Hu7 CD SYSTEM

TABLE OF CONTENTS

Part 1 HARDWARE

Chapter 1 Outline 6
1.1 Hu7 CD System Contents 6

Chapter 2 Hardware Configuration 7
2.1 System Hardware Configuration 7
2.2 System Hardware Connection Method 8
2.3 Individual Hardware 9
2.3.1 HDD unit 9
2.3.2 Expansion I/O box 9
2.3.3 Linear PCM board for PC-98 9
2.3.4 ADPCM + Parallel I/O board for PC-98 9
2.3.5 CD-ROM2 Interface board for Hu7 9
2.3.6 SCSI Interface board for PC-98 9
2.4 Illustration for Hardware Connection 10

Part 2 PSG DRIVER

Chapter 1 Outline 12

Chapter 2 PSG Driver Function Call 13
2.1 PSG Driver ON (PSG_ON) 13
2.2 PSG Driver OFF (PSG_OFF) 13
2.3 Initialize PSG (PSG_INIT) 13
2.4 Register Sound Data Bank No. (PSG_BANK).. 14
2.5 Register Track Data Index Address .... 14 (PSG_TRACK)
2.6 Register Wave Form Data Address .... 14 (PSG_WAVE)
2.7 Register Envelope Data Address (PSG_ENV). 14
2.8 Register Frequency Modulation Data Address (PSG_FM) 15
2.9 Register Pitch Envelope Data Address (PSG_PE) 15
2.10 Register Percussion Data Address (PSG_PC) 15
2.11 Set Tempo (PSG_TEMPO) 15
2.12 Play Track Data (PSG-PLAY) 16
2.13 Check Main Track (PSG_MSTAT) 16
2.14 Check Subtrack (PSG_SSTAT) 17
2.15 Stop Main Track Play (PSG_MSTOP) 17
2.16 Stop Subtrack Play (PSG_SSTOP) 18
2.17 Stop All Track Play (PSG_ASTOP) 18
2.18 Cut Volume of Main Track (PSG_MVOFF) 18
2.19 Continue Play (PSG_CONT) 19
2.20 Fade Out (PSG_FDOUT) 19
2.21 Set Main Track Delay Counter (PSG_DCNT).. 19

Chapter 3 Track Data 20
3.1 Register Track Data 20
3.2 Format Track Data 21

Time Base	: 2bytes	22
Octave	: 1byte	22
Octave Up	: 1byte	22
Octave Down	: 1byte	22
Tie	: 1byte	22
Tempo	: 2bytes	22
Volume	: 2bytes	23
Pan pot	: 2bytes	23
Sound Length Ratio	: 2bytes	23
Relative Volume	: 2bytes	23
Dal Segno	: 1byte	23
Segno	: 1byte	23
Repeat Begin	: 2bytes	24
Repeat End	: 1byte	24
Wave	: 2bytes	24
Envelope	: 2bytes	24
Frequency Modulation	: 2bytes	25
FM Delay	: 2bytes	25
FM Correction	: 2bytes	25
Pitch Envelope	: 2bytes	25
PE Delay	: 2bytes	25
Detune	: 2bytes	26
Sweep	: 2bytes	26
Sweep Time	: 2bytes	26
Jump	: 3bytes	26
Call	: 3bytes	26
Return	: 1byte	27
Transpose	: 2bytes	27
Relative Transpose	: 2bytes	27
Absolute Transpose	: 2bytes	27
Volume Change	: 2bytes	27
Pan Right Change	: 2bytes	28
Pan Left Change	: 2bytes	28
Mode	: 2bytes	29
Fade Out	: 2bytes	30
Data End	: 1byte	31

Chapter 4 Wave Form Data 31
User Defined Wave Form Data 31

Chapter 5 Envelope Data 32
User Defined Envelope Data 33

Chapter 6 Frequency Modulation Data 34
User Defined Frequency Modulation Data .. 34

Chapter 7 Pitch Envelope Data 35
User Defined Pitch Envelope Data 35

Chapter 8 Percussion Data 36
User Defined Percussion Data 37

HARDWARE

Chapter 1 OUTLINE

The Hu7 CD System consists of the hardware and software required to both develope software and create master tapes for the CD-ROM2 system. The Hu7 CD system makes it possible to edit linear PCM and ADPCM data, programs, and data for developing the CD-ROM. It is also possible to create a final master with this system.

1.1 Hu7 CD System Contents

The Hu7 CD system provides an environment for the development of:

. Record editing of linear PCM data
. Record editing of ADPCM data
. Developing programs by CD-ROM2 system emulation
. Registration editing of CD-ROM2 system data
. Creating 8mm MT master
. Data backup to 8mm MT

As shown above, the Hu7 CD system provides a total development environment for the CD-ROM2 system.

Chapter 2 HARDWARE CONFIGURATION

2.1 Hu7 CD System Hardware Configuration

The Hu7 CD system consists of the following hardware:

1) Hu7 CD Contents Hard Disk Unit, 8mm MT unit attached .... (1)
This is a 620MB hard disk.
Accessories: Power Cable .... (2)
SCSI Cable .... (3)

2) Expansion I/O Box .... (4)
(1) Hu7 CD CDPCM Board (Linear PCM Board) .... (5)
This board is to be installed in the PC-9801VX interface slot to play linear PCM recordings.
(2) Hu7 CD ADPCM Board .... (6)
This parallel I/O board is to be installed in the PC-9801VX interface slot to play ADPCM recordings.
Accessories: Parallel Cable .... (7)

3) SCSI Host Adapter Board .... (8)
This is a SCSI interface board to install in the PC-9801VX interface slot. It connects to a Hu7 CD contents hard disk unit when used.

Other than the above, the following hardware is required:

1) Hu7 System .... (9)
PC98 Parallel Interface .... (10)
Parallel Cable .... (11)

2) Interface Board for the Hu7 .... (12)
This interface unit for the CD-ROM2 system is to be installed in the top slot of the Hu7.

3) Display (which is able to input RF/video audio) .... (13)
Usage : The Hu7 system

4) PC-9801VX + Analog Display + Bus Mouse) .... (14)
Usage : The CD-ROM2 system emulation

5) PC-9801VX (Memory 640K + possibly VM with 16 color board)
+ Analog Display + Bus Mouse .... (15)
Usage : The Hu7 emulation

6) Audio System .... (16)
Usage : Audio check. It would work well with a mixer.

2.2 Hu7 CD System Hardware Connection Method

Now, we are going to explain the connection used by the Hu7 CD system. Numbers listed on the previous page are used. Follow these connection procedures.

1. Connect I/O box (4) to PC-9801 (14)

Digital Connection

2. Install SCSI board (8) to PC-9801 (14)
3. Connect SCSI board (8) and Hu CD contents hard disk (1) with a SCSI cable (3)
4. Install ADPCM board (6) onto the I/O box (4)
5. Install CD system board (12) onto the Hu7 system (9)
6. Connect ADPCM board (6) and CD system board (12) with a parallel cable
7. Install Parallel I/O board (10) on PC-9801 (15)
8. Connect Hu7 system and PC-9801 (15) with a parallel cable (11)

Analog Connection

9. Install CDPCM board (5) onto the I/O box (4)
10. Connect audio "OUT" (16) and CDPCM "IN" (5) with an audio cable
11. Connect CDPCM board "OUT" (5) and CD system board "IN" (12) with an audio cable
12. Connect ADPCM board "IN" (6) and audio "OUT" (16) with an audio cable
13. Connect ADPCM board "OUT" (6) and audio "IN" (16) with an audio cable

Video Connection

14. Connect CD system board "Audio OUT" (12) and display (13) with a video cable

2.3 Individual Hardware

2.3.1 Hu7 CD contents hard disk unit

The Hu7 CD contents hard disk unit is a hard disk with 620MB capacity and performs emulation for the CD-ROM2 system media and drive. Other than emulating the CD-ROM2 using the provided tools, the Hu7 CD contents hard disk unit functions as a MS-DOS hard disk. A 8mm MT accessory is provided with the Hu7 CD contents hard disk unit and it can be used to create a backup master by using software. This Hu7 CD contents hard disk unit also has a IN/OUT bus for SCSI.

2.3.2 Expansion I/O box

This should be installed on the PC-9801VX. The CDPCM board and ADPCM board are installed on this expansion I/O box.

2.3.3 Hu7 CD CDPCM board

This should be installed in the interface slot of the PC-9801VX. It can playback linear PCM recordings using the provided tools. This board has stereo IN/OUT pin jacks. Connecting an audio system to these pins enables creating linear PCM data.

2.3.4 Hu7 ADPCM board

This should be installed in the interface slot of the PC-9801VX. It can playback ADPCM recordings using the provided tools. This board has monaural IN/OUT pin jacks. Connecting an audio system to these pins enables creating ADPCM data. This interface with parallel I/O also transfers data when emulating CD-ROM2.

2.3.5 CD system board for Hu7 (Optional)

This should be installed on the top slot of the Hu7. This board has an equivalent function to the CD-ROM2 interface unit IFU-30 and allows CD audio stereo output.

2.3.6 SCSI host adaptor board

This should be installed in the interface slot of the PC-9801VX. This board has a SCSI bus and it should be connected to the Hu7 CD contents hard disk unit.

2.4 Illustration for Hardware Connection

PSG DRIVER

Chapter 1 OUTLINE

This PSG driver contains 12 virtual channels (called tracks) internally ; 6 channels for music (called the main track) and 6 channels for effect (called subtracks), and these are individually controlled. The each track can be separated when necessary, and the work area of the separated track become free.

PSG driver control is made possible with 21 function calls.

With 42 music data commands (internal command level), envelop, vibrato etc., various music data can be created. As the data command levels for the main track and the subtrack are similar, music play of a subtrack is possible.

The PSG driver contains 45 wave form data internally. User-created wave form data and envelop data can be registered outside of the driver.

The PSG driver can be called by either the timer or an IRQ. If the driver is called, the bank where sound data is stored will be spread into a logical address space (2 banks from address $8000), but the user can specify the bank number.

Chapter 2 PSG DRIVER FUNCTION CALL

Note: If following function call is executed, contents of ACC (accumulator) will be destroyed.

2.1 PSG Driver ON (PSG_ON)

_DH=0

Call	_AL = 0	Timer Call
	1	IRQ Call
Return	None	
Description	Enables the PSG driver. Selects with _AL calling the PSG driver by the timer or by IRQ. If the timer is selected, the timer is set to ON with this function call.	

2.2 PSG Driver OFF (PSG_OFF)

_DH=1

Call None

Return None

Description Stops the use of the PSG driver. If timer is selected, it stops the timer operation.

2.3 Initialize PSG (PSG_INIT)

_DH=2

Call _AL = PSG driver system number (0 ~ 5)

Return None

Description Initializes the PSG internal register and work area,then sets the system. The system numbers will set the system as follows:

0 : Enables main track only. Work on sub track side is set free. Stops functions related to subtrack. 1/60 second interval is used for the timer.
1 : Enables subtrack only. Work on main track side is set free. Stops functions related to main track. 1/60 second interval is used for the timer.
2 : Uses both main and sub tracks. 1/60 second interval is used for the timer.
3 : Uses both main and sub tracks. 1/120 second interval is used for the timer.
4 : Uses both main and sub tracks. 1/240 second interval is used for the timer.
5 : Uses both main and sub tracks. 1/300 second interval is used for the timer.

Timer interval settings by system number only effects the subtrack. Main track maintains 1/60 second speed by delay counter. In case of IRQ call interval settings will be ignored.

2.4 Register Sound Data Bank Number (PSG_BANK)

_DH=3

Call _AL = Bank Number 0
_AH = Bank Number 1

Return None

Description Specifies the bank number where sound data is stored. If an interrupt occurs, AL value and AH value will be set to mapping registers MPR4 and MPR5, respectively, and data will be spread into logical address space.

2.5 Register Track Data Index Address (PSG_TRACK)

_DH=4

Call _AX = Top Index Address of Track

Return None

Description Registers the top index address of the track data. (See Chapter 3 "Track Data" for track data format)

2.6 Register Wave Form Data Address (PSG_WAVE)

_DH=5

Call _AX = Wave Form Data Top Address

Return None

Description Registers the top address of user defined wave form data. (See Chapter 5 "Wave Form Data" for wave form data format)

2.7 Register Envelope Data Address (PSG_ENV)

_DH=6

Call _AX = Envelope Data Top Index Address

Return None

Description Registers the top index address of user defined envelope data. (See Chapter 5 "Envelope Data" for envelope data format)

2.8 Register Frequency Modulation Data Address (PSG_FM)

	_DH=7
Call	_AX = Frequency Modulation Data Top Index Address
Return	None
Description	Registers the top index address of user defined frequency modulation data. (See Chapter 6 "Frequency Modulation Data" for frequency modulation data format)

2.9 Register Pitch Envelope Data Address (PSG_PE)

	_DH=8
Call	_AX = Pitch Envelope Data Top Index Address
Return	None
Description	Registers the top index address of user defined pitch envelope data. (See Chapter 7 "Pitch Envelope Data" for pitch envelope data format)

2.10 Register Percussion Data Address (PSG_PC)

	_DH=9
Call	_AX = Percussion Data Top Index Address
Return	None
Description	Registers the top index address of user defined percussion data. (See Chapter 8 "Percussion Data" for percussion data format)

2.11 Set Tempo (PSG_TEMPO)

	_DH=10
Call	_AL = Tempo Counter (35 ~ 255)
Return	None
Description	Sets play speed by changing the timer interrupt. If less than 35 is set, it will be corrected to 35. When a sub track is used, try not to change the tempo.

2.12 Play Track Data (PSG_PLAY)

DL=11

Call _AL = Sound Number (0 ~ 127)
_AH = Wave Form Number (effective only at debug mode)

Return None

Description. Plays the track data registered at the specified number. At debug mode, _AH contents will be registered internally as a wave form number. (All tracks to be played are subjected to the change, but if there is specified wave number in the track data, it will be changed to that number.)

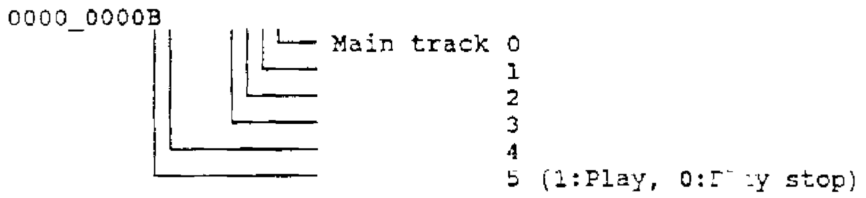
Note: Do not specify number for which track data is not specified.

2.13 Check Main Track (PSG_MSTAT)

_DH=12

Call None

Return ACC = Check Information



Description Checks the presently playing main track and returns bit information. If a main track is playing, the main track corresponding bit will be set to 1 and the value will be returned. If it is not used, a value $80 will be returned.

2.14 Check Subtrack (PSG_SSTAT)

_DH=13

Call   None

Return   ACC = Check Information

```
0000_0000B
|   ||||
|   |||└── Subtrack 0
|   ||└───          1
|   |└────          2
|   └─────          3
└─────────          4
└─────────          5(1:Play, 0:Play Stop)
```

Description Searches the playing subtrack and returns the information as bit information. If it is playing, the corresponding bit to the subtrack is set to 1 and will be returned. If a subtrack is not used, the value $80 will be returned.

2.15 Stop Main Track Play (PSG_MSTOP)

_DH=14

Call   _AL = Specifies Stop Track

```
0000_0000B
|||  ||||
|||  |||└── Main Track 0
|||  ||└───            1
|||  |└────            2
|||  └─────            3
||└────────            4
|└─────────        5 (1:Play Stop, 0:Play Continue)
└────────── Pause Switch   (1:Pause ON)
```

Return   None

Description Stops main track play. Specifies stop track with least significant bit 6. If bit 7 is specified, entire main track will be temporarily stopped. (In this case, track specification is ignored.)

2.16 Stop Subtrack Play (PSG_SSTOP)

_DH=15

Call _AL = Specifies Stop Track

```
0000_0000B
|  |  | ||└─ Subtrack 0
|  |  | |└──          1
|  |  | └───          2
|  |  └─────          3
|  └────────          4
|  ───────      5 (1:Play Stop, 0:Play Continue)
└─────────── Pause Switch   (1:Pause ON)
```

Return None

..ption Stops subtrack play. Specifies stop track with least significant bit 6. If bit 7 is specified, entire subtrack will be temporarily stopped. (In this case, tra[illegible] specification is ignored.)

2.17 Stop All Track Play (PSG_ASTOP)

_DH=16

Call None

Return None

Description Stops entire track play

2.18 Cut Volume of Main Track (PSG_MVOFF)

_DH=17

Call _AL = Specifies Volume Cut Track

```
0000_0000B
||    ||└─ Main Track 0
||    |└──            1
||    └───            2
|└────────            3
|─────────            4
└─────────            5
          (1:Volume OFF, 0: Play Continue)
```

Return None

Description Temporarily turns off the presently playing main track volume. Volume can be returned later.

2.19 Continue Play (PSG_CONT)

	_DH = 18
Call	_AL = 0 : Restart Main track 1 : Restart Subtrack 2 : Restart All Track
Return	None
Description	Restarts playing of the track presently stopped or whose volume is off.

2.20 Fade Out (PSG_FDOUT)

	_DH = 19
Call	_AL = Fade Out Speed (1 ~ 127)
Return	None
Description	Executes a main track fade out. Fade out speed increases as the value increases. If the specified value is negative, it will be corrected to positive.

2.21 Set Main Track Delay Counter (PSG_DCNT)

	_DH = 20
Call	_AL = Delay Counter Value (0 ~ 7)
Return	None
Description	Sets the delay counter. Play speed of the main track slows down as the value increases. Play speed is 1/1 at 0, 1/2 at 1, 1/3 at 2, ... 1/8 at 7. Note: As the delay counter value is set by PSG initialization, this function is usually not used. Set value to 1 when playing with a tempo change.

Chapter 3 TRACK DATA

3.1 Register Track Data

Registers the top address of data created for each track as shown in the format below and distributes the address to each channel.

```
TRK_INDEX:                     ... Track data index table
      DW      TEST0
      DW      TEST1
              .
              .
              .
;
TEST0:                         ... Track data top address register
table
      DB      0011_1111B
      DW      PART_1           ... Track data top address
      DW      PART_2
      DW      PART_3
      DW      PART_4
      DW      PART_5
      DW      PART_6
;
TEST1:
              .
              .
              .
```

Registers the top address of the track data top address register by using the track data index table. The top address of the index table (TRK_INDEX) is registered to the PSG driver by a function call. In this way, by giving the sound number by a function call, PSG driver will assign the track top address to each channel by the track data top address registration table and will start playing.

The track data top address register table sets how to assign each track data to a channel and if it is registered as a main track or a subtrack. The top 1 byte is a switch and bits 0~5 are switches for channels 1~6. By setting each bit to 1, that channel becomes ON and the track data top address will be assigned from the most recent number within the ON channels.

```
;
TEST1:

      DB      0011_0011B
      DW      PART_A
      DW      PART_B
      DW      PART_C
      DW      PART_D
;
```

In example on left, channel 1,2 and 5,6 are assigned to PART_A,B and PART_C,D respectively. Channel 3,4 becomes OFF.

Bit 7 is an interchange switch for main and sub tracks, and if it is 0, it will be registered as the main track. Bit 6 is a debug switch, and if it is 1, it will be in debug mode.

3.2 Format Track Data

The music track data format used by the PSG driver is shown below. These formats are common for both main and sub tracks except in one area. This driver uses two data creation methods: One is the Tempo method where tempo is set by a change of timer interval. The other is the Length method where play speed is set by adjusting the sound length. With the Tempo method, a change in tempo changes the play speed of the subtrack (effect) as well, therefore, create data using the Length method when using a subtrack.

INTERVAL

2 bytes(Interval code, Sound length)	(Direct length mode)
1 byte (Interval code + Sound length)	(Time base length mode)

Code	Do	: $10	Sol	: $80
	Do#	: $20	Sol#	: $90
	Re	: $30	La	: $A0
	Re#	: $40	La#	: $B0
	Mi	: $50	Ti	: $C0
	Fa	: $60		
	Fa#	: $70	Rest	: $00

Sound length 1 ~ 255	(Direct length mode)
0 ~ 15	(Time base length mode)

Specifies interval. If code is $00 it becomes a rest. There are 2 modes, direct length and time base length, to specify the sound length.

Direct length mode :	Directly specifies a sound length between 1~255. It is convenient to specify a sound length directly or to create tempo method data, except the data format is 2 bytes long. The least significant 4 bits in the interval code are ignored.
Time base length mode :	Specifies the sound length between 0~15. The specified value is corrected to 1~16 inside the driver. It is convenient to create length method data. Actual length is time bass value times sound length value, which is 1~240. The data format is the most significant 4 bits make up the interval code and the least significant 4 bits make up the sound length data.

Length mode can be interchanged with time bass value.

For tempo method data, the sound length equals 192 divided by the note length. (Quarter note : 192/4=48) At this time, set the delay counter value to 1 with a function call.

For length method data, do not specify the tempo.

TIME BASE	: 2 bytes (Time base code, Time base value)
Code	$D0
Time base value	0 ~ 15

Specifies the time base value. If value is 0, length mode will become direct mode. If value is other than 0, it will change to time base mode. Default value is 0 (direct length mode).

OCTAVE	: 1 byte (Octave code)
Code	$D1 ~ $D7

Specifies octave. Octave 1~7 corresponds to code $D1 ~ $D7. If the octave specification is omitted, octave 4 will be specified.

OCTAVE UP	: 1 byte (Octave up code)
Code	$D8

Elevates one octave up.

OCTAVE DOWN	: 1 byte (Octave down code)
Code	$D9

Drops one octave down.

TIE	: 1 byte (Tie code)
Code	$DA

Links front and rear sound together.

TEMPO	: 2 bytes (Tempo code, Tempo value)
Code	$DB
Tempo value	35 ~ 255

Specifies tempo. If multiple channels are specified, the channel with the most recent number has priority.

* Subtrack does not have this command.

VOLUME	: 2 bytes (Volume code, Volume value)
Code	$DC
Volume value	0 ~ 31

Specifies volume. Default value is 31.

PAN POT	: 2 bytes (Pan pot code, RL value)
Code	$DD
RL value	$00 ~ $FF

Specifies right and left volume. In the specified value upper is for the left and lower is for the right value. Note: A default value is not defined, so use of this command is necessary.

SOUND LENGTH RATIO : 2 bytes (Sound length ratio, Sound length ratio value)

Code	$DE
Sound length ratio value	1 ~ 8

Specifies the sound generation ratio within 1 sound. Ratio between 1/8 ~ 8/8 can be specified by a value 1~ 8. Default value is 8.

RELATIVE VOLUME : 2 bytes (Relative volume code, Relative value)

Code	$DF
Relative value	-31 ~ 31

Specifies the volume relatively to the present volume value. Aelative value of 1 raises one volume and -1 brings down one volume.

DAL SEGNO	: 1 byte (Dal Segno code)
Code	$E1

Returns to the specified location by Segno and repeats play. If there is no segno specified, it will go back to the top of the data. (Da Capo)

SEGNO	: 1 byte (Segno code)
Code	$E2

Specifies the data location where play will be repeated after using Dal Segno.

REPEAT BEGIN	: 2 bytes (Repeat begin code, Loop number)
Code	$E3
Loop number	1 ~ 255

Specifies the beginning point of repeated play and the number of times. If the number of times specified is 0, it will be corrected to 2.

REPEAT END	: 1 byte (Repeat end code)
Code	$E4

Specifies the ending point of repeated play.

- Repeat begin and repeat end can be nested. One round trip uses 3 bytes of user stack area secured inside. The user stack area secures 1 channel of 12 bytes, therefore, a maximum of 4 levels of nests is possible. More than 4 nests will result in stack overflow, so be careful.
- User stack size for subtracks is 9 bytes.

WAVE	: 2 bytes (Wave code, Wave number)
Code	$E5
Wave number	0 ~ 127

Specifies a wave form number. Wave form numbers 0~44 are for PSG driver internal definition and 45~127 are for user definition. A default is not defined.

Note: Do not specify a number which does not define a wave form.

ENVELOPE	: 2 bytes (Envelope code, Envelope number)
Code	$E6
Envelope number	0 ~ 127

Specifies an envelope number. 0~15 are for PSG driver internal definition numbers and 16~127 are for user definition numbers. Default value is 0.

Note: Do not specify a number which does not define an envelope.

FREQUENCY MODULATION (FM) : 2 bytes (FM code, FM number)

Code	$E7
FM number	0 ~ 127

Specifies a frequency modulation data number.

FM DELAY	: 2 bytes (FM delay code, Delay value)
Code	$E8
Delay value	0 ~ 255

Specifies a frequency modulation delay time. Delay value is equivalent to sound length. Default value is 0.

Note: If you have not defined frequency modulation data, set the delay value to 0.

FM CORRECTION	: 2 bytes (FM correction code, Standard octave)
Code	$E9
Standard octave	0 ~ 7

Corrects frequency modulation. Standard octave is an octave that has no correcting modulation data. Correction is done by moving an octave up or down. If you specify a standard octave of 0, correction has no effect. Default value is 0. Correction will work on pitch envelope, detune, and sweep, simultaneously.

Note: There is a period where modulation has no effect if an octave is raised too high with correction.

PITCH ENVELOPE	: 2 bytes (PE code, PE number)
Code	$EA
PE number	0 ~ 127

Specifies a pitch envelope number.

PE DELAY	: 2 bytes (PE delay code, Delay value)
Code	$EB
Delay value	0 ~ 255

Specifies delay time for the pitch envelope. Delay value is equivalent to sound length. Default value is 0.

Note: If you have not defined pitch envelope data, set the delay value to 0.

DETUNE	: 2 bytes (Detune code, Detune value)
Code	$EC
Detune value	-128 ~ 127

Fine tunes sound interval. Default value is 0.

SWEEP	: 2 bytes (Sweep code, Change value)
Code	$ED
Change value	-128 ~ 127

Specifies change value for sweep. The larger the number is, the faster the change becomes. If the change value is positive, the interval goes down. If negative, interval goes up. If 0 is specified, sweep is set to OFF. Default value is 0.

SWEEP TIME	: 2 bytes (Sweep time code, Time value)
Code	$EE
Time value	0 ~ 255

Specifies time for sweep. If 0 is specified, time becomes invalid and will remain in effect until sweep becomes key OFF. Default value is 0.

JUMP	: 3 bytes (Jump code, Address)
Code	$EF
Address	Lower address, Upper address

Jump to track data indicated by the address.

CALL	: 3 bytes (Call code, Address)
Code	$F0
Address	Lower address, Upper address

Calls track data at the address indicated. A return command returns stream back to the data following the call.

RETURN	: 1 byte (Return code)
Code	$F1

Returns to the data following the call.

- A call and return could be nested. One call uses 2 bytes of user stack. Since the user stack area has 12 bytes, a maximum of 6 levels of nests is possible. Use condition of the user stack needs to be taken into consideration since it is commonly used by the repeat command.
- User stack size for subtracks is 9 bytes.

TRANSPOSE	: 2 bytes (Transpose code, Transpose value)
Code	$F2
Transpose value	-128 ~ 127

Transposes sound interval. A transpose value unit is a half step, and if it is positive, the interval elevates. Default value is 0.

RELATIVE TRANSPOSE : 2 bytes (Relative transpose code, Relative transpose value)

Code	$F3
Relative transpose value	-128 ~ 127

Relatively transposes from the present transpose value.

ABSOLUTE TRANSPOSE : 2 bytes (Complete transpose code, Transpose value)

Code	$F4
Transpose value	-128 ~ 127

Transposes all channels. Send this command to the channel with the least recent number.

Note: If transposing an upper area or lower area octave, it might go out of sound range.

VOLUME CHANGE : 2 bytes (V change code, Change quantity)

Code	$F5
Change quantity	-128 ~ 127

Gradually changes the volume. If the change quantity is positive, the volume will increase and if negative, it will decrease. If 0 is specified, change will be stopped sustaining the present volume. If volume is set while changing, the change will be canceled. Default value is 0.

PAN RIGHT CHANGE : 2 bytes (PR change code, Change quantity)

Code	$F6
Change quantity	-128 ~ 127

Gradually changes right volume. If change quantity is positive, volume will increase and if negative, it will decrease. If 0 is specified, change will be stopped sustaining the present volume. If pan pot is set while changing, the change will be canceled. Default value is 0.

PAN LEFT CHANGE : 2 bytes (PL change code, Change quantity)

Code	$F7
Change quantity	-128 ~ 127

Gradually changes left volume. If change quantity is positive, volume will increase and if negative, it will decrease. If 0 is specified, change will be stopped sustaining the present volume. If pan pot is set while changing, the change will be canceled. Default value is 0.

MODE	: 2 bytes (Mode code, Mode number)
Code	$F8
Mode number	0 ~ 2

Specifies music play mode. 0 is normal (sound interval) mode, 1 is percussion mode, and 2 is noise mode. 2 is only valid for channel numbers 5 and 6 as the other channels cannot generate this sound correctly. Sound also cannot be generated correctly for mode 1 on channels other than 5 and 6 if noise is contained. Default value is 0.

Note: If mode 2 is specified, interval code $10 becomes noise number 0 and $C0 becomes number 11. There are noise numbers up to 31. Specify noise numbers more than 12 by using transpose.

- Specify rest at percussion mode (mode 1) using one of the following methods:

1. Specify top rest at normal mode (mode 0)

```
PC_PRT:
      DB    $D0,$0F            ; TIME BASE=15
      DB    $DC,$1F            ; VOL=31
      DB    $DD,$EE            ; PAN=$EE
      DB    $F8,$01            ; MODE=1
      DB    $01          ; R
      DB    $01          ; R
      DB    $81          ; G
      DB    $01          ; R
      DB    $81          ; G
      DB    $FF          ; DATA END
```

```
PC_PRT:
      DB    $D0,$0F            ; TIME BASE=15
      DB    $DC,$1F            ; VOL=31
      DB    $DD,$EE            ; PAN=$EE
      DB    $F8,$00            ; MODE=0
      DB    $01          ; R
      DB    $01          ; R
      DB    $F8,$01            ; MODE=1
      DB    $81          ; G
      DB    $01          ; R
      DB    $81          ; G
      DB    $FF          ; DATA END
```

2. Create rest data inside percussion data table

```
--------PERCUSSION DATA TABLE-----------------

PC_INDEX_ADR:
     DW   PC0        ; C
     DW   PC1        ; C+
     DW   PC2        ; D
          :
;
PC0: DB   $F0        ; DATA END   ...  Create this kind of data
;

--------MUSIC DATA----------------------------

PC_PRT:
     DB   $D0,$0F         ; TIME BASE=15
     DB   $DC,$1F         ; VOL=31
     DB   $DD,$EE         ; PAN=$EE
     DB   $F8,$01         ; MODE=1
     DB   $01        ; R
     DB   $01        ; R
     DB   $81        ; G
     DB   $01        ; R
     DB   $81        ; G
     DB   $FF        ; DATA END


PC_PRT:
     DB   $D0,$0F         ; TIME BASE=15
     DB   $DC,$1F         ; VOL=31
     DB   $DD,$EE         ; PAN=$EE
     DB   $F8,$01         ; MODE=1
     DB   $11        ; C (R)
     DB   $11        ; C (R)
     DB   $81        ; G
     DB   $11        ; C (R)
     DB   $81        ; G
     DB   $FF        ; DATA END
---------------------------------------------------
```

FADE OUT	: 2 bytes (Fade out code, Speed)
Code	$FE
Speed	1 ~ 127

Performs fade out. The larger the value is, the faster fade out speed is. If the specified value is negative, it will be corrected to positive.

DATA END	: 1 byte (Data end code)
Code	$FF

Indicates end of track data. Always put this code at a play stop location.

Chapter 4 WAVE FORM DATA

The PSG driver defines 45 kinds of wave form data internally but user created wave form data can be defined outside with a function call. The user can define 83 kinds of wave forms. Wave form numbers 0~44 are for internal definition and 45~127 are for user definition.

The wave form data format is 32 bytes of wave form data arranged in memory in order. The top addresses of the data are registered to the PSG driver through a function call.

------User defined wave form data--------------------

```
WAVE_TOP:
        DB    $00,$00,$00,$00,$00,$00,$00,$00        ; NO.45
        DB    $1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F
        DB    $00,$00,$00,$00,$00,$00,$00,$00
        DB    $1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F
;
        DB    $00,$01,$02,$03,$04,$05,$06,$07        ; NO.46
        DB    $08,$09,$0A,$0B,$0C,$0D,$0E,$0F
        DB    $10,$11,$12,$13,$14,$15,$16,$17
        DB    $18,$19,$1A,$1B,$1C,$1D,$1E,$1F
;
        DB    $00,$08,$0F,$14,$19,$1B,$1D,$1E        ; NO.47
        DB    $1E,$1D,$1B,$19,$14,$0F,$08,$00
        DB    $1F,$1E,$1D,$1C,$1B,$1A,$09,$08
        DB    $07,$06,$05,$04,$03,$02,$01,$00
;
```

Chapter 5 ENVELOPE DATA

In this PSG driver, defined envelope data is not related to wave form data, so the user can select it by number. Envelope data contains 16 definitions inside the driver, but the user can define 112 kinds outside the driver by a function call. Envelope number 0~ 15 are for internal definition and 16~ 127 are for user definition.

Envelope data consists of an index and data parts. At registration, register the top address of the index to the PSG driver with a function call.

Envelope data format is as follows:

RELEASE RATE DATA : 3 bytes (Code, Level change quantity)

Code	$FB
Change quantity	-$7C00 ~ $7C00

Specifies the release rate (change speed after key OFF until level becomes 0). If change quantity is negative, level decreases and if positive, level increases and it will change until the maximum levelis reached.

LEVEL DATA	: 3 bytes (Code, Level set value)
Code	$FC
Set quantity	0 ~ $7C00

Set the initial level for time of level change.

DECAY RATE DATA : 3 bytes (Time, Level change quantity)

Time	0 ~ 250
Change quantity	-$7C00 ~ $7C00

Specifies the decay rate (level change speed during key ON). If change quantity is positive, level increases and if negative, it decreases. Time sets the time length for level to change during key ON. If time specified is 0, the level will keep changing during key ON. Time value is the equivalent of sound length.

DATA END	: 1 byte (Code)
Code	$FF

Indicates end of data. If this code is reached during key ON, sustain until the present level changes to key OFF. This does not apply if the decay time is 0.

Level change quantity value $400 is equal to level 1 and the maximum $7C00 is equal to level 31.

Place data in the order of number. (Put the release rate on top otherwise it will be regarded as an omission.)

Release, level, and decay data can be omitted, and data will be set to 0.

Release rate will be automatically set to 0 for envelope data used for percussion data. (Set release data will be ignored.)

Multiple level and decay data can be set. That will allow you to define complicated envelopes. (Data number for one envelope data is up to 85.)

If sound length ratio is 8, release will have no effect.

```
-------User defined envelope data-----------

ENV_INDEX:
        DW    ENV16
        DW    ENV17
                :
;
ENV16:
        DB    $FB          ; RELEASE RATE DATA
        DW    -$100
        DB    $FC          ; LEVEL DATA
        DW    31*$400
        DB    25           ;DECAY RATE DATA
        DW    -$180
        DB    $FF          ; DATA END
;
ENV17:
        DB    $FB
        DW    -$80
        DB    $FC
        DW    25*$400
        DB    3
        DW    $800
        DB    0
        DW    -$C0
        DB    $FF
;
```

Chapter 6 FREQUENCY MODULATION DATA

This frequency modulation data generates soft LFO for a sound interval and it can generate vibrato effects, etc. One modulation data table consists of one wave form cycle and various wave forms can be created using the data. One byte of data is an additional value for sound interval frequency number of sound length 1.

Frequency modulation data values are -127 ~ 127 and -128($80) indicates end of data. The modulation table length maximum is 256 bytes. Data configuration consists of index and data parts. At registration, register the top address of the index to the PSG driver with a function call.

———User defined frequency modulation data————

```
FM_INDEX:
      DW    FMD0
      DW    FMD1
      DW    FMD2

;
FMD0:
      DB    00,01,02,03,02,01,00,-1,-2,-3,-2,-1,$80
FMD1:
      DB    00,-2,-4,-6,-4,-2,00,02,04,06,04,02,$80
FMD2:
      DB    03,03,02,02,01,01,00,00,-1,-1,-2,-2,-3,-3,$80
```

Chapter 7 PITCH ENVELOPE DATA

Pitch envelope data is a data table used to partially change pitch (sound interval). Data configuration is the same as for frequency modulation data.

At registration, register the top address of index part with a function call.

———User defined pitch envelope data————

```
PE_INDEX:
        DW    PEG0
        DW    PEG1
        DW    PEG2
                :
;
PEG0:
        DB    -15,-12,-10,-8,-6,-5,-4,-3,-2,-1,00,$80
PEG1:
        DB    0,1,2,4,6,8,12,15,17,00,$80
PEG2:
        DB    -1,-3,-6,-10,-13,-14,-13,-10,-6,-3,-1,00,$80
```

Chapter 8 PERCUSSION DATA

This PSG driver creates percussion data with effective sound and plays the data using interval.

Data configuration is divided into index and data parts. At registration, register the top address of the index to the PSG driver with a function call.

12 addresses can be defined for the index part and sound interval code $10 ~ $C0 will be assigned from its first address. When generating sound from percussion data, define assigned interval code by track data so that the data will play. (When play mode 1) Data part defines percussion data using the following format. Sound will be generated in order of definition.

NOISE NUMBER

$00 ~ $1F

Defines noise number. This data becomes the length of sound length 1.

INTERVAL FREQUENCY NUMBER

$B0 + Upper frequency number data, Lower frequency number data

Defines the interval frequency number. This data becomes the length of sound length 1.

ENVELOPE NUMBER

$C0, 0 ~ 127

Specifies the envelope number. The release rate of envelope data specified here will be set to 0.

PAN POT

$D0, $00 ~ $FF

Specifies pan pot. Upper data is for the left and lower is for right specification.

WAVE FORM NUMBER

$E0, 0 ~ 127

Specifies the wave form number.

DATA END

$F0

Indicates end of data.

———User defined percussion data—————

```
PC_INDEX:
        DW      PCN0            ; C   ($10)
        DW      PCN1            ; C+  ($20)
        DW      PCN2            ; D   ($30)
        DW      PCN3            ; D+  ($40)
        DW      PCN4            ; E   ($50)
        DW      PCN5            ; F   ($60)
        DW      PCN6            ; F+  ($70)
        DW      PCN7            ; G   ($80)
        DW      PCN8            ; G+  ($90)
        DW      PCN9            ; A   ($A0)
        DW      PCN10           ; A+  ($B0)
        DW      PCN11           ; B   ($C0)
;
PCN0:
        DB      $C0,7           ; ENVELOPE NO.
        DB      $E0,6           ; WAVE NO.
        DB      $D0,$FF         ; PAN POT
        DB      $B0+3,$AC       ; TONE FRQ
        DB      $B0+4,$0C
        DB      $1F             ; NOISE FRQ
        DB      $1B
        DB      $1D
        DB      $1A
        DB      $F0             ; DATA END
;
PCN1:
        DB      $C0,21
        DB      $E0,6
        DB      $D0,$FF
        DB      $B0+4,$7C
        DB      $B0+4,$8C
        DB      $B0+4,$9C
        DB      $B0+4,$BC
        DB      $B0+4,$DC
        DB      $B0+4,$FC
        DB      $B0+5,$0C
        DB      $F0
;
                :
                :
```