ICS WaveFront Interface for Turtle Beach Systems Maui

Integrated Circuit Systems, Inc.
P.O. Box 968
Valley Forge, PA 19482-0968
Voice: (215) 630-5300
Fax: (215) 630-5399

Turtle Beach Systems, Inc.
52 Grumbacher Road
York, PA 17402
Voice: (717) 767-0200
Fax: (717) 767-6033

This document defines the register level interface to the Maui hardware. The registers available on the Maui are:

Base Address	330, 338, 320, 300, 290, 260, 230, 210
Base+0	MPU-401 Emulation Data Register
Base+1	MPU-401 Emulation Control/Status Register
Base+2	Host Data Register
Base+3	Host Control Status Register

Host Interface

The host interface provides for functions not generally associated with MIDI, such as controlling the mixing levels.

Host Control Register

The Host Control Register (write only) provides the host with access to several utility functions. The following table describes the function of the various bits in the register:

Bit	Function
0	Host Rx Interrupt Enable (1=Enabled)
1	Unused
2	Unused
3	Unused
4	Host Tx Interrupt Enable
5	Mute (0=Mute; 1=Play)
6	Master Interrupt Enable (1=Enabled)
7	Master Reset (0=Reset; 1=Run)

The **Host Rx Interrupt Enable** enables the host data receive interrupts. For example, the host processor will receive an interrupt when the Host Data Receive Register is full.

The **Host Tx Interrupt Enable** enables host data transmit interrupts. For example, the host processor will receive an interrupt when the Host Data Transmit Register is empty.

Upon power-up, the **Mute** bit is set to zero (0), which mutes the audio output. Unmute the board by setting the mute switch to one (1).

When set to zero (0), the **Master Interrupt Enable** disables all interrupts from the board. When set to one (1), any of the individual interrupts which have been enabled separately can generate interrupts. This affects the MIDI emulation interrupts as well.

When the **Master Reset** is set to zero (0), the audio board is held in reset, which is the power-up condition. Setting Master Reset to one (1) allows the on-board processor to run. It takes approximately two to four seconds, depending on the memory configuration, for the on-board processor to complete it's initialization routine before it will respond to commands after a reset. Following a Master Reset, the board must be reloaded with operating system microcode.

Host Status Register

The Host Status Register provides the host with status of the two Host Interface interrupt sources. The following table describes the function

of the various bits:

Bit	Function
0	Host Rx Interrupt Enable (1=Enabled)
1	Host Rx Register Full (1=Full)
2	Host Rx Interrupt Pending (1=Interrupt)
3	Unused
4	Host Tx Interrupt (1=Enabled)
5	Host Tx Register empty (1=Empty)
6	Host Tx Interrupt Pending (1=Interrupt)
7	Unused

Host Rx Interrupt Enable indicates the status of the Host Rx Interrupt enable bit in the Host Control Register.

Host Rx Register Full indicates when a data byte is available in the Host Receive Data Register.

Host Rx Interrupt Enable indicates that an interrupt is pending, when the Host Receive Data Register contains data and Host Rx interrupts are enabled.

Host Tx Interrupt Enable indicates the status of the Host Tx Interrupt enable in the Host Control Register.

Host Tx Register Empty indicates when the Host Transmit Data Register is empty.

Host Tx Interrupt Pending indicates that an interrupt is pending, when the Host Transmit Data Register is empty and Host Tx interrupts are enabled.

Interrupts

In addition to the individual interrupt controls, there is a **Master Interrupt Enable** bit in the Host Control Register which must be set to allow any interrupts from the board. Available IRQs are 2/9, 5, 12, and 15 and the driver software should allow for this. Note also that it is necessary to unmask the interrupt in the Programmable Interrupt Controller onboard the PC.

MIDI Interface

The registers for the MIDI interface are compatible with MPU-401 registers.

Message Support

The onboard synthesizer will appear to the software as an external MIDI-based synthesizer. Notes are played using standard MIDI Note-On and Note-Off messages. The following MIDI messages are supported:

Message	Function
8x	Note Off
9x	Note On
Bx	Controller Change
Cx	Program Change
Dx	Channel Pressure
Ex	Pitch Bend Change
F9	Switch to onboard synthesizer (Virtual MIDI Mode)
FD	Switch to external MIDI (Virtual MIDI Mode)

The following controllers are supported:

Controller	Function
1	Mod Wheel
2	Breath Controller
4	Foot Controller
7	Volume
10	Stereo Pan
11	Expression Controller
64	Sustain Pedal

With the exception of the Volume and Sustain pedal controller, the effect of the various controllers on the sounds produced will be determined by the individual patches.

SysEx Support

The operating system software for the ICS WaveFront supplies full access to synthesizer function via MIDI System Exclusive messages. This includes downloading/uploading of patch data, program data, sample header data, multisample data, sample alias data, and other synthesizer parameters and variables. It also allows the downloading of sample data via SysEx, however this is a very data intensive operation and requires substantial time to perform.

Virtual MIDI Interface

The MIDI interface supports a virtual mode which, coupled with appropriate driver software, provides the equivalent of a second MIDI interface. The board defaults to a mode whereby MIDI data is sent from the host to both the onboard synthesizer and the external MIDI interface. The board can be switched into Virtual MIDI Mode, via a Host command, where the two ports are treated separately. The MIDI stream from the host interface is switched between the onboard synthesizer and the external MIDI port by means of two MIDI messages.

Once the board is in Virtual MIDI Mode (Host Command A8h), all MIDI data will be sent to the onboard synthesizer until an FDh is received in the MIDI data. The FDh will not be processed by the synthesizer, and any new MIDI data received will be sent to the external MIDI interface. This process continues until an F9h is received in the MIDI data. The F9h will not be sent to the external MIDI interface, and any new MIDI data will be sent to the onboard host. At any time the host command of A9h will disable Virtual MIDI Mode and MIDI data from the host will again be routed to both the synthesizer and the external MIDI interface. Similarly, while in Virtual MIDI mode, the synthesizer operating system will also use these same two MIDI flags to indicate the source of MIDI data being input to the host. If MIDI data is received on the serial MIDI interface, the O.S. will input an FDh to the host prior to inputting the data. Likewise, any data being input from the O.S., as in response to System Exclusive operations, will precede this data with an F9h.

Host Interface

The ICS WaveFront functions listed in this document are accessible both through the SysEx and directly through the host port. To access the functions via SysEx, the string must be built as shown in this section. Through the host port, these functions can be executed directly without the SysEx header and F7 terminator. When using the host port, the commands must have their high bit set. This can be accomplished by adding 80h("h" throughout the document signifies the hex numerical system) to the commands.

The Host Interface provides access to many functions that aren't necessarily associated with the standard MIDI interface. The functions break down into two primary categories: Controls and Sample and Patch Download.

Messages to the board consist of a single command byte. If required the command will be followed by some number of data bytes. The command bytes will always have their high bit set to allow for resynchronization in case of an incomplete message caused by some

programming or system anomaly.

Data bytes will always have their high bit reset, limiting them to values from 0 to 7Fh. Due to this, some data values will need to be split across multiple bytes. When this occurs, the data is passed least significant byte (LSB) first. When the most significant byte (MSB) is less than seven bits wide, it will be right justified in the byte that it occupies. As an example, a value that has a range of 0-FFh will be passed as two bytes. The first byte will be the least significant seven bits of the value, the second byte will have the most significant bit in bit zero (LSB).
For this example, to convert an A3h from one (1) to two (2) MIDI SysEx bytes perform the following steps.

1. Convert the hex value to binery.

Hex	A	3
Binary	1001	0011

2. Copy the least significant seven (7) bytes into the least significant seven (7) bits of the first data byte.

Binary	0001	0011
Hex	1	3

3. Copy the most significant bit into the least significant bit of the second data byte.

Binary	0000	0001
Hex	0	1

4. Assemble the string.

13	01

When a value is signed, the sign bit always occupies the higher order transmitted bit. For example, when a 14 bit value is transmitted as two bytes, the sign bit will be in bit six (6) of the second (high order) byte. When the word is assembled, the sign bit will be in bit 13 of the word. The sign bit will usually be extended to bits 14 and 15 by the host before it is used. To send the signed value back to the board, the host need only truncate bits 14 and 15 before transmitting it.

In nearly all cases, messages to the board will be acknowledged by either an explicit acknowledge byte, or by some other response. Under all conditions, it only responds to direct requests from the host.

The acknowledge response is 80h. An error is reported by first sending an FFh, followed by a one byte error code. The current list of error codes includes:

Code	Error
01	Bad sample number
02	Out of sample memory
03	Bad patch number
04	Error in number of voices
06	Sample load already in progress
0B	No sample load request pending
0E	Bad MIDI channel number
10	Download Record Error

System Exclusive Messages

Follow this example for all function changes in the Controls and Sample and Patch Download sections:

The System Exclusive Message for Download Sample Alias would be:

F0 00 00 65 10 CH 03 nn nn ss ss aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd dd ee ee ee ff ff F7

The **F0** indicates the beginning of a System Exclusive message.

00 00 65 indicates the manufacturer number for Turtle Beach Systems.

10 indicates the product number for Turtle Beach Maui.

CH indicates the MIDI Channel Number.

03h is the Host Command byte for a Download Alias command.

nn nn is the Sample number (0-FFh).

ss ss is the Aliased sample number (0-FFh).

aa aa aa aa is the Offset to start of sample.

bb bb bb bb is the Offset to start of loop.

cc cc cc cc is the Offset to end of loop.

dd dd dd dd is the Offset to end of sample.

ee ee ee is the Sample Frequency Bias.

ff ff is the flag byte bits 7-0.

F7 is the EOX or End of System Exclusive message byte.

A similar scheme is used to respond to ICS WaveFront commands via MIDI System Exclusive messages. The System Exclusive message returned by the Wave-Board in response to a download Sample Alias command would be:

F0 00 00 65 10 CH 00 F7

00 (following CH) indicates “Acknowledge” from the Wave-Board.

For the error case:

F0 00 00 65 10 CH 7F xx F7

7F indicates an Error from the Wave-Board.

xx indicates the Error code. (See the ICS WaveFront Interface Specification for more details on this code.)

Note that all bytes between the System Exclusive status byte and the EOX (or the next status byte) must have a zero in the MS (most significant) bit.

In the cases where the commands are sent to request uploading of information. The information will not be returned as described for the ICS WaveFront command response, but will instead be returned in the exact System Exclusive format that can then be used to download data at a later time. For example, a request for patch data will result in the return of a System Exclusive message which is formatted as a Download Patch command with its associated data.

Control Functions

Control functions provide access to the mixing controls and global controls that modify the behavior of the board. The following is a list of messages that can be sent and their expected responses.

Command: **Set Synthesizer Volume**
Sets the synthesizer playback volume. This control

determines the overall volume of the synthesizer.
Message: F0 00 00 65 10 CH 09 vv F7
Return: F0 00 00 65 10 CH 00 F7
Variable: vv=0-127 (127=0 dB; -1/2 or 1/2 dB increments)

Command: **Get Synthesizer Volume**
Returns the current Synthesizer Volume Setting. This function is useful for Control Panel functions where the actual setting might have been modified by another program. If the Get Synthesizer Volume command is received via MIDI System Exclusive, the response will be formatted as the associated Set Synthesizer Volume command.
Message: F0 00 00 65 10 CH 12 F7
Return: F0 00 00 65 10 CH 09 vv F7
Variable: vv=0-127 (127=0 dB; -1/2 or 1/2 dB increments)

Command: **Set Number of Voices**
Sets the number of voices used for synthesis. The number of voices has a direct effect on the quality of the audio. The lower the number of voice, the higher the sample rate is for each individual voice. However, reducing the number of voices can have adverse effects such as voice-stealing, which occurs when there are more notes being played than there are voices to play them. The Set Number of Voices function results in the computation of numerous frequency and timer related tables. Therefore, allow adequate time for its completion and the return of the associated Acknowledge.
Message: F0 00 00 65 10 CH 0B nn F7
Return: F0 00 00 65 10 CH 00 F7
Variable: nn=24-32

Command: **Get Number of Voices**
Returns the number of voices currently allocated for synthesis. See Set Number of Voices for a complete discussion. If the Get Number of Voices command is received via MIDI System Exclusive, the response will be formatted as the associated Set Number of Voices command.
Message: F0 00 00 65 10 CH 14 F7
Return: F0 00 00 65 10 CH 0B nn F7
Variable: nn=24-32

Command: **Set Synthesizer Tuning**
Sets the master tuning of the synthesizer, which is useful for tuning the synthesizer to other instruments. The value is relative to A-440 in increments of 1/2048 of an octave (a semitone is 2048/12 or approximately 171/2048 of an octave). Because the ICS WaveFront only resolves pitch down to approximately 1/512 of an octave, the lower two (2) bits are dropped just before output to the chip. It is recommended that the user interface be limited to +/- one-half semitone, which should be more than sufficient.
Message: F0 00 00 65 10 CH 26 nn nn F7
Return: F0 00 00 65 10 CH 00 F7
Variable: nn nn=-8192 to 8191

Command: **Get Synthesizer Tuning**
Returns the current synthesizer tuning. This function is useful for Control Panel applications where the actual value may have been modified by other programs. If the Get Synthesizer Tuning command is received via MIDI System Exclusive, the response will be formatted as the associated Set Synthesizer Tuning command.
Message: F0 00 00 65 10 CH 27 F7
Return: F0 00 00 65 10 CH 27 nn nn F7
Variable: nn nn=-8192 to 8191

Command: **Disable Synth Channel**
Disables the specified synthesizer MIDI channel. Messages to the synthesizer will no longer be accepted on the specified MIDI channel. Make sure that all pending notes are turned off using NOTE-OFF messages before issuing thismessage or the notes will be stuck on. By default all synthesizer channels are enabled.
Message: F0 00 00 65 10 CH 1A nn F7
Return: F0 00 00 65 10 CH 00 F7
Variable: nn=MIDI Channel Number (0-15)

Command: **Enable Synth Channel**
Enables the specified synthesizer channel. Messages to the synthesizer will again be accepted on the specified channel.
Message: F0 00 00 65 10 CH 1B nn F7
Return: F0 00 00 65 10 CH 00 F7

Variable: nn=MIDI Channel Number (0-15)

Command: **Get Synth Channel Status**
Reports the status of the synthesizer channels. The bits 0 to 6 are in the first byte, bits 7 to 13 in the second byte, and bits 14 to 15 are in the last byte. If the bit is set, the associated synthesizer channel will respond to MIDI messages. This message is useful for restoring context when exiting a program that may have manipulated the synthesizer enables.
Message: F0 00 00 65 10 CH 2B F7
Return: F0 00 00 65 10 CH ss ss ss F7
Variable: Synth Channel Status
Bit 0=Channel 0 Enabled (LSB of the first byte received.)
Bit 1=Channel 1 Enabled
Bit 2=Channel 2 Enabled
...
Bit 15=Channel 15 Enabled

Command: **Disable MIDI-In to Synth**
Disables the connection from the MIDI-In connector to the onboard synthesizer. This is useful if the host driver software is performing its own loop back to the synthesizer. Make sure that all pending notes are turned off using NOTE-OFF messages before issuing this message, or the notes will be stuck on. By default the path to the synthesizer is enabled.
Message: F0 00 00 65 10 CH 1D F7
Return: F0 00 00 65 10 CH 00 F7

Command: **Enable MIDI-In to Synth**
Enables the connection from the MIDI-In connector to the onboard synthesizer.
Message: F0 00 00 65 10 CH 1E F7
Return: F0 00 00 65 10 CH 00 F7

Command: **Enable Virtual MIDI Mode**
Enables Virtual MIDI Mode. See Virtual MIDI Mode in the MIDI Interface section of this document.
Message: F0 00 00 65 10 CH 28 F7
Return: F0 00 00 65 10 CH 00 F7

Command: **Disable Virtual MIDI Mode**
Disables Virtual MIDI Mode. See Virtual MIDI Mode in the MIDI Interface section of this document.
Message: F0 00 00 65 10 CH 29 F7
Return: F0 00 00 65 10 CH 00 F7

Command: **Report MIDI Status**
Reports the current MIDI status. If Bit 0 is set, the Virtual MIDI Mode is currently enabled. If Bit 1 is reset, the MIDI output will go to the onboard synthesizer, if set, it will go to the external MIDI interface. If bit two (2) is set, the connection from the MIDI input port to the onboard synthesizer is disabled.
Message: F0 00 00 65 10 CH 2A F7
Return: F0 00 00 65 10 CH ss F7
Variable: ss=MIDI Status
Bit 0=Vitrtual MIDI Mode (1=Enabled)
Bit 1=Virtual MIDI switch (0=Synth; 1=External)
Bit 2=MIDI-In to Synth Mode (1=Disabled)

Command: **Report Firmware Version**
Reports the version of the firmware installed in the board.
Message: F0 00 00 65 10 CH 1F F7
Return: F0 00 00 65 10 CH aa bb F7
Variables: aa=Major version number
bb=Minor version number

Command: **Report Hardware Version**
Reports the version number of the hardware installed in the board. The Major version number will be 01. The Minor version number will contain a number that reflects the Revision/Features of the chip specified by the Major version number.
Message: F0 00 00 65 10 CH 4F F7
Return: F0 00 00 65 10 CH aa bb F7
Variables: aa=Major version number
bb=Minor version number

Command: **Report Number of Samples**
Reports the number of samples that are currently loaded into the board.

Message: F0 00 00 65 10 CH 20 F7
Return: F0 00 00 65 10 CH nn nn F7
Variable: nn=Number of samples

Command: **Report Instantaneous Output Levels**
Reports the instantaneous levels of the left and right channels of the Sound-Core output accumulator. The values range from 0000h through 7FFFh.
Message: F0 00 00 65 10 CH 34 F7
Return: F0 00 00 65 10 CH ll ll ll rr rr rr F7
Variables: ll ll ll=Left Channel Output Level
rr rr rr=Right Channel Output Level

Command: **Report Peak Output Levels**
Reports the peak levels of the left and right channels of the Sound-Core output accumulator and a saturation count value. The level values will range from 0000h through 7FFFh. The saturation count indicates the number of consecutive times that the value sent to the accumulator was saturated to the maximum (or minimum) 16-bit value. Saturation count values range from 0 through 32. Values less than 32 are accurate, but once they hit 32 (bit 5 becomes set), the counter stops accumulating saturations. The operating system software performs a periodic monitoring of the Sound-Core output accumulator, and retains peak level and saturation values which are reset to zero each time that this command is executed.
Message: F0 00 00 65 10 CH 35 F7
Return: F0 00 00 65 10 CH ll ll ll rr rr rr ss F7
Variables: ll ll ll=Left Channel Output Level
rr rr rr=Right Channel Output Level
ss=Saturation Count

Sample and Patch Download Functions

The following messages are used to download samples, patches, and programs to the board. The only multiple message sequence is the sample download. It begins with a Download Sample message. It is followed by one or more Download Block messages, each block containing up to 4096 bytes of wavesample data.

The sample offset values are transferred in a format compatible with

the internal hardware. The values contain both a 20-bit integer and a 4-bit fractional component. The fractional component occupies bits 0-3 of the LSB, while the integer occupies the upper bits.

Command: **Download Sample**

This message begins the sample download process. After receiving the length of the sample, the onboard processor will attempt to allocate memory for the sample. If this fails, the board will immediately return an error message. Once the acknowledge has been received, the data blocks can be transmitted using the Download Block message.

Message: F0 00 00 65 10 CH 00 nn nn ll ll ll ll aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd dd ee ee ee ff ff F7

Return: F0 00 00 65 10 CH 00 F7

Variables: nn nn=Sample number (0-1FFh)
ll ll ll ll=Length of sample
aa aa aa aa=Offset to start of sample
bb bb bb bb=Offset to start of loop
cc cc cc cc=Offset to end of loop
dd dd dd dd=Offset to end of sample
ee ee ee=Sample Frequency Bias
*Use this Frequency Bias Equation: (log (44,100/SR)/log(2))*2048 + (MIDI Root Key*2048)/12
ff= Bits 0-1 Sample Type (See below)
Bit 2 Unused (set to 0)
Bit 3 Loop
Bit 4 Bi-directional loop
Bit 5 Unused (set to 0)
Bit 6 Reverse
Bit 7 Unused (set to 0)

Sample Type Listing:

Bit 1	Bit 0	Sample Type
0	0	16 Bit Linear
0	1	Unused
1	0	8 Bit Linear
1	1	8 Bit uLaw

Command: **Download Block**

The host should send blocks of 4,096 bytes of sample data (2,048 16-bit samples or 4,096 8-bit samples) for all but the last block of data. The last block should be the

remaining count rounded up to the next paragraph (16 bytes). Upon completion of each block, the board will send a DMA Complete message. If there are more blocks to be sent, the host should begin sending the next block.

Message: F0 00 00 65 10 CH 01 (sample data) F7
Return: F0 00 00 65 10 CH 00 01 F7

Command: **Download Sample Header**

This message can be used by the host to modify the sample header without having to reload all of the sample data. For example, the loop points can be modified allowing the user to “tune” the loops interactively.

Message: F0 00 00 65 10 CH 2C nn nn aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd dd ee ee ee ff ff F7
Return: F0 00 00 65 10 CH 00 F7
Variables: nn nn=Sample number (0-1FFh)
aa aa aa aa=Offset to start of sample
bb bb bb bb=Offset to start of loop
cc cc cc cc=Offset to end of loop
dd dd dd dd=Offset to end of sample
ee ee ee=Sample frequency Bias (See Download Sample.)
ff ff= Bit 0-1 Sample Type (See Download Sample.)
Bit 2 Unused (set to 0)
Bit 3 Loop
Bit 4 Bi-directional loop
Bit 5 Unused (set to 0)
Bit 6 Reverse
Bit 7 Unused (set to 0)

Command: **Upload Sample Header**

This message uploads the sample header to the host. If the Upload Sample Header command is received via MIDI System Exclusive, the response will be formatted as the associated Download Sample Header command.

Message: F0 00 00 65 10 CH 2D nn nn F7
Return: F0 00 00 65 10 CH 2C aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd dd ee ee ee ff ff F7
Variables: nn nn=Sample number (0-1FFh)
aa aa aa aa=Offset to start of sample
bb bb bb bb=Offset to start of loop
cc cc cc cc=Offset to end of loop
dd dd dd dd=Offset to end of sample

ee ee ee=Sample Frequency Bias (See Page 16 for equation.)

ff= Bits 0-1 Sample Type (See Page 16 for listing.)
Bit 2 Unused (set to 0)
Bit 3 Loop
Bit 4 Bi-directional loop
Bit 5 Unused (set to 0)
Bit 6 Reverse
Bit 7 Unused (set to 0)

Command: **Download Multisample**

This message is used to download a Multisample. The number of data bytes following the number of samples is a function of the number of samples. After the number of samples has been received, the onboard processor will attempt to allocate memory for the multisample. If this fails, the board will immediately return an error message. If it is more convenient, the host need not check for the error message until after all the data has been sent.

Message: F0 00 00 65 10 CH 02 nn nn ss aa aa bb bb cc cc ... zz zz F7
Return: F0 00 00 65 10 CH 00 F7
Variables: nn nn=Sample number (0-1FFh)
ss=Number of samples 0-7 (1=>2, 2=>4, 3=>8, 4=>16, 5=> 32, 6=>64, 7=>128)
aa aa=First sample number (0-1FFh)
bb bb=Second sample number
cc cc=Third sample number
...=More sample numbers
zz zz=Last sample number

Command: **Upload Multisample**

This message is used to upload a Multisample to the host. The number of data bytes following the number of samples is a function of the number of samples. If the Upload Multisample command is received via MIDI System Exclu sive, the response will be formatted as the associated Download Multisample command.

Message: F0 00 00 65 10 CH 2E nn nn F7
Return: F0 00 00 65 10 CH 02 ss aa aa bb bb cc cc ... zz zz F7
Variables: nn nn=Sample number (0-1FFh)
ss=Number of samples 0-7 (1=>2, 2=>4, 3=>8, 4=>16, 5=>32, 6=>64, 7=>128)
aa aa=First sample number
bb bb=Second sample number

cc cc=Third sample number
...=More sample numbers
zz zz=Last sample number

Command: **Download Sample Alias**

This Download Alias message is used to create an alias for an existing sample. The alias sample can have different loop points, start or end in a different place, or even be reversed. Its main purpose is to conserve memory by allowing the reuse of existing samples.

Message: F0 00 00 65 10 CH 03 nn nn ss ss aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd dd ee ee ee ff ff F7

Return: F0 00 00 65 10 CH 00 F7

Variables:
nn nn=Sample number (0-1FFh)
ss ss=Aliased sample number (0-1FFh)
aa aa aa aa=Offset to start of sample
bb bb bb bb=Offset to start of loop
cc cc cc cc=Offset to end of loop
dd dd dd dd=Offset to end of sample
ee ee ee=Sample Bias Frequency (See Download Sample.)
ff ff=
Bits 0-1 Unused (set to 0)
Bit 2 Unused (set to 0)
Bit 3 Loop
Bit 4 Bi-directional loop
Bit 5 Unused (set to 0)
Bit 6 Reverse
Bit 7 Unused (set to 0)

Command: **Upload Sample Alias**

The Upload Alias message is used to upload a Sample Alias Header to the host. If the Upload Sample Alias command is received via MIDI System Exclusive, the response will be formatted as the associated Download Sample Alias command.

Message: F0 00 00 65 10 CH 2F nn nn F7

Return: F0 00 00 65 10 CH 03 ss ss aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd dd ee ee ee ff ff F7

Variables:
nn nn=Sample number (0-1FFh)
ss ss=Aliased sample number (0-1FFh)
aa aa aa aa=Offset to start of sample
bb bb bb bb=Offset to start of loop
cc cc cc cc=Offset to end of loop
dd dd dd dd=Offset to end of sample
ee ee ee=Sample Frequency Bias (See Download Sample.)

ff ff= Bits 0-1 Unused (set to 0)
Bit 2 Unused (set to 0)
Bit 3 Loop
Bit 4 Bi-directional loop
Bit 5 Unused (set to 0)
Bit 6 Reverse
Bit 7 Unused (set to 0)

Command: **Delete Sample**
The Delete Sample message can be used to free up memory from a sample that is no longer required. Before downloading new samples to the board, it is a good idea to delete any unused samples, then send a Crunch message to collect all the unused samples into a single block.
Message: F0 00 00 65 10 CH 04 nn nn F7
Return: F0 00 00 65 10 CH 00 F7
Variable: nn nn=Sample number (0-1FFh)

Command: **Identify Sample Type**
The Identify Sample Type message can be used to determine the type of sample so that the appropriate upload message can be sent. It also gives the amount of sample memory taken up by the sample, including the header data, which can be useful for determining the actual memory overhead.
Message: F0 00 00 65 10 CH 30 nn nn F7
Return: F0 00 00 65 10 CH nn ss ss ss ss F7
Variables: nn nn=Sample number (0-1FFh)
nn= 0=Sample
1=Multisample
2=Alias
ss ss ss ss=Amount of memory occupied by sample in bytes

Command: **Upload Sample Parameters**
The Upload Sample Parameters command provides a means of uploading sample header, sample alias, or multisample data via a single command, without having to first issue an Identify Sample Type command. It was specifically implemented to make it feasable for a single SYSEX command string to return paramenters that could be used to reload the retrieved information at a later time. If the sample number supplied is that of a

regular sample, the response on a MIDI interface will be formatted as a Download Sample Header command (on the "host" interface the returned parameters will be described as a Upload Sample Header command). If the sample number supplied is that of a sample alias, the response on a MIDI interface will be formatted as a Download Sample Alias command (on the "host" interface the return will be as described for an Upload Sample Alias command). If the sample number supplied is that of a multisample, the response on a MIDI interface will be formatted as a Download Multisample command (on the "host" interface the returned paramenters will be described for an Upload Multisample command). If the sample number supplied is that of a valid sample number, but for which no associated sample data exists, the response on a MIDI interface will be formatted as a Delete Sample command (on the "host" interface the NAK/Bad Sample number indication will be returned).

Message: F0 00 00 65 10 CH 57 nn nn F7
Return: F0 00 00 65 10 CH (sample dependent parameter data) F7
Variable: nn nn=Sample number (0-1FFh)

Command: **Report Free Memory**
This command reports the amount of available memory in the synthesizer.
Message: F0 00 00 65 10 CH 05 F7
Return: F0 00 00 65 10 CH ff ff ff ff F7
Variable: ff ff ff ff=Free memory in bytes

Command: **Download Patch**
The Download Patch message is used to download patch data to the board. There are 66 bytes in a patch, which will be transmitted as 132 bytes of MIDI data. Note that each data byte is sent as two bytes with the first containing the least significant seven bits and the second containing the most significant bit.
Message: F0 00 00 65 10 CH 06 nn nn aa aa bb bb cc cc ... zz zz F7
Return: F0 00 00 65 10 CH 00 F7
Variables: nn nn=Patch number (0-FFh)
aa aa=First byte of patch data
bb bb=Second byte of patch data
cc cc=Third byte of patch data
...=More data

zz zz=Last byte of patch data

Command: **Upload Patch**

The Upload Patch message is used to upload patch data sent to the board. There are 66 bytes in a patch, which will be transmitted as 132 bytes of MIDI data. Note that each data byte is sent as two bytes with the first containing the least significant seven bits and the second containing the most significant bit. The response will be formatted as the associated Download Patch command.

Message: F0 00 00 65 10 CH 23 nn nn F7
Return: F0 00 00 65 10 CH 06 aa aa bb bb cc cc ... zz zz F7
Variables: nn nn=Patch number (0-FFh)
aa aa=First byte of patch data
bb bb=Second byte of patch data
cc cc=Third byte of patch data
...=More data
zz zz=Last byte of patch data

Command: **Download Program**

The Download Program message is used to download program data to the board. There are 16 bytes in a program, which will be transmitted as 32 bytes of MIDI data. Note that each data byte is sent as two bytes with the first containing the least significant seven bits and the second containing the most significant bit.

Message: F0 00 00 65 10 CH 07 nn aa aa bb bb cc cc ... zz zz F7
Return: F0 00 00 65 10 CH 00 F7
Variables: nn=Program number (0-7Fh)
aa aa=First byte of program data
bb bb=Second byte of program data
cc cc=Third byte of program data
...=More data
zz zz=Last byte of program data

Command: **Upload Program**

The Upload Program message is used to upload program data from the board to the host. There are 16 bytes in a program, which will be transmitted as 32 bytes of MIDI data. Note that each data byte is sent as two bytes with the first containing the least significant seven bits and the second containing the most significant bit. The response will be formatted as the associated Download Program command.

Message: F0 00 00 65 10 CH 24 nn F7
Return: F0 00 00 65 10 CH 07 aa aa bb bb cc cc ... zz zz
Variables: nn=Program number (0-7Fh)
aa aa=First byte of program data
bb bb=Second byte of program data
cc cc=Third byte of program data
...=More data
zz zz=Last byte of program data

Command: **Download Enhanced Drum Program**

The Enhanced Drum Program consists of a four-byte program for each of the 128 MIDI note numbers on the drum channel. The Download Enhanced Drum Program message is used to download one MIDI note's drum program data to the board. These four bytes in a single enhanced drum mode, are transmitted as eight bytes of MIDI data. Note that each data byte is sent as two bytes with the first containing the least significant seven bits and the second containing the most significant bit. When this message is received by the board, only the enhanced drum program data for the MIDI note number to which this data was directed will be changed.

Message: F0 00 00 65 10 CH 31 nn aa aa bb bb cc cc dd dd F7
Return: F0 00 00 65 10 CH 00 F7
Variables: nn=MIDI note number (0-127)
aa aa=First byte of enhanced drum program data
bb bb=Second byte of enhanced drum program data
cc cc=Third byte of enhanced drum program data
dd dd=Fourth byte of enhanced drum program data

Command: **Upload Enhanced Drum Program**

The Upload Enhanced Drum Program message is used to upload drum program data from the board to the host. There are four bytes in an enhanced drum program, which will be transmitted as eight bytes of MIDI data. Note that each data byte is sent as two bytes with the first containing the least significant seven bits and the second containing the most significant bit. The response will be formatted as the associated Download Enhanced Drum Program command.

Message: F0 00 00 65 10 CH 32 nn F7
Return: F0 00 00 65 10 CH 31 aa aa bb bb cc cc dd dd F7
Variables: nn=MIDI note number (0-127)
aa aa=First byte of enhanced drum program data

bb bb=Second byte of enhanced drum program data
cc cc=Third byte of enhanced drum program data
dd dd=Fourth byte of enhanced drum program data

Command: **Set Enhanced Drum Program Channel**
The Set Enhanced Drum Program Channel message is used to place the selected MIDI channel in a drum program mode using enhanced drum program data. Any number of MIDI channels can be set to enhanced drum program mode by repeated use of the Set Enhanced Drum Program Channel message. Once a channel has been set into an enhanced drum channel mode, it will play whatever program data has been loaded via previous or subsequent Download Enhanced Drum Program messages. It will ignore all Program Changes sent to this channel until the board is reset, or a Disable Drum Program message is sent.
Message: F0 00 00 65 10 CH 33 nn F7
Return: F0 00 00 65 10 CH 00 F7
Variable: nn=MIDI Channel number (0-15)

Command: **Disable Drum Program**
The disable Drum Program message is used to disable the Drum Program on a particular channel. After this message is received, the channel will once again respond to the Program Changes.
Message: F0 00 00 65 10 CH 22 nn F7
Return: F0 00 00 65 10 CH 00 F7
Variable: nn=MIDI Channel number (0-15)

Command: **Report Channel Program Numbers**
The Report Channel Number message is used to upload the program number for the 16 MIDI Channels. The standard MIDI program numbers range from 0 through 127 and are set via the standard MIDI Program Change message. A MIDI channel which reports a program number of 129 indicated that it is selected as an Enhanced Drum Channel as set using the Set Enhanced Drum Program Channel message.
Message: F0 00 00 65 10 CH 36 F7
Return: F0 00 00 65 10 CH aa aa bb bb cc cc dd dd ee ee ff ff gg gg hh hh ii ii jj jj kk kk ll ll mm mm nn nn oo oo pp pp F7
Variables: aa aa=Program number for MIDI Channel #1

bb bb=Program number for MIDI Channel #2
cc cc=Program number for MIDI Channel #3
dd dd=Program number for MIDI Channel #4
ee ee=Program number for MIDI Channel #5
ff ff=Program number for MIDI Channel #6
gg gg=Program number for MIDI Channel #7
hh hh=Program number for MIDI Channel #8
ii ii=Program number for MIDI Channel #9
jj jj=Program number for MIDI Channel #10
kk kk=Program number for MIDI Channel #11
ll ll=Program number for MIDI Channel #12
mm mm=Program number for MIDI Channel #13
nn nn=Program number for MIDI Channel #14
oo oo=Program number for MIDI Channel #15
pp pp=Program number for MIDI Channel #16

Host Command and System Exclusive Command Tables

The ICS WaveFront functions listed in this document are accessible through the SysEx and directly through the host port. To access the functions via SysEx, a string must be built as shown in the Host interface section of this document. When using the host port, the commands must have their high bit set. This can be accomplished by adding 80h to the commands. The following tables show both the Host Command Number and System Exclusive ID for the System Exclusive messages.

Host Command Number	System Exclusive ID	Command Descritption
89h	09h	Set Synthesizer Volume
92h	12h*	Get Synthesizer Volume
8Bh	0Bh	Set Number of Voices
94h	14h*	Get Number of Voices
A6h	26h	Set Synthesizer Tuning
A7h	27h*	Get Synthesizer Tuning
9Ah	1Ah	Disable Synth Channel
9Bh	1Bh	Enable Synth Channel
ABh	2Bh	Get Synth Channel Status
9Dh	1Dh	Disable MIDI-In to Synth
9Eh	1Eh	Enable MIDI-In to Synth
A8h	28h	Enable Virtual MIDI Mode
A9h	29h	Disable Virtual MIDI Mode
AAh	2Ah	Report MIDI Status

9Fh	1Fh	Report Firmware Version
CFh	4Fh	Report Hardware Version
A0h	20h	Report Number of Samples
B4h	34h	Report Instantaneous Output Levels
B5h	35h	Report Peak Output Levels

* These "get parameter" commands via MIDI System Exclusive will result in the return of the data formatted as the associated "set parameter" command.

Host Command Number	**System Exclusive ID**	**Command Description**
80h	00h	Download Sample
81h	01h	Download Block
ACh	2Ch	Download Sample Header
ADh	2Dh*	Upload Sample Header
82h	02h	Download Multisample
AEh	2Eh*	Upload Multisample
83h	03h	Download Sample Alias
AFh	2Fh*	Upload Sample Alias
84h	04h	Delete Sample
B0h	30h	Identify Sample Type
D7h	57h	Upload Sample Parameters
85h	05h	Report Free Memory
86h	06h	Download Patch
A3h	23h*	Upload Patch
87h	07h	Download Program
A4h	24h*	Upload Program
B1h	31h	Download Enhanced Drum Program
B2h	32h*	Upload Enhanced Drum Program
B3h	33h	Set Enhanced Drum Program Channel
A2h	22h	Disable Drum Program
B6h	36h	Report Channel Program Numbers

* These "get parameter" commands via MIDI System Exclusive will result in the return of the data formatted as the associated "set parameter" command.