6)
00	Brass	UpPiano	Horn2	FnkSyn2	JOrgan1
01	Horn	Spiano	Horn3	FnkSyn3	JOrgan2
02	Trumpet	Piano2	Horns	SynOrgn	COrgan1
03	LoStrig	Piano3	Flugelh	SynFeed	COrgan2
04	Strings	Piano4	Trombon	SynHarm	EOrgan3
05	Piano	Piano5	Trumpt2	SynClar	EOrgan4
06	NewEP	PhGrand	Brass2	SynLead	EOrgan5
07	EGrand	Grand	Brass3	HuffTak	EOrgan6
08	Jazz Gt	DpGrand	HardBr1	SoHeavy	EOrgan7
09	EBass	LPiano1	HardBr2	Hollow	EOrgan8
10	WodBass	LPiano2	HardBr3	Schmooh	SmlPipe
11	EOrgan1	EGrand2	HardBr4	MonoSyn	MidPipe
12	EOrgan2	Honkey1	HuffBrs	Cheeky	BigPipe
13	POrgan1	Honkey2	PercBr1	SynBell	SftPipe
14	POrgan2	Pfbell	PercBr2	SynPluk	Organ
15	Flute	PfVibe	String1	EBass3	Guitar
16	Picolo	NewEP2	String2	RubBass	Folk Gt
17	Oboe	NewEP3	String3	SolBass	PluckGt
18	Clarine	NewEP4	String4	PlukBas	BriteGt
19	Glocken	NewEP5	SoloVio	UprtBas	Fuzz Gt
20	Vibes	EPiano1	RichSt1	Fretles	Zither2
21	Xylophn	EPiano2	RichSt2	FlapBs	Lute
22	Koto	EPiano3	RichSt3	MonoBas	Banjo
23	Zither	EPiano4	RichSt4	SynBas1	SftHarp
24	Clav	EPiano5	Cello1	SynBas2	Harp2
25	Harpsic	HighTin	Cello2	SynBas3	Harp3
26	Bells	HardTin	LoStrg3	SynBas4	SftKoto
27	Harp	PercPf	LoStrg4	SynBas5	HitKoto
28	SmadSyn	WoodPf	LoStrg5	SynBas6	Sitar1
29	Harmoni	EPStrng	Orchest	SynBas7	Sitar2
30	SteelDr	EPBrass	5th Str	Marimb2	HuffSyn
31	Timpani	Clav2	Pizzic1	Marimb3	Fantasy
32	LoStrg2	Clav3	Pizzic2	Xyloph2	Synvoic
33	Horn Lo	Clav4	Flute2	Vibe2	M.Voice
34	Whistle	FuzzClv	Flute3	Vibe3	VSAR
35	zingPlp	MuteClv	Flute4	Glockn2	Racing
36	Metal	MuteCl2	Pan Flt	TubeBe1	Water
37	Heavy	SynClv1	SlowFlt	TubeBe2	WildWar
38	FunkSyn	SynClv2	5th Flt	Bells 2	Ghostie
39	Voices	SynClv3	Oboe2	TempleG	Wave
40	Marimba	SynClv4	Bassoon	SteelDr	Space 1
41	EBass2	Harpsi2	Reed	ElectDr	SpChime
42	SnareDr	Harpsi3	Harmon2	HandDr	SpTalk
43	RD Cymb	Harpsi4	Harmon3	SynTimp	Winds
44	Tom Tom	Harpsi5	Harmon4	clock	Smash
45	Mars to	Circust	MonoSax	Heifer	Alarm
46	Storm	Celeste	Sax 1	SnareD2	Helicop
47	Windbel	Squeeze	Sax 2	SnareD3	SineWav

ROM2 categories: Piano, Electric Piano, Other Keyboard.
ROM3 categories: Brass, Strings, Woodwind.
ROM4 categories: Synthesizer, Bass, Percussion.
ROM5 categories: Organ, Plucked, Sound effect.

Figure A-5. Preset Voices Table

A-6 Technical Reference

CONFIGURATION NUMBER: **16**   NAME: **Single**

COMBINE MODE: **On**   NOTE NUMBER RECEPTION MODE: **All**

INSTRUMENT NUMBER	0	1	2	3	4	5	6	7
MIDI CHANNEL	0	1	2	3	4	5	6	7
NUMBER OF NOTES	8	0	0	0	0	0	0	0
NOTE NUMBER LIMIT HIGH	G8	G8	G8	G8	G8	G8	G8	G8
NOTE NUMBER LIMIT LOW	C-2	C-2	C-2	C-2	C-2	C-2	C-2	C-2
VOICE NUMBER	5	0	0	0	0	0	0	0
VOICE BANK NUMBER	2	2	2	2	2	2	2	2
OUTPUT LEVEL	127	127	127	127	127	127	127	127
OCTAVE TRANSPOSE	0	0	0	0	0	0	0	0
DETUNE	0	0	0	0	0	0	0	0
PAN	L+R	L+R	L+R	L+R	L+R	L+R	L+R	L+R
LFO ENABLE MODE	ON	ON	ON	ON	ON	ON	ON	ON
VOICE FUNCTION DATA:								
PITCHBENDER RANGE	5	2	2	2	2	2	2	2
PORTAMENTO TIME	0	0	0	0	0	0	0	0
MONO/POLY MODE	POLY	POLY	POLY	POLY	POLY	POLY	POLY	POLY
PMD CONTROLLER	WHL	WHL	WHL	WHL	WHL	WHL	WHL	WHL

Figure   A-6.   Preset Configuration Data: Configuration 16

CONFIGURATION NUMBER: **17**   NAME: **Mono 8**

COMBINE MODE: **On**   NOTE NUMBER RECEPTION MODE: **All**

INSTRUMENT NUMBER	0	1	2	3	4	5	6	7
MIDI CHANNEL	0	0	2	3	4	5	6	7
NUMBER OF NOTES	1	1	1	1	1	1	1	1
NOTE NUMBER								
LIMIT HIGH	G8	G8	G8	G8	G8	G8	G8	G8
NOTE NUMBER								
LIMIT LOW	C-2	C-2	C-2	C-2	C-2	C-2	C-2	C-2
VOICE NUMBER	0	1	2	3	4	5	6	7
VOICE BANK NUMBER	2	2	2	2	2	2	2	2
OUTPUT LEVEL	127	127	127	127	127	127	127	127
OCTAVE TRANSPOSE	0	0	0	0	0	0	0	0
DETUNE	0	0	0	0	0	0	0	0
PAN	L+R	L+R	L+R	L+R	L+R	L+R	L+R	L+R
LFO ENABLE MODE	ON	ON	ON	ON	ON	ON	ON	ON
VOICE FUNCTION DATA:								
PITCHBENDER RANGE	2	4	4	7	2	5	2	0
PORTAMENTO TIME	0	0	0	0	0	0	0	0
MONO/POLY MODE	POLY	POLY	POLY	POLY	POLY	POLY	POLY	POLY
PMD CONTROLLER	WHL	WHL	WHL	WHL	WHL	WHL	WHL	WHL

Figure   A-7.   **Preset Configuration Data: Configuration 17**

CONFIGURATION NUMBER: **18**   NAME: **Dual**

COMBINE MODE: **On**   NOTE NUMBER RECEPTION MODE: **All**

INSTRUMENT NUMBER	0	1	2	3	4	5	6	7
MIDI CHANNEL	0	0	2	3	4	5	6	7
NUMBER OF NOTES	4	4	0	0	0	0	0	0
NOTE NUMBER								
LIMIT HIGH	G8	G8	G8	G8	G8	G8	G8	G8
NOTE NUMBER								
LIMIT LOW	C-2	C-2	C-2	C-2	C-2	C-2	C-2	C-2
VOICE NUMBER	0	0	0	0	0	0	0	0
VOICE BANK NUMBER	2	2	2	2	2	2	2	2
OUTPUT LEVEL	127	127	127	127	127	127	127	127
OCTAVE TRANSPOSE	0	0	0	0	0	0	0	0
DETUNE	0	+4	0	0	0	0	0	0
PAN	L+R	L+R	L+R	L+R	L+R	L+R	L+R	L+R
LFO ENABLE MODE	ON	ON	ON	ON	ON	ON	ON	ON
VOICE FUNCTION DATA:								
PITCHBENDER RANGE	2	2	2	2	2	2	2	2
PORTAMENTO TIME	0	0	0	0	0	0	0	0
MONO/POLY MODE	POLY	POLY	POLY	POLY	POLY	POLY	POLY	POLY
PMD CONTROLLER	WHL	WHL	WHL	WHL	WHL	WHL	WHL	WHL

Figure   A-8.   Preset Configuration Data: Configuration 18

CONFIGURATION NUMBER: **19**   NAME: **Split**

COMBINE MODE: **On**   NOTE NUMBER RECEPTION MODE: **All**

INSTRUMENT NUMBER	0	1	2	3	4	5	6	7
MIDI CHANNEL	0	1	2	3	4	5	6	7
NUMBER OF NOTES	4	4	0	0	0	0	0	0
NOTE NUMBER								
LIMIT HIGH	G8	B2	G8	G8	G8	G8	G8	G8
NOTE NUMBER								
LIMIT LOW	C3	C-2	C-2	C-2	C-2	C-2	C-2	C-2
VOICE NUMBER	0	0	0	0	0	0	0	0
VOICE BANK NUMBER	2	2	2	2	2	2	2	2
OUTPUT LEVEL	127	127	127	127	127	127	127	127
OCTAVE TRANSPOSE	0	0	0	0	0	0	0	0
DETUNE	0	0	0	0	0	0	0	0
PAN	L+R	L+R	L+R	L+R	L+R	L+R	L+R	L+R
LFO ENABLE MODE	ON	ON	ON	ON	ON	ON	ON	ON
VOICE FUNCTION DATA:								
PITCHBENDER RANGE	2	2	2	2	2	2	2	2
PORTAMENTO TIME	0	0	0	0	0	0	0	0
MONO/POLY MODE	POLY	POLY	POLY	POLY	POLY	POLY	POLY	POLY
PMD CONTROLLER	WHL	WHL	WHL	WHL	WHL	WHL	WHL	WHL

Figure   A-9.   Preset Configuration Data: Configuration 19

Appendix B. Technical Specifications

This chapter contains hardware technical specifications for the IBM PC Music Feature.

Power Requirement

+5	Vdc(+/- 5%)	:450 mA
+12	Vdc(+/- 5%)	:200 mA

PC Bus Pin Assignment (A-Side)

Pin	Symbol	Signal Name	Function
A1		Not used	–
A2	SD7	(System data bus)	I/O
A3	SD6	(System data bus)	I/O
A4	SD5	(System data bus)	I/O
A5	SD4	(System data bus)	I/O
A6	SD3	(System data bus)	I/O
A7	SD2	(System data bus)	I/O
A8	SD1	(System data bus)	I/O
A9	SD0	(System data bus)	I/O
A10		Not used	–
A11	AEN	(Address enable)	I
A12		Not used	–
A13		Not used	–
A14		Not used	–
A15		Not used	–
A16	SA15	(System address bus)	I
A17	SA14	(System address bus)	I
A18	SA13	(System address bus)	I
A19	SA12	(System address bus)	I
A20	SA11	(System address bus)	I
A21	SA10	(System address bus)	I
A22	SA9	(System address bus)	I
A23	SA8	(System address bus)	I
A24	SA7	(System address bus)	I
A25	SA6	(System address bus)	I
A26	SA5	(System address bus)	I
A27	SA4	(System address bus)	I
A28	SA3	(System address bus)	I
A29	SA2	(System address bus)	I
A30	SA1	(System address bus)	I
A31	SA0	(System address bus)	I

I - Input, O - Output, I/O - Input/Output

PC Bus Pin Assignment (B-Side)

Pin	Symbol	Signal Name	Function
B1	GND	(Ground)	I
B2	RES	(Reset)	I
B3	+ 5 Vdc	(Power)	I
B4	IRQ2(9)	(Interrupt level)	O
B5		Not used	—
B6		Not used	—
B7		Not used	—
B8		Not used	—
B9	+ 12 Vdc	(Power)	I
B10	GND	(Ground)	I
B11		Not used	—
B12		Not used	—
B13	-IOW	(I/O Write)	I
B14	-IOR	(I/O Read)	I
B15		Not used	—
B16		Not used	—
B17		Not used	—
B18		Not used	—
B19		Not used	—
B20		Not used	—
B21	IRQ7	(Interrupt level)	O
B22	IRQ6	(Interrupt level)	O
B23	IRQ5	(Interrupt level)	O
B24	IRQ4	(Interrupt level)	O
B25	IRQ3	(Interrupt level)	O
B26		Not used	—
B27		Not used	—
B28		Not used	—
B29	+ 5 Vdc	(Power)	I
B30		Not used	—
B31	GND	(Ground)	I

I - Input, O - Output, I/O - Input/Output

Card D Shell Connector Pin Assignment

Pin	Symbol	Signal Name	Function
1	TxC	GND	for MIDI OUT (DC coupling)
2	TxA	+ Current	for MIDI OUT
3	ThC	GND	for MIDI THRU (DC coupling)
4	RxB	+ Current	for MIDI IN
5	RxC	GND	for MIDI IN (AC coupling)
6	TxB	- Current	for MIDI OUT
7	ThB	- Current	for MIDI THRU
8	ThA	+ Current	for MIDI THRU
9	RxA	- Current	for MIDI IN
E	FG	PC Frame Ground	

Note:

+ **Current** Current outflow from the music card

- **Current** Current inflow into the music card

E Connector metal shield

Signal levels correspond to MIDI Specification 1.0.

MIDI Adapter D Shell Connector Pin Assignment

Pin	Symbol	Signal Name	Function
1	TxC	GND	for MIDI OUT
2	TxA	+ Current	for MIDI OUT
3	ThC	GND	for MIDI THRU
4	RxB	+ Current	for MIDI IN
5	RxC	GND	for MIDI IN
6	TxB	- Current	for MIDI OUT
7	ThB	- Current	for MIDI THRU
8	ThA	+ Current	for MIDI THRU
9	RxA	- Current	for MIDI IN
E	FG	Cable shield	

Note:

+ Current Current outflow from the music card

- Current Current inflow into the music card

E Connector metal shield

Signal levels correspond to MIDI Specification 1.0.

MIDI Adapter MIDI IN Pin Assignment

Pin	Symbol	Signal Name	Function
1		Not used	
2	RxC	GND	for MIDI IN
3		Not used	
4	RxA	- Current	for MIDI IN (Normally 5mA)
5	RxB	+ Current	for MIDI IN (Normally 5mA)
E	FG	Case ground	

Note:

+ **Current** Current outflow from the music card

- **Current** Current inflow into the music card

E Connector metal shield

Signal levels correspond to MIDI Specification 1.0.

MIDI Adapter MIDI OUT Pin Assignment

Pin	Symbol	Signal Name	Function
1		Not used	
2	TxC	GND	for MIDI OUT
3		Not used	
4	TxA	+ Current	for MIDI OUT (Normally 5mA)
5	TxB	- Current	for MIDI OUT (Normally 5mA)
E	FG	Case ground	

Note:

+ Current Current outflow from the music card

- Current Current inflow into the music card

E Connector metal shield

Signal levels correspond to MIDI Specification 1.0.

MIDI Adapter MIDI THRU Pin Assignment

Pin	Symbol	Signal Name	Function
1		Not used	
2	ThC	GND	for MIDI THRU
3		Not used	
4	ThA	+ Current	for MIDI THRU (Normally 5mA)
5	ThB	- Current	for MIDI THRU (Normally 5mA)
E	FG	Case ground	

Note:

+ Current	Current outflow from the music card
- Current	Current inflow into the music card
E	Connector metal shield

Signal levels correspond to MIDI Specification 1.0.

Analog Outputs

Audio Output Channels (Left/Right)

The white jack is for audio left channel output.
The red jack is for the right channel output.

Characteristics	Value
Max output level (1KHz, 0.3%, 47KΩ)	1.8 V r.m.s.
Distortion (normal) (1KHz, + 2.0 dBv)	0.2%
Output level deviation	+ /- 1.2dB (1KHz)
L/R level deviation	+ /- 0.5dB (1KHz)
Frequency deviation	+ /- 0.3%
Zout	1K ohm (1KHz)

Stereo Mini Jack

The stereo mini jack is used for stereo headphone
output. The audio L/R outputs are not shut
down when a headphone plug is inserted.

Characteristics	Value
Zout	56 ohm
Output power (1KHz, 0.4%, 125Ω)	6.6 mW
Recommended headphone impedence	125 ohm

Note: Specifications are subject to change
without notice.

Index

R

Notes:

Tech Ref   SA23 - 1035
Tutorial   GK20 - 2252
Flyer   G58C - 0802
Pocket SW ref   G580 - 0803

Notes:

Notes: