WAVESTATION

ADVANCED VECTOR SYNTHESIS • WAVE SEQUENCING

EX

by Dan Phillips

KORG

Ⓔ①

avi AV Synthesis System

TABLE OF CONTENTS

ABOUT THIS ADDENDUM 1

EXPANDED EFFECTS 2
- DISTORTION - FILTER - EQ 2
- MOD PITCH SHIFT-DELAY 3
- STEREO COMPRESSOR-LIMITER/GATE 4
- SMALL VOCODER 1/2/3/4 5
- STEREO VOCODER-DELAY 1/2 6
- Cross-Timbral Modulation Synthesis using the Vocoders 7

SYSEX DATA TRANSMIT 9

GLOBAL 9

OTHER ENHANCEMENTS 10
- Multiset Channel Level now sends out MIDI Volume 10
- Key and Velocity Zones now programmable via MIDI 10
- Bank changing streamlined on Performance Select page 10
- Modulation names more consistent 10

TROUBLESHOOTING 11
- Wavestation makes no sound: Audio Troubleshooting 11
- Wavestation makes no sound: MIDI Troubleshooting 11
- The Wavestation doesn't respond to some MIDI notes 12
- Notes cut off unexpectedly 12
- Only one step of a Wave Sequence is being played 13
- ROM Wave Sequences cannot be edited 13
- Wave Sequence does not seem to sync to MIDI 13
- When a Performance is used in a Multiset, not all Parts are heard 14
- When the Effects Mix of a Performance is changed, not all Parts are heard 14
- Performance Effects seem to have changed 14
- Performance Effects cannot be edited 14
- Multiset Effects cannot be edited 14
- Local footpedals do not function correctly 15

COMPATIBILITY 16

SYSTEM EXCLUSIVE DATA FORMAT 17

KORG Wavestation Version 3 Addendum

Written by Dan Phillips

Editing and additional material by:
John Bowen
Joe Bryan
Charlie Bright
Karl Hirano
Stanley Jungleib
Ray Keller
Alex Limberis
Scott Peterson

ABOUT THIS ADDENDUM

Version 3 of the Wavestation software enhances the instrument in a number of ways, while maintaining complete compatibility with the sounds and features of the older versions (so that you can still do all of your favorite tricks). Here is a quick list of the major improvements:

- 8 new effects have been added, including:
  - Mod Pitch Shift-Delay, a modulatable pitch shifter
  - Stereo Compressor-Limiter/Gate
  - Small Vocoder 1/2/3/4
  - Stereo Vocoder Delay 1/2
- The Overdrive and Distortion effects now offer a modulatable output level.
- The Performance Select Map may now be sent as a SysEx Dump.
- The local keyboard may now be transposed.
- Multiset channels now send and receive the MIDI Volume controller.
- Key and Velocity Zones information may now be entered via MIDI.

This addendum briefly introduces and documents those enhancements, in addition to providing a new troubleshooting reference. It is intended to be a companion to the Wavestation Player's and Reference Guides, and not a substitute. If you have any questions about the features discussed within, a quick glance at the other manuals should clear things up.

For more information - read on!

EXPANDED EFFECTS

The Wavestation now features 8 new effects, numbered 48-55. These include the Mod Pitch Shift-Delay, Stereo Compressor-Limiter/Gate, and 6 Vocoders.

Additionally, the Distortion-Filter-EQ and Overdrive-Filter-EQ effects now feature a modulatable output level.

DISTORTION - FILTER - EQ

The amount of Distortion/Overdrive is related to the level of the input signal. Using MIDI Volume to change the level of a Performance with these effects will thus also change its timbre. To change the volume of a distorted/overdriven Performance without altering its timbre, use the Level modulation parameter instead.

29 Distortion - Filter - EQ

This effect has a "dirty" sound and "wah" effect. It is effective for solos.

30 Overdrive - Filter - EQ

This is an effect that simulates the overdrive generally used by guitars, and is particularly effective when applied to organs and electric pianos to create guitar-like lines and solos. It also has a "wah" effect.

Parameters

Dry/Wet Mix	***DRY, 9/1, . . .1/9, WET*** Output balance of processed and unprocessed sound.
Footswitch	***DISABLE/ENABLE*** Enables or disables use of EFFECTS SWITCH to turn effect on or off.
Edge	***1 to 111*** Amount of drive.
Hot Spot	***0 to 100*** Controls the center frequency of the "wah" filter. Try modulating this parameter with a pedal or mod wheel for the classic"wah-wah" effect.
Hot Spot mod source	***Mod source***
Hot Spot mod amount	***-15 to +15***
Resonance	***0 to 100*** Filter "Q" factor. This controls the amount of "wah" effect.
Level	***0 to 100*** Output level of the effect.
Level mod source	***Mod source***
Level mod amount	***-15 to +15***

Low EQ

-12 to +12 dB
Control for cutting or boosting the low frequencies.
EQ is applied to the wet signal only; the direct signal is unaffected.

High EQ

-12 to +12 dB
Control for cutting or boosting the high frequencies.
EQ is applied to the wet signal only; the direct signal is unaffected.

MOD PITCH SHIFT-DELAY

48 Mod pitch shift-Dly

This pitch shifter allows the amount of shift to be modulated. The input may be shifted either up or down, and the shifted signal may also be delayed with respect to the original signal, with an adjustable feedback amount. Some applications of this include "whammy-bar" pitch bending and special effects.

Parameters

Dry/Wet Mix

DRY, 9/1, . . .1/9, WET
Output balance of processed and unprocessed sound.

Dry/Wet Mix mod source

Mod source

Dry/Wet Mix mod amount

-15 to +15

Delay Left

0 to 490 ms

Delay Right

0 to 490 ms

Feedback

0 to 100
This is the feedback amount for the delay lines.

Max Shift

-12 to +12
This is the maximum amount of pitch shift, in semitones.

Shift Scaler

1 to 100%
This determines the initial amount of pitch shift without modulation, as a percentage of the Max Shift amount.

Shift Scaler mod source

Mod source

Shift Scaler mod amount

-15 to +15
If the Shift Scaler is set to 1, only positive modulation will have an effect; if it is set to 100, only negative modulation will have an effect.

STEREO COMPRESSOR-LIMITER/GATE

49 Stereo comp-lim/Gate

The compressor provides an automatically controlled volume envelope, which can be used to smooth out the level of a signal (often done with guitars), or to give a sound more "punch" (often done with drums). The ability to use a single FX Bus as the control source allows you to create side-chain effects, linking the compression of one signal to the level of another.

A gate is also provided. Signals of a certain minimum volume (the Threshold amount) "lift" the gate, and are allowed to pass through; signals under that volume are not.

Parameters

Control Source

NORMAL, BUS A+B, BUS C+D, BUS A/B/C/D
NORMAL uses the input signal to control the compression amount. To allow you to achieve side-chain effects, BUS A+B and C+D use the sum of the two FX Buses to control the compression amount, and BUS A-D use the levels from a single FX Bus .

Control Source Sensitivity

0-10
This parameter sets the input level for the Control Source.

Compression Ratio

0 to 100
This parameter sets the amount of compression.

Compression Threshold

0 to 100
This parameter sets the level at which compression will begin.

Gate Threshold

0 to 100
This parameter sets the level at which the gate is lifted, letting the signal through.

Output Level

0 to 100
This parameter sets the output level of the compressor.

SMALL VOCODER 1/2/3/4

The Vocoder effects superimpose the timbre of one signal (the Modulator) onto that of a second signal (the Carrier). A standard application of this is the "talking" instrument, in which you talk into a microphone and a guitar or keyboard sound is made to mimic the harmonic content of the speech.

Speech effects are the most commonly used application of the Vocoder, and they're what the first vocoders were designed to do; but they are not by far the limit of its capabilities. The Vocoders can modulate one or more Wavestation Patches or Waves to achieve new, dynamic timbres. You can even combine Vector and/or Wave Sequence sounds in this cross-timbral modulation synthesis, and then store them as a new Performance (for more information on this subject, please see the tutorial below).

A vocoder is essentially a combination of a frequency analyzer and a dynamic EQ. The Modulator signal is divided up into a number of frequency bands, and the levels of each of these bands are measured in real time. A dynamic EQ is slaved to the analyzer, following the changes in each band of the Modulator with similar changes in the EQ of the Carrier. This causes the Carrier to assume some of the timbre of the Modulator. It is best for the Carrier to contain a wide range of frequencies, because if there is little or no material in some of the bands to begin with, the EQ will have nothing to alter, and the Vocoder's effect will be diminished.

The more frequency bands which are used, the greater the definition of the Vocoder effect. To achieve the highest quality Vocoder, the two Stereo Vocoder - Delay algorithms use both effects slots; the four Small Vocoder algorithms use the normal effects configuration, making another effect simultaneously available.

The Vocoder may be used with any combination of sounds. Since the designation of Carrier and Modulator is based on the FX Bus, you must make sure that any applicable settings on the Patch FX Bus Assignment page, as well as the Performance Part Detail FX Bus parameter, are configured appropriately.

50 Small vocoder 1

This vocoder uses low to mid-high frequency bands. It has a wider band covering the bass range, for enhanced low-end response.

51 Small vocoder 2

This vocoder uses mid-low to high frequency bands. It has a wider band covering the treble range, for enhanced high-end response.

52 Small vocoder 3

This vocoder uses a number of low to mid-high frequency bands of even width.

53 Small vocoder 4

This vocoder uses a number of mid-low to high frequency bands of even width.

Parameters

Modulator Bus	***A,B,C,D*** FX Bus used as source for the Vocoder Modulator.
Modulator Bus Sensitivity	***0 to 100*** This sets the input level for the Modulator. If you hear distortion, try turning this value down.
Carrier Bus	***A,B,C,D*** FX Bus used as source for the Vocoder Carrier.

Carrier Bus Sensitivity — ***0 to 100***
This sets the input level for the Carrier. If you hear distortion, try turning this value down.

Sibilance — ***0 to 10***
Controls the amount of high frequencies from the Modulator (such as vocal consonants, as in "ch" and "ss") included in the mix.

Sibilance mod source — ***Mod source***
The default mod source is KEYDN, which allows you to use the Key Down time to gate the sibilance amount.

Sibilance mod amount — ***-15 to +15***

STEREO VOCODER-DELAY 1/2

The two Stereo Vocoder - Delays are extremely powerful algorithms, and use both effects slots. When you select one of the Stereo Vocoders for Effect 1 or 2, the other Effect changes to display Stereo Vocoder as well.

For more information on the Vocoders, see the discussion of the Small Vocoders, above.

54 Stereo vocoder - Delay 1

This vocoder uses wide frequency bands on the treble and bass ranges, and a number of narrower bands in the mid-range.

55 Stereo vocoder - Delay 2

This Vocoder uses a number of bands of even width, across the frequency range.

Parameters

Modulator Bus — ***A,B,C,D***
FX Bus used as source for the Vocoder Modulator.

Modulator Bus Sensitivity — ***0 to 100***
This sets the input level for the Modulator. If you hear distortion, try turning this value down.

Carrier Bus — ***A,B,C,D***
FX Bus used as source for the Vocoder Carrier.

Carrier Bus Sensitivity — ***0 to 100***
This sets the input level for the Carrier. If you hear distortion, try turning this value down.

Sibilance — ***0 to 10***
Controls the amount of high frequencies from the Modulator (such as vocal consonants, as in "ch" and "ss") included in the mix.

Sibilance mod source — ***Mod source***
The default mod source is KEYDN, which allows you to use the Key Down time to gate the sibilance amount.

Sibilance mod amount — ***-15 to +15***

Stereo Width — ***0 to 10***
Increasing this value causes the stereo effect to become more prominent.

Delay Time — ***0 to 1000 ms***

Feedback — ***0 to 100***

Delay Level — ***0 to 100***

Cross-Timbral Modulation Synthesis using the Vocoders

In this simple example, we'll use two Patches to create a hybrid sound. You can use two or more Patches as the Carrier and/or Modulator, simply by setting them to the same FX Bus.

- Initialize a Performance.
- Go to the Performance Part Detail page. Select two Patch(es) which you would like to use in your cross-timbral modulation, and place them in the first two Parts of the Performance. It's best to use sounds with a wide frequency range, such as sawtooth-like waves.
- On the same page, set the FX Bus for the first Part to "A," and the FX Bus for the second to B.
- Go to the EFFECTS page, and select Stereo Vocoder -Delay 1 or 2 as Effect 1 for the Performance. These effects are so powerful that they require both effects slots, so you'll notice that Effect 2 changes to read Stereo Vocoder - Delay also.
- Press the soft key FX 1 EDIT, so that you can set up the Vocoder.
- On the Vocoder edit page, set the Modulator Bus to "A" (which you assigned Part 1 to, above) and the Carrier Bus to "B" (Part 2). This means that Part 1 is the Modulator, and Part 2 is the Carrier, so that Part 1's sound will be superimposed on Part 2's.
- Try playing the sound!

More Vocoder Tips

The Vocoder's Sibilance parameter determines how much of the original Modulator's higher frequencies are heard. If you want to hear more of that sound, turn this parameter up.

In addition to using two sounds to modulate each other, you can try using a single sound to modulate itself. To do this, you can either place the same sound on two Parts (similar to the example above), or simply assign the FX Bus of a single Part to 50/50.

Another interesting application is to use a rhythmic, percussive Wave Sequence as the modulator, and a bright pad as the carrier. The pad will be "triggered" by the Wave Sequence's percussion. This is especially effective when using a sequencer and synching Wave Sequences to MIDI Clocks, so that the Vocoder timbre creates a cool, percolating rhythm track.

Stereo Vocoder-Delay 1/2 and the Effects Mix

Since the Stereo Vocoder-Delays can use any of the effects buses for both the Carrier and the Modulator, the routings on the Effects Mix page work slightly differently from those of other effects. Buses A and B can only be routed through the Vocoder; if they are not used as Carrier or Modulator, they are not heard. Buses C and D, however, may be routed both through the Vocoder and as set by the Effects Mix page, which works with the Stereo Vocoders in a couple of special ways.

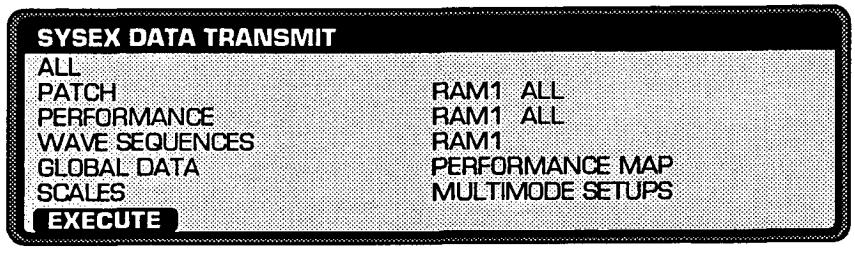
The Effects Mix Parallel routing functions almost as usual, allowing you to pan C and D across the stereo outputs as if FX 2 were set to the NULL EFFECT. The Effects Mix Series routing is somewhat more altered, so that the Wet/Dry Mix controls how much of the original sounds of Buses C and D are heard, without

affecting the level of the Vocoder output. Wet means that only the Vocoder output is heard, and Dry means that the original sounds are heard at full volume, along with the Vocoder output. This feature allows you to use buses A and B for the vocoder, and simultaneously route buses C and D directly to the stereo outputs. By using buses C and/or D as the Modulator, you can also use the Effects Mix to blend in some of the Modulator's original sound. If you wish to pass through only the high frequencies of the Modulator (a typical vocoder application), use the Vocoder's Sibilance parameter instead.

For more information on the Effects Mix, please see Section 7.2 of the Player's Guide (Effects Buses and Routing), and Effects Mix in the Reference Guide.

SYSEX DATA TRANSMIT

Path: MIDI - SYSEX

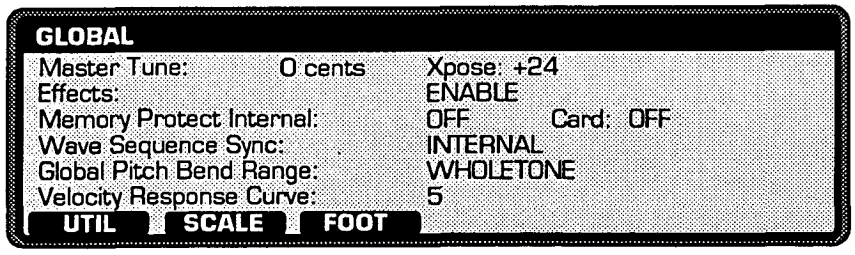


The SYSEX DATA TRANSMIT page now allows you to send a dump of the Performance Map. To do this, move the cursor until PERFORMANCE MAP is in inverse video, and then press EXECUTE.

For more information on the Sysex Data Transmit page, please see the Reference Guide.

GLOBAL

Path: GLOBAL



The GLOBAL page now features a local Xpose feature.

Xpose

Xpose will transpose within a range of +/- 24 semitones. Setting Xpose to +4, for instance, will transpose notes played by the local keyboard up by 4 semitones.

This parameter is intended as a performance feature, and transposes Wavestation sounds played by the local keyboard only. It has no effect on MIDI transmitted or received by the Wavestation, and so is not optimally used in a sequencing environment. To transpose MIDI notes, use the Key Offset Amount on the MIDI page.

For more information on the Global page, please see the Reference Guide.

OTHER ENHANCEMENTS

Multiset Channel Level now sends out MIDI Volume

In Multi and Mono modes, each of the 16 channels has its own volume level. This parameter has always responded to MIDI Volume (Controller #7); now, MIDI Volume may be transmitted as well. Whenever you change the volume of a Performance in a Multiset, a MIDI Volume message is sent out on its channel, so that it can be recorded by a sequencer and played back into the Wavestation for automated mixing.

Key and Velocity Zones now programmable via MIDI

Key and velocity limits may now be entered over MIDI, just as they are by using the local keyboard. You may find this to be convenient if you are using another instrument as a master controller, so that velocity ranges can be made to match its particular responsiveness.

Bank changing streamlined on Performance Select page

To avoid confusion, the BANK soft key is now the only way to change Banks on the Performance Select page. It is no longer possible to cursor to the Bank field.

Modulation names more consistent

In the effects, MIDI Controllers 1/2 are now known as MIDI 1/2, as opposed to XMIDI1/2. They are thus consistent with the names in the Patch Modulation Matrix.

Additionally, the Foot Controller (MIDI Controller #4) is now called MOD PEDAL in the Patch Modulation Matrix. It is thus consistent with modulation names in the effects and on the FOOT PEDAL ASSIGN page.

TROUBLESHOOTING

Wavestation makes no sound: Audio Troubleshooting

- Check the MASTER VOLUME slider and volume pedal (if used). The polarity of a Volume Pedal is also important; if this is inverted, the volume will be at zero when the pedal is at maximum. If this seems to be the case, change the polarity of the pedal on the FOOT PEDAL ASSIGN page, under GLOBAL.
- Perhaps someone has edited the current Performance into silence. Try selecting different Performances -- particularly ones in the ROM bank.
- If you still do not obtain audio output, it is easy to check whether the problem is in the Wavestation or your sound system by plugging headphones directly into the back panel PHONES jack. If you can hear sound through the headphones, check the connections to your sound system.
- If you don't hear any sound through the headphones, do the MIDI check explained below.

Wavestation makes no sound: MIDI Troubleshooting

If you are using the Wavestation with a sequencer or other MIDI controller, and the Wavestation is not making any sound, it is possible that your MIDI system is not properly connected or configured. To see if this is the case, first go to the MIDI Status page.

This page displays an asterisk under the number of each MIDI channel on which the Wavestation is receiving MIDI data. If your controller is sending on channel 3, for instance, you will see an asterisk under the "3" every time you press down a key.

- Go to the MIDI STATUS page (path: MIDI-STATUS). As you play on your controller, you should see an asterisk (*) appear under one or more of the channel numbers.

If no asterisk appears:

- Check to see that your MIDI cable connections are properly made (MIDI Out from your controller or sequencer to MIDI In on the Wavestation).

  If you are using a MIDI patch bay, or the "through" function of a sequencer, try directly connecting the MIDI Out of your controller with the MIDI In of the Wavestation.

If an asterisk does appear:

- Note its channel number.

Next, press EXIT to get back to the MIDI page, and check to see which MIDI Mode (OMNI, POLY, MULTI, or MONO) the Wavestation is currently in.

- If it is set to OMNI Mode, the Wavestation will respond to MIDI information on any channel. If the STATUS page showed any activity, then, you should be hearing something. If you aren't, check again to see that the audio connections are OK.

- If it is set to POLY Mode, then the Wavestation will ignore all MIDI data except that which it receives on its Basic Channel. Adjust either the Wavestation's Basic Channel or your controller's send channel so that they match.
- If it is set to MULTI Mode, go to the MULTIMODE SETUP page (reachable by pressing the MULTI softkey on the MIDI page). In this mode, the Wavestation can receive on up to 16 channels simultaneously. Check to see that the desired channels are turned ON (the second column), and that their levels are reasonably high (the third column).
- If it is set to MONO Mode, note the (#) MONO CHANNELS parameter to the right of the Basic Channel. This sets the total number of channels to be used. These begin with the current Basic Channel, up to the number of mono channels requested, to the limit of 16. For example, if the Basic Channel is set to 1, and the "# MONO Channels" set to 6, then the Wavestation would receive MIDI on channels 1 through 6. Check that these parameters are set appropriately.

  This mode, like MULTI, uses Multisets to assign Performances to the different MIDI channels. Go to the MULTIMODE SETUP page by pressing the MULTI softkey on the MIDI page, and check to see that the desired channels are turned ON (the second column), and that their levels are reasonably high (the third column).

There are a few MIDI parameters which can cause silence regardless of the current MIDI mode.

- In a MIDI network, a controller can send unintended low Volume control messages. If you think this is the case, try raising the same controller, or reset the Wavestation by cycling power off, then on.
- Check that the Play Mode on the PERFORMANCE PART DETAIL page is set to BOTH or LOCAL for each Part. If any are set to MIDI, they will not sound, but will only transmit MIDI notes, Program changes, and controller data.

For more information on the Wavestation and MIDI, see Section 5 (MIDI) of the Player's Guide.

The Wavestation doesn't respond to some MIDI notes

- Check that the Note On/Off parameter on the MIDI RECEIVE page is set to ALL.

This feature is designed to allow you to link two Wavestations (or a Wavestation and a Wavestation A/D rack module) together. Setting one instrument to EVEN and the other to ODD causes each to ignore half of the MIDI notes (playing a single whole-tone scale), effectively doubling the available polyphony. Unless you have two modules operating in this manner, Note On/Off should be set to ALL.

Notes cut off unexpectedly

- Check the All Notes Off parameter on the MIDI RECEIVE page, and try setting it to IGNORE.

Some controllers send this MIDI message whenever there are no keys held down, and this can occasionally cause notes to cut off; ignoring the message will solve this problem.

Only one step of a Wave Sequence is being played

There are several possible causes for this situation.

- Check the Wave Sequence Sync parameter on the Global page. Unless you are specifically using MIDI clocks to control the playback of the Wave Sequence, this should be set to INTERNAL. If you are intending to use MIDI clocks for sync, make sure that your clock source - probably a sequencer or drum machine - is indeed sending MIDI clocks, and that its MIDI Out is connected to the Wavestation's MIDI In. MIDI Time Code is not the same as MIDI Clocks, and will not work for this purpose.
- Check that the SOLO soft key on the Wave Sequence page is not marked by brackets, which indicates that SOLO is on. If it is, press the soft key again to turn it off.

If SOLO is on, only the currently selected Wave Sequence Step is played.

- Check that the Mod Amount parameter on the Wave Sequence Utilities page is set to greater than 0, or that the Mod Source parameter is set to Linear Keyboard, Centered Keyboard, Linear Velocity, or Exponential Velocity.

If the source is not one of the four listed above, and the Mod Amount is set to a very small amount (such as 0), then only the start step of the sequence will be played.

ROM Wave Sequences cannot be edited

Each time you make any change to a Wave Sequence, the change is saved. Since you cannot write to ROM, you cannot directly edit a ROM Wave Sequence. If you first copy the Wave Sequence to a RAM bank, you can then edit it as much as you like.

- Go to the UTILITIES page, under GLOBAL, and copy the ROM Wave Sequence into either the RAM1, RAM2, or CARD banks. It may now be edited.

Wave Sequence does not seem to sync to MIDI

- Check that Wave Sequence Sync parameter on the GLOBAL page is set to MIDI.

If this is set to INTERNAL, MIDI clocks will not affect Wave Sequences.

- Check that the step durations are in multiples of 6 (12, 24, etc.) for all Wave Sequences in the current Performance.

A duration of 24 equals one quarter note; 12 equals an eighth note; 6 equals a sixteenth note, and so on.

- Check that rhythm is not partially due to the Wave Sequence being run through a delay effect. If this is the case, you should adjust the delay time to match the tempo of the MIDI clocks.
- For best results when using a sequencer and syncing Wave Sequences to MIDI clocks, quantize all notes playing Wave Sequences to a few milliseconds before the beat. This will ensure that your sequencer will send out the notes before the clock message, so that the Wave Sequence rhythms will be right on the beat.

When a Performance is used in a Multiset, not all Parts are heard

When the Effects Mix of a Performance is changed, not all Parts are heard

If you have changed the FX Mix of a Performance, or if you are using it in a Multiset and have not explicitly copied the effects from the original Performance, waves assigned to the C and/or D buses may not be heard.

- ☛ Check the Mix 3/4 parameters on the EFFECTS MIX page, and make sure that these are not set to OFF. If they are, change them to another setting.

If any of your Parts are assigned to the C,D,or C+D FX Buses (or if the Part is assigned to Patch, and the Patch FX Bus Assign has Waves which are routed to only C and/or D), and you are using only the stereo outputs, then it is necessary to use the Mix 3/4 parameters on the FX MIX page to route those Parts to the stereo outs. ROM 0 Wave Song is an example of such a Performance.

Performance Effects seem to have changed

If Effects have been set to DISABLE on the GLOBAL page, no effects will be heard.

- ☛ Go to the GLOBAL page, and make sure that the Effects are set to ENABLE.

When you play a Performance in MIDI MULTI or MONO modes, it is processed through the effects for the current Multiset, as opposed to its own effects.

- ☛ Check the MODE parameter on the MIDI page. If this is set to MULTI or MONO, the Performance is using the effects of the current Multiset (which may be accessed through the MULTI button on this page).
- ☛ Change the MIDI MODE to OMNI or POLY, which will enable the Performance to use its own effects.
- ☛ On the EFFECTS COPY page, copy the effects from the desired Performance into the Multiset.

Performance Effects cannot be edited

Multiset Effects cannot be edited

There are two sets of effects in the Wavestation: those that belong to Performances, and those that belong to Multisets. The MODE parameter on the MIDI page determines, among other things, which set of effects is in use. If the MIDI MODE is set to OMNI or POLY, the Performance effects are heard; if it is set to MULTI or MONO, the Multiset effects are heard.

Only the effects currently in use may be edited. Thus, if you are in OMNI or POLY modes, Multiset effects cannot be edited; if you are in MULTI or MONO modes, the Performance effects cannot be edited.

Also, if Effects have been set to DISABLE on the GLOBAL page, the effects will not be heard, and cannot be edited.

- ☛ Go to the GLOBAL page, and make sure that the Effects are set to ENABLE.

Local footpedals do not function correctly

If you are using a footpedal plugged into the back of the Wavestation, as opposed to over MIDI, and the pedal seems to be functioning oppositely from the way that you would expect (sustaining when it is not depressed, and damping when it is depressed, for example), the polarity of the pedal may be set incorrectly.

☛ Try changing the Polarity parameter for that footpedal on the FOOT PEDAL ASSIGN page.

If you are using the pedal as MOD PEDAL or FX SWITCH, remember that the values may be scaled and inverted at the modulation destination. A positive value from the pedal, then, may produce a decrease in modulation, depending on the settings of the Patches and Effects in the current Performance.

COMPATIBILITY

Wavestation software version 3.0 is completely compatible with Program and PCM data created for earlier versions. Several features have been added, including extra effects and the optional EXK-W PCM expansion kit, which require that you be slightly careful when transferring data in the other direction, back to earlier or unexpanded Wavestations.

This appendix addresses two of the major improvements made to the Wavestation in software version 3.0, and how these are handled by earlier Wavestations. Wavestations with this software update are referred to as "3.0 or later," and those without it are referred to as "pre-3.0."

Extra effects

The Wavestation's Stereo Mod Pitch Shift/Delay, Compressor - Limiter/Gate, and Vocoder effects are not present in pre-3.0 Wavestation keyboards (these extra effects are also featured in the Wavestation A/D). If a Performance which uses these effects is transferred to a pre-3.0 keyboard, they will appear as effect #0, NO EFFECT, and will not be heard. As long as these effects are not edited, they will still be intact if transferred back to a 3.0 or later Wavestation (or Wavestation A/D), but changing the effects will erase the original data.

The Distortion/Overdrive effects are slightly enhanced in 3.0 or later Wavestations, with the addition of a modulation source and amount on the output level. These parameters will not appear on pre-3.0 keyboards, but the effects will otherwise function normally.

Extra PCM waves

The optional EXK-W expansion kit may be used to upgrade the Wavestation to include an extra 2 megabytes of PCM sound ROM, for double the amount of the original keyboard. The Wavestation A/D also comes standard with this expanded PCM. This means that they have a large number of additional PCM waveforms which are not included in the keyboard; any wave numbered over 396 is part of this expanded PCM. When transferring a Patch which uses these waves to a pre-3.0 Wavestation keyboard, the wave numbers will be "clipped" to 396 (Pulse31), and that Patch will therefore not sound the same. On a 3.0 or later Wavestation keyboard without the PCM expansion, the appropriate number will be shown, but the name will be shown as NO EXP and the wave will not sound. If you really want to play a sound which uses the expanded PCM on an unexpanded Wavestation keyboard, you will have to re-create the sound using waves numbered 396 or below.

When a Wave Sequence using expanded waves is transferred to an unexpanded 3.0 or later Wavestation, the correct wave numbers will be shown, but steps using expanded PCM will be shown as NO EXP PCM and will not sound. On pre-3.0 keyboards, however, the results are somewhat different. The steps with expanded waves play PCM from the normal, non-expanded ROM (the number of the wave played is equal to the number of the original wave minus 365). These steps are also transposed up 6 octaves, so it's easy to hear the change. Again, if you want to play such a Wave Sequence from an unexpanded Wavestation keyboard, you'll have to re-program it using waves from the non-expanded PCM.

SYSTEM EXCLUSIVE FORMAT

This system exclusive format contains data for both the Wavestation keyboard and Wavestation A/D rack-mount models. Data used by the Wavestation A/D but not referenced by the Wavestation keyboard is printed in **bold** type.

1.0 Header Format

The following is a description of the Wavestation system exclusive header. This format is common for all Wavestation system exclusive messages.

These bytes are excluded from the computation of the checksum.

```
11110000 (F0) System Exclusive status byte
01000010 (42) Korg ID
0011nnnn (3n) Format ID, n = channel number
00101000 (28) Wavestation device ID
0mmmmmmm      Message type
```

1.1 Message Type Codes

The following table contains a list of the message types in hex.

41	Parameter Change Message
42	Parameter Change Message Expanded
40	Single Patch Dump
49	Single Performance Dump
4C	All Patch Dump (within bank)
4D	All Performance Dump (within bank)
50	All Data (system, patch, performance, wave sequence) Dump
51	System Setup Dump
54	All Wave Sequence Dump
5A	Micro Tune Scales Dump
5C	System Setup Dump Expanded
55	Multi Mode Setup Dump
5D	Performance Map Dump
23	Data Load Completed
24	Data Load Error
11	Patch Write Command
1A	Performance Write Command
21	Write Complete Message
22	Write Error Message
5B	Multi Mode Setup Select

06	Multi Mode Setup Dump Request
07	Performance Map Dump Request
08	Micro Tune Scales Dump Request
0C	Wave Sequence Data Dump Request
0E	System Setup Dump Request
0F	All Data Dump Request
10	Single Patch Dump Request
19	Single Performance Dump Request
1C	All Patch Dump Request
1D	All Performance Dump Request

1.2 Binary data format

All 8 bit binary data is transmitted as two bytes in the following format:

0000LLLL	Low 4 bits of the data
0000HHHH	High 4 bits of the data

So that a byte is reconstructed as follows:

HHHHLLLL

This is referred to as Nibble data.

2.0 Transmit and Receive Messages

The following messages are both transmitted from the Wavestation and received by the Wavestation.

2.1 Data Messages

2.1.1 Single Patch Data

The following message contains a dump of a single patch. On reception the patch is placed in the edit buffer. To transfer a patch to a RAM location use the patch write command.

F0 42 3n 28	Wavestation sysex header
01000000 (40)	Single Patch Dump
000000xx (0x)	Bank number (0..3)
0xxxxxxx	Patch number.
Nibble data	Patch structure (section 5.2)
0ccccccc	Checksum
11110111 (F7)	End of exclusive.

2.1.2 Single Performance Data

The following message contains a dump of a single performance. On reception the performance is placed in the edit buffer. To place the performance in memory use the performance write command.

F0 42 3n 28	Wavestation sysex header
01001001 (49)	Single Performance Dump
000000xx (0x)	Bank number (0..3)
0xxxxxxx	Performance number
Nibble data	Performance structure (section 5.1)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.3 All Patch Data

This message contains all 35 patches within the bank specified.

F0 42 3n 28	Wavestation sysex header
01001100 (4C)	All Patch Dump
000000xx (0x)	Bank number (0..3)
Nibble data	35 patch structures (section 5.2)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.4 All Performance Data

This message contains all 50 performances within the bank specified.

F0 42 3n 28	Wavestation sysex header
01001101 (4D)	All Performance Dump
000000xx (0x)	Bank number (0..3)
Nibble data	50 performance structures (section 5.1)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.5 System Setup Parameter Data

This message is always accompanied by the System Setup Expanded data (as described below).

F0 42 3n 28	Wavestation sysex header
01010001 (51)	System Setup Dump
Nibble data	System structure (section 5.6)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.6 System Setup Parameter Expanded Data

This message always accompanies the System Setup Data (as described above).

F0 42 3n 28	Wavestation sysex header
01011100 (5C)	System Setup Expanded Dump
Nibble data	System Expanded structure (section 5.7)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.7 Wave Sequence Data

F0 42 3n 28	Wavestation sysex header
01010100 (54)	Wave Sequence Dump
000000xx (0x)	Bank number (0..3)
Nibble data	Ws_block structure (section 5.4)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.8 Multi Mode Setup Data

F0 42 3n 28	Wavestation sysex header
01010101 (55)	Multi Mode Setup Dump
Nibble data	Multiset_block structure (section 5.3)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.9 Performance Map Data

F0 42 3n 28	Wavestation sysex header
01011101 (5D)	Performance Map Dump
Nibble data	Performance Map_block structure (section 5.8)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.10 Micro Tune Scale Data

F0 42 3n 28	Wavestation sysex header
01011010 (5A)	Micro Tune Scale Dump
Nibble data	Mtune_block structure (section 5.5)
0ccccccc	Checksum
11110111 (F7)	End of exclusive

2.1.11 All Data

This message is always accompanied by the System Setup Expanded Data.

F0 42 3n 28	Wavestation sysex header
01010000 (50)	All Data Dump
Nibble data	All_data structure (section 5.9)
0ccccccc	Checksum
11110111 (F7)	End Of exclusive

2.1.12 Parameter Change Messages

The normal parameter change messages include parameters numbered up to 379.

F0 42 3n 28	Wavestation sysex header
01000001 (41)	Parameter Change Message
0LLLLLLL	LSB of parameter number (section 5.10)
0HHHHHHH	MSB of parameter number
0xxxxxxx	Parameter value in 7 bit ASCII (16 characters max) (7F = space)
.	
.	
00000000 (00)	ASCII null termination
11110111 (F7)	End of exclusive

2.1.13 Parameter Change Messages Expanded

The expanded parameter change messages include parameters numbered 380 and greater. They are otherwise completely the same as the normal parameter change messages.

F0 42 3n 28	Wavestation sysex header
01000010 (42)	Parameter Change Message Expanded
0LLLLLLL	LSB of parameter number (section 5.10)
0HHHHHHH	MSB of parameter number
0xxxxxxx	Parameter value in 7 bit ASCII (16 characters max) (7F = space)
.	
.	
00000000 (00)	ASCII null termination
11110111 (F7)	End of exclusive

2.1.14 Multi-Mode Setup Select

Sent whenever the current multi set is changed. On reception it will change the current multi setup.

F0 42 3n 28	Wavestation sysex header
01011011 (5B)	Multi Mode Setup Select
0xxxxxxx	Multi Mode Setup number
11110111 (F7)	End of exclusive

2.2 Status messages

Status messages are transmitted after reception of data messages. They indicate the receive status of the data. When received they will display an appropriate message.

2.2.1 Data Load Error

This message is transmitted whenever a message is received and the checksum failed.

F0 42 3n 28	Wavestation sysex header
00100100 (24)	Data Load Error message type
11110111 (F7)	End of exclusive

2.2.2 Data Load Complete

This message is transmitted whenever a data message is received successfully.

F0 42 3n 28	Wavestation sysex header
00100011 (23)	Data Load Complete message type
11110111 (F7)	End of exclusive

3.0 Transmit Only Messages

3.1 Status messages

Status messages are transmitted after reception of data messages. They indicate the receive status of the data.

3.1.1 Write Complete

F0 42 3n 28	Wavestation sysex header
00100001 (21)	Write Complete message type
11110111 (F7)	End of exclusive

3.1.2 Write Error

F0 42 3n 28	Wavestation sysex header
00100010 (22)	Write Error message type
11110111 (F7)	End of exclusive

3.2 Device ID Message

11110000 (F0)	System Exclusive
01111110 (7E)	Non Real Time message
0000xxxx (0X)	Channel number
00000110 (06)	Inquiry message
00000010 (02)	ID reply
01000010 (42)	KORG ID
00101000 (28)	Wavestation family code (LSB)
00000000 (00)	(MSB)

```
00000000 (00)          Member code (LSB)
00000000 (00)          (MSB)
0xxxxxxx (0x)          Minor software version (LSB)
0xxxxxxx (0x)          (MSB)
0xxxxxxx (0x)          Major software version(LSB)
0xxxxxxx (0x)          (MSB)
11110111 (F7)          End of exclusive
```

4.0 Receive Only Messages

4.1 Request Messages

4.1.1 Single Patch Dump Request

```
F0 42 3n 28            Wavestation sysex header
00010000 (10)          Single Patch Dump Request
000000xx (0x)          Bank number (0..3)
0xxxxxxx               Patch number
11110111 (F7)          End of exclusive
```

4.1.2 Single Performance Dump Request

```
F0 42 3n 28            Wavestation sysex header
00011001 (19)          Single Performance Dump Request
000000xx (0x)          Bank number (0..3)
0xxxxxxx               Performance number
11110111 (F7)          End of exclusive
```

4.1.3 All Patch Dump Request

```
F0 42 3n 28            Wavestation sysex header
00011100 (1c)          All Patch Dump Request
000000xx (0x)          Bank number (0..3)
11110111 (F7)          End of exclusive
```

4.1.4 All Performance Dump Request

```
F0 42 3n 28            Wavestation sysex header
00011101 (1d)          All Performance Dump Request
000000xx (0x)          Bank number (0..3)
11110111 (F7)          End of exclusive
```

4.1.5 All Data Request

```
F0 42 3n 28            Wavestation sysex header
00001111 (0F)          All Data Dump Request
11110111 (F7)          End of exclusive
```

4.1.6 System Setup Dump Request

```
F0 42 3n 28            Wavestation sysex header
00001110 (0E)          System Setup Dump Request
11110111 (F7)          End of exclusive
```

4.1.7 Wave Sequence Data Dump Request

F0 42 3n 28	Wavestation sysex header
00001100 (0C)	Wave Sequence Dump Request
000000xx (0x)	Bank number (0..3)
11110111 (F7)	End of exclusive

4.1.8 Performance Map Dump Request

F0 42 3n 28	Wavestation sysex header
00000111 (07)	Performance Map Dump Request
11110111 (F7)	End of exclusive

4.1.9 Multi Mode Setup Dump Request

F0 42 3n 28	Wavestation sysex header
00000110 (06)	Multi Mode Setup Dump Request
11110111 (F7)	End of exclusive

4.1.10 Micro Tune Scales Dump Request

F0 42 3n 28	Wavestation sysex header
00001000 (08)	Micro Tune Scales Dump Request
11110111 (F7)	End of exclusive

4.2 Commands

4.2.1 Patch Write Command

F0 42 3n 28	Wavestation sysex header
00010001 (11)	Patch Write Command
000000xx (0x)	Bank number (0..3)
0ppppppp (pp)	Patch number (0-34)
11110111 (F7)	End of exclusive

4.2.2 Performance Write Command

F0 42 3n 28	Wavestation sysex header
00011010 (1A)	Performance Write Command
000000xx (0x)	Bank number (0..3)
0ppppppp (pp)	Performance number (0-49)
11110111 (F7)	End of exclusive

4.3 Device Inquiry Message

11110000 (F0)	System Exclusive
01111110 (7E)	Non Real Time
0xxxxxxx (0x)	Channel number
00000110 (06)	Inquiry message
00000001 (01)	Inquiry request
11110111 (F7)	End of exclusive

5.0 Data Structure Tables

```
typedef   char            byte;     /* 8 bits, signed */
typedef   short           word;     /* 16 bits, signed */
typedef   unsigned char   ubyte;    /* 8 bits, unsigned */
typedef   unsigned short  uword;    /* 16 bits, unsigned */
typedef   unsigned long   ulong;    /* 32 bits, unsigned */
typedef   unsigned char   boolean;  /* Boolean TRUE or FALSE */
```

5.1 Performance Data Structure

```
typedef struct
{
   char Perf_Name[NAME_SIZE];/* Performance name - up to 16
                             characters */
   byte Fx_Perf_Block[21];   /* Leave space for effects
                             parameters */
   part Parts[8];            /* This is where the PART
                             blocks start, of which 8 can be
                             appended to the performance */
} performance;

typedef struct
{
   byte      Bank_Num;       /* Bank number this PART is playing */
   byte      Patch_Num;      /* Patch number this PART is playing */
   ubyte     Level;          /* Volume for this part */
   byte      Output;         /* OUTPUT CHAN FOR THIS Part
                              (-1 = stereo) */
   ubyte     Part_Mode;      /* KEYBOARD ASSIGN MODE
                             (Polyphonic,UNI) */
                             /* bit 6 */
                             /* 1 = Patch is from Expansion RAM Bank
                             (RAM3) */

                             /* bit 5-4 */
                             /* 00= **** */
                             /* 01= Local play mode*/
                             /* 10= MIDI play mode*/
                             /* 11 = Both */

                             /* bit 3-2 */
                             /* 00= **** */
                             /* 01= polyphonic*/
                             /* 10= unison re-trigger*/
                             /* 11= unison legato*/

                             /* bit 1-0 */
                             /* 00= low note*/
                             /* 01= high note*/
                             /* 10= last note*/
                             /* 11 = **** */
```

```
    ubyte     Lo_Key;          /* Lower note of keyboard range*/
    ubyte     Hi_Key;          /* Upper note of keyboard range*/
    ubyte     Lo_Vel;          /* Lower limit of velocity range*/
    ubyte     Hi_Vel;          /* Upper limit of velocity range */
    byte      Trans;           /* Transpose value in semitones */
    byte      Detune;          /* Detune value in cents*/
    ubyte     Tunetab;         /* Micro tuning table for this PART */
    ubyte     Micro_Tune_Key;  /* Root key for pure major/minor and
                               USER scales */
    ubyte     Midi_Out_Chan;   /* MIDI transmit channel for this
                               PART */
    byte      Midi_Prog_Num;   /* MIDI prog# to xmit when PART
                               selected, -1 =off) */
    byte      Sus_Enable;      /* Sustain Pedal enable/disable */
    uword     Delay;           /* Delay value in milliseconds */
} part;
```

5.2 Patch Data Structure

```
/*        Individual Patch Data Structure */
/* This is the structure for data that is individual to the */
/* 1, 2, or 4 oscillators that make up a Patch.*/
/* Four of these structures are included in a Patch.*/

typedef struct
{
    byte      Wave_Coarse;     /* Wave detuning in semitones*/
    byte      Wave_Fine;       /* Wave detuning in cents */
    ubyte     Wave_Bank;       /* Wave bank */
    uword     Wave_Num;        /* Wave number*/
    byte      Wave_Scale;      /* Wave pitch scaling slope */
    ubyte     Lfo1_Rate;       /* LFO 1 Rate */
    ubyte     Lfo1_Amt;        /* LFO 1 Amount*/
    ubyte     Lfo1_Delay;      /* LFO 1 Delay*/
    ubyte     Lfo1_Fade;       /* LFO 1 Fade in*/
    ubyte     Lfo1_Shape;      /* LFO 1 Shape (bits 0-6)1-127*/
                               /* LFO 1 Sync (bit 7) */
                               /*1 = Sync on */
                               /*0 = Sync off*/
    byte      S1_Lfo1_R;       /* Mod Source to LFO 1 Rate pointer*/
    byte      S1_Lfo1_R_Amt;   /* Mod Source to LFO 1 Rate amount*/
    byte      S1_Lfo1_A;       /* Mod Source to LFO 1 Amt pointer*/
    byte      S1_Lfo1_A_Amt;   /* Mod Source to LFO 1 Amt amount*/
    ubyte     Lfo2_Rate;       /* LFO 2 Rate*/
    ubyte     Lfo2_Amt;        /* LFO 2 Amount*/
    ubyte     Lfo2_Delay;      /* LFO 2-Delay*/
    ubyte     Lfo2_Fade;       /* LFO 2-Fade in*/
    ubyte     Lfo2_Shape;      /* LFO 2-Shape (bits 0-6)1-127*/
                               /* LFO 2 Sync (bit 7) */
                               /* 1 = Sync on */
                               /* 0 = Sync off*/
    byte      S1_Lfo2_R;       /* Mod Source to LFO 1 Rate pointer*/
    byte      S1_Lfo2_R Amt;   /* Mod Source to LFO 2 Rate amount*/
    byte      S1_Lfo2_A;       /* Mod Source to LF0 2 Amt pointer*/
```

```
byte      S1_Lfo2_A Amt;  /* Mod Source to LFO 1 Amt amount*/
ubyte     EG_Rate1;       /* Envelope 1 Rate 1 */
ubyte     EG_Rate2;       /* Envelope 1 Rate 2 */
ubyte     EG_Rate3;       /* Envelope 1 Rate 3 */
ubyte     EG_Rate4;       /* Envelope 1 Rate 4 */
ubyte     EG_Level0;      /* Envelope 1 Level 0 */
ubyte     EG_Level1;      /* Envelope 1 Level 1 */
ubyte     EG_Level2;      /* Envelope 1 Level 2 */
ubyte     EG_Level3;      /* Envelope 1 Level 3 */
ubyte     EG_Level4;      /* Envelope 1 Level 4 */
byte      Vel_EG_A;       /* Velocity to Env1 Amount Amt */
ubyte     AEG_Rate1;      /* Amplitude Envelope Rate 1 */
ubyte     AEG_Rate2;      /* Amplitude Envelope Rate 2 */
ubyte     AEG_Rate3;      /* Amplitude Envelope Rate 3 */
ubyte     AEG_Rate4;      /* Amplitude Envelope Rate 4 */
ubyte     AEG_Level0;     /* Amplitude Envelope Level 0 */
ubyte     AEG_Level1;     /* Amplitude Envelope Level 1 */
ubyte     AEG_Level2;     /* Amplitude Envelope Level 2 */
ubyte     AEG_Level3;     /* Amplitude Envelope Level 3 */
byte      Pitch_Mac;      /* Pitch Macro number*/
byte      Fil_Mac;        /* Filter Macro number*/
byte      Amp_Mac;        /* Amplitude Envelope Macro number*/
byte      Pan_Mac;        /* Pan Macro number*/
byte      Env_Mac;        /* Envelope 1 macro number*/
byte      Pw_Range;       /* Pitchwheel Range */
byte      S1_Pitch;       /* Modulation Source 1 to Pitch
                          pointer*/
byte      S1_Pitch_Amt;   /* Modulation Source 1 to Pitch
                          Amount*/
byte      S2_Pitch;       /* Modulation Source 2 to Pitch
                          pointer*/
byte      S2_Pitch_Amt;   /* Modulation Source 2 to Pitch
                          Amount*/
byte      Key_Filter;     /* Keyboard to Filter Cutoff Amount*/
byte      S1_Filter;      /* Modulation Source 1 to Filter
                          pointer*/
byte      S1_Filter_Amt;  /* Modulation Source 1 to Filter
                          Amount*/
byte      S2_Filter;      /* Modulation Source 2 to Filter
                          pointer*/
byte      S2_Filter_Amt;  /* Modulation Source 2 to Filter
                          Amount*/
byte      Vel_AEG_A;      /* Velocity to Amp Env Amount Amount*/
byte      Vel_AEG_R;    /* Velocity To Amp Env Attack Rate Amt*/
byte      Key AEG_R;    /* Keyboard to Amp Env Decay Rate Amt*/
byte      S1_Amp;       /* Modulation Source 1 to Amp pointer*/
byte      S1_Amp_Amt;   /* Modulation Source 1 to Amp Amount*/
byte      S2_Amp;       /* Modulation Source 2 to Amp pointer*/
byte      S2_Amp_Amt;   /* Modulation Source 2 to Amp Amount*/
byte      Key_Pan_Amt;  /* Keyboard to Pan Amount*/
byte      Vel_Pan_Amt;  /* Velocity to Pan Amount*/
ubyte     Cutoff;       /* Filter Cutoff value */
ubyte     Filter_Exciter;/* Filter Exciter value */
byte      Vel_EG_R;     /* Velocity to ENV1 rate amount*/
```

```
    byte      Key_EG_R;      /* Keyboard to ENV1 rate amount*/
    byte      PEG_Amt;       /* Pitch Ramp amount*/
    ubyte     PEG_Rate;      /* Pitch Ramp rate*/
    byte      Vel_PEG_A;     /* Velocity to pitch ramp amount */
    byte      Indiv_Level;   /* Velocity to pitch ramp rate amount*/
    long      Lfo1_Inc;      /* Lfo fade in amount increment*/
    long      Lfo2_Inc;      /* Lfo fade in amount increment*/
    byte      Patch_Output;  /* Individual output routing */
    byte      Wave_Num_Exp;  /* Wave number expansion to access
                             Expansion PCM data (Waves numbered
                             397 and over), if present. This number
                             is added to the value of Wave_Num to
                             determine the actual wave number.*/
} indiv;

/* Patch data structure*/

typedef struct
{
    char      Patch Name[16];/* Patch name up to 16 characters*/
    ubyte     Mix_Rate1;     /* Mix envelope rate for segment 1 */
    ubyte     Mix_Rate2;     /* Mix envelope rate for segment 2 */
    ubyte     Mix_Rate3;     /* Mix envelope rate for segment 3 */
    ubyte     Mix_Rate4;     /* Mix envelope rate for segment 4 */
    uword     Mix_Count1;    /* Number of update cycles for env seg*/
    uword     Mix_Count2;    /* Number of update cycles for env seg*/
    uword     Mix_Count3;    /* Number of update cycles for env seg*/
    uword     Mix_Count3B;   /* Number of update cycles for env seg*/
    uword     Mix_Count2B;   /* Number of update cycles for env seg*/
    uword     Mix_Count1B;   /* Number of update cycles for env seg*/
    uword     Mix_Count4;    /* Number of update cycles for env seg*/
    long      Mix_XSlope1;   /* Increment size for env seg 1 */
    long      Mix_XSlope2;   /* Increment size for env seg 2 */
    long      Mix_XSlope3;   /* Increment size for env seg 3 */
    long      Mix_XSlope4;   /* Increment size for env seg 4 */
    long      Mix_YSlope1;   /* Increment size for env seg 1 */
    long      Mix_YSlope2;   /* Increment size for env seg 2 */]
    long      Mix_YSlope3;   /* Increment size for env seg 3 ~/
    long      Mix_YSlope4;   /* Increment size for env seg 4 */
    ubyte     Mix_X0;        /* Mix Envelope Point 0 level */
    ubyte     Mix_X1;        /* Mix Envelope Point 1 level */
    ubyte     Mix_X2;        /* Mix Envelope Point 2 level */
    ubyte     Mix_X3;        /* Mix Envelope Point 3 level */
    ubyte     Mix_X4;        /* Mix Envelope Point 4 level */
    ubyte     Mix_Y0;        /* Mix Envelope Point 0 level */
    ubyte   Mix_Y1;          /* Mix Envelope Point 1 level */
    ubyte   Mix_Y2;          /* Mix Envelope Point 2 level */
    ubyte   Mix_Y3;          /* Mix Envelope Point 3 level */
    ubyte   Mix_Y4;          /* Mix Envelope Point 4 level */
    ubyte   Mix_Repeats;     /* Number of repeats of mix envelope*/
    ubyte   Mix_Env_Loop;    /* Start segment of Mix Envelope loops*/
    ubyte   S1_MixAC;        /* Modulation Source 1 to MixAC
                             pointer*/
    byte    S1_MixAC_Amt;    /* Modulation Source 1 to MixAC Amount*/
```

```
    ubyte   S2_MixAC;          /* Modulation Source 2 to MixAC
                               pointer*/
    byte    S2_MixAC_Amt;      /* Modulation Source 2 to MixAC Amount*/
    ubyte   S1_MixBD;          /* Modulation Source 1 to MixBD
                               pointer*/
    byte    S1_MixBD_Amt;      /* Modulation Source 1 to MixBD Amount*/
    ubyte   S2_MixBD;          /* Modulation Source 2 to MixBD
                               pointer*/
    byte    S2_MixBD_Amt;      /* Modulation Source 2 to MixBD Amount*/
    byte    Number_Of_Waves;   /* Number of WAVES/WAVESEQS in Patch*/
    ubyte   Hard_Sync;         /* Hard Sync Flag*/
    byte    Bank_Exp;          /* Bit 3 = 1; Wave D uses RAM3 waveseq */
                               /* Bit 2 = 1; Wave C uses RAM3 waveseq */
                               /* Bit 1 = 1; Wave B uses RAM3 waveseq */
                               /* Bit 0 = 1; Wave A uses RAM3 waveseq */
    byte    Dummy141;          /* Extra for future use */
    indiv   waveA;             /* Individual parameters for WAVE A */
    indiv   waveB;             /* Individual parameters for WAVE B */
    indiv   waveC;             /* Individual parameters for WAVE C */
    indiv   waveD;             /* Individual parameters for WAVE D */
} patch;
```

5.3 Multi Mode Setup Data Structure

This contains the data for the Multimode Setups.

```
/*          Data structures of the multi-set map which*/
/*          specifies the initial program on each track.*/
/*          There are 16 setups. Each one holds bank/prog */
/*          numbers for each MIDI channel. */

typedef struct
{
   ubyte   Multimap_Chan_Enable;/* MIDI channel enable/disable */
   ubyte   Multimap_Bank;   /* Bank number of this program */
   ubyte   Multimap_Prog;   /* Program number of this program */
   ubyte   Multimap_Level;  /* Performance level */
} multimap;

typedef struct
{
   ubyte   Multiset_FX_Chan;/* Effects control channel number*/
   ubyte   Fx_Multi_Block[21];/* Space for effects parameters*/
   multimap Multiset_Map[16];/* Bank and program numbers */
} multiset;

typedef struct
{
   multiset                multisets[16];
   byte                    spare_multiset_byte;
} multiset_block;
```

5.4 Wave Sequence Data Structure

```
/* This is repeated for the number of wave sequences in the bank. */

typedef struct
{
   uword   WS_Link;        /* Pointer to Wave Sequence Start Step */
   uword   WS_Slink;       /* Pointer to Startmod Start Step */
   ubyte   WS_Loop_Start;  /* Step number of WAVESEQ Loop Start
                           Point
                           step*/
   ubyte   WS_Loop_End;    /* Step number of WAVESEQ Loop End Point
                           step*/
   ubyte   WS_Loop_Count;  /* - Loop repeat count
                           (bits 0-6)1-127*/
                           /* O=OFF      */
                           /~ 127=1NF */
                           /*Loop Direction (bit 7)*/
                           /* O = FOR */
                           /* 1 = B/F       */
   ubyte   WS_Start_Step,  /* Startmod starting step number*/
   ubyte   WS_Mod_Src;     /* Controller number to use for
                           startmod */
   byte    WS_Mod_Amt;     /* Startmod sensitivity */
   word    WS_Dyno_Mod;    /* (Total_Time * Mod_Amt)/255 */
   uword   WS_Start_Time;  /* Cumulative time up to start step */
   uword   WS_Time;        /* Total time of Wave Sequence */
} waveseq;

/* Data structure of each STEP in a WAVE SEQUENCE  */

typedef struct
{
   uword   WS_Flink;       /* Step number of step in WAVSEQ after
                           this one */
   uword   WS_Blink;       /* Step number of step in WAVSEQ before
                            this one */
   uword   WS_Llink;       /* Pointer to loop start (0xFFFF except
                           last step) */
   uword   WS_Wave_Num;    /* Wave number of this step in wave
                           sequence */
   byte    WS_Coarse       /* -24 to 24: Coarse tuning of wave */
                           25 to 47: illegal values
                           48 to 96: subtract 72 for actual coarse
                           tuning and use expanded PCM, if present,
                           adding 365 to WS_Wave_Num value for actual
                           PCM wave number. */
   byte    WS_Fine;        /* Fine tuning of wave */
   uword   WS_Xfade;       /* Crossfade time of wave */
   uword   WS_Duration;    /* Duration of wave */
   ubyte   WS_Level;       /* Level of wave */
   ubyte   WS_Mod_Index;   /* Modulation Index */
} wavestep;
```

```
typedef struct
{
   char    Wave_Seq_Name[8];
} ws_name;

/* This is the entire structure which is transmitted */

typedef struct
{
   waveseq   waveseq_block[32];/* 32 wavseq locations */
   wavstep   wavstep_block[501];/* 501 wave seq steps */
   ws_name   ws_name_block[32];/* 32 wave seq names */
} ws_block;
```

5.5 Micro Tune Scale Data Structures

```
typedef struct
{
   byte      c key;      /* Offset from equal tempered for C note */
   byte      cs key;     /* Offset from equal tempered for C# note */
   byte      d key;      /* Offset from equal tempered for D note */
   byte      ds key;     /* Offset from equal tempered for D# note */
   byte      e key;      /* Offset from equal tempered for E note */
   byte      f key;      /* Offset from equal tempered for F note */
   byte      fs key;     /* Offset from equal tempered for F# note */
   byte      g key;      /* Offset from equal tempered for G note */
   byte      gs key;     /* Offset from equal tempered for G# note */
   byte      a key;      /* Offset from equal tempered for A note */
   byte      as key;     /* Offset from equal tempered for A# note */
   byte      b key;      /* Offset from equal tempered for B note */
} mtune;

typedef struct
{
   mtune     mtunes[12];
   byte      spare_mtune_byte;
} mtune_block;
```

5.6 System Setup Data Structure

```
typedef struct
{
   ubyte   current_multi;          /* CURRENT MULTISET */
   ubyte   current_tune;           /* CURRENT _MTUNE* /
   byte    master_tune;            /* MASTER TUNE */
   byte    effects_enable;         /* EFFECTS ENABLE */
   ubyte   pitch_bend_range;       /* PITCH BEND RANGE */
   ubyte   velocity_response;      /* VELOClTY RESPONSE*/
   byte    midi_mode;              /* MIDI MODE */
   ubyte   midi_base;              /* MIDI BASE CHAN */
   ubyte   num_mono_chans;         /* NUM MONO CHANS */
   byte    key_num_offset;         /* KEY NUM OFFSET */
   byte    param_enable;           /* MIDI PARAM ENABLE */
```

```
    byte    midi_1;                 /* CONTROLLER 1 */
    byte    midi_2;                 /* CONTROLLER 2*/
    byte    xmit_mode;              /* XMIT MODE */
    byte    local_kd;               /* LOCAL_KBD */
    byte    xmit_program_enable;    /* XMlT PROG CHANGE */
    byte    xmit_pressure_enable;   /* XMIT AFTERTOUCH */
    byte    xmit_pitch_enable;      /* XMlT PITCH BEND */
    byte    xmit_control_enable;    /* XMlT CONTROLLERS*/
    byte    rec_program_enable;     /* REC PROG CHANGE */
    byte    rec_pressure_enable;    /* REC AFTERTOUCH */
    byte    rec_pitch_enable        /* REC PITCH BEND */
    byte    rec_control_enable;     /* REC CONTROLLERS*/
    byte    note_enable;            /* REC NOTE ON OFF*/
    byte    alloff_enable;          /* REC ALL NOTES OFF*/
    byte    progmap_enable;         /* PROGMAP ENABLE */
    ubyte   foot_damper_function;
    ubyte   foot_damper_polarity;
    ubyte   foot_assign_1_function;
    ubyte   foot_assign_1_polarity;
    ubyte   foot_assign_2_function;
    ubyte   foot_assign_2_polarity;
    ubyte   ws_midi_clock;
    byte    spare_system_byte;
} system;
```

5.7 System Setup Expanded Data Structure

This contains the local transpose amount, as well as data used by the Wavestation A/D Rack module.

```
typedef struct
{
   ubyte                prog_to_multi_fx;
   ubyte                change_multi_with;
   ubyte                remap_to_joy_x;
   ubyte                remap_to_joy_y;
   ubyte                remap_to_fx_switch;
   ubyte                local_xpose;
   ubyte                analog_setup_number;
   byte                 analog_bus_macro;
   ubyte                analog_lev_1;
   ubyte                analog_lev_2;
   byte                 analog_chan_1;
   byte                 analog_chan_2;
   ubyte                analog_1_bus;
   ubyte                analog_2_bus;
   ubyte                analog_1_filter;
   ubyte                analog_2_filter;
   ubyte                analog_1_exciter;
   ubyte                analog_2_exciter;
   ubyte                analog_input_disable;
}system_ext;
```

5.8 Performance Map Structures

This contains the data for the Performance Select Map.

```
typedef struct
{
   ubyte     Perfmap_Bank;   /* Bank number of this performance */
   ubyte     Perfmap_Prog;   /* MIDI Program Change number of this
                             performance */
} perfmap;

typedef struct
{
   perfmap   perfmaps[128];
   byte      spare_perfmap_byte;
} perfmap_block;
```

5.9 All Data Structure

```
typedef struct
{
   system                    system_all;
   multiset_block            multiset_all;
   mtune_block               mtune_all;
   perfmap_block             perfmap_all;
   performance               perf_ram1[50];
   performance               perf_ram2[50];
   patch                     patch_ram1[35];
   patch                     patch_ram2[35];
   ws_block                  ws_ram1;
   ws_block                  ws_ram2;
} all_data;
```

5.10 Parameter Number Table

```
enum /* Parameter numbers. */

{
/* 0 */   CURRENT_BANK,
/* 1 */   CARD_NAME,
/* 2 */   CURRENT_PROG,
/* 3 */   PROG_NAME,
/* 4 */   MIDI_MODE,
/* 5 */   MIDI_BASE_CHAN,
/* 6 */   NUM_MONO_CHANS,
/* 7 */   KEY_NUM_OFFSET,
/* 8 */   MIDI_PARAM_ENABLE,
/* 9 */   CONTROLLER_1,
/* 10 */   CONTROLLER_2,
/* 11 */   XMIT_MODE,
/* 12 */   LOCAL_KBD,
/* 13 */   XMIT_PROG_CHANGE,
/* 14 */   XMIT_AFTERTOUCH,
/* 15 */   XMIT_PITCH_BEND,
/* 16 */   XMIT_CONTROLLERS,
/* 17 */   REC_PROG_CHANGE,
/* 18 */   REC_AFTERTOUCH,
/* 19 */   REC_PITCH_BEND,
/* 20 */   REC_CONTROLLERS,
/* 21 */   REC_NOTE_ON_OFF,
/* 22 */   REC_ALL_NOTES_OFF,
/* 23 */   PROGMAP_ENABLE,
/* 24 */   PROGMAP_CHANGE_NUM,
/* 25 */   PROGMAP_PROG_BANK,
/* 26 */   PROGMAP_PROG_NUM,
/* 27 */   PROGMAP_PROG_NAME,
/* 28 */   CURRENT_MULTISET,
/* 29 */   MULTISET_FX_CONTROL_CHAN,
/* 30 */   MULTISET_CHAN,
/* 31 */   MULTISET_CHAN_ENABLE,
/* 32 */   MULTISET_LEVEL,
/* 33 */   MULTISET_PROG_BANK,
/* 34 */   MULTISET_PROG_NUM,
/* 35 */   MULTISET_PROG_NAME,
/* 36 */   SYSEX_PATCH_BANK,
/* 37 */   SYSEX_PATCH_NUM,
/* 38 */   SYSEX_ALL_BANK,
/* 39 */   SYSEX_WAVESEQ_BANK,
/* 40 */   SYSEX_PROG_BANK,
/* 41 */   SYSEX_PROG_NUM,
/* 42 */   MASTER_TUNE,
/* 43 */   EFFECTS_ENABLE,
/* 44 */   MEM_PROTECT_INTERNAL,.
/* 45 */   MEM_PROTECT_CARD,
/* 46 */   PITCH_BEND_RANGE,
/* 47 */   VELOCITY_RESPONSE,
/* 48 */   SAVE_DATA_TYPE,
/* 49 */   SAVE_SOURCE_BANK,
/* 50 */   SAVE_SOURCE_NUM,
/* 51 */   SAVE_SOURCE_NAME,
/* 52 */   SAVE_DEST_BANK,
/* 53 */   SAVE_DEST_NUM,
/* 54 */   SAVE_DEST_NAME,
/* 55 */   SAVE_PLAY,
/* 56 */   CURRENT_PART,
/* 57 */   PART_PATCH_BANK,
/* 58 */   PART_PATCH_NUM,
/* 59 */   PART_PATCH_NAME,
/* 60 */   PART_MODE,
/* 61 */   PART_VOLUME,
/* 62 */   PART_OUTPUT,
/* 63 */   PART_KEY_LIMIT_LOW,
/* 64 */   PART_KEY_LIMIT_HIGH,
/* 65 */   PART_VEL_LIMIT_LOW,
/* 66 */   PART_VEL_LIMIT_HIGH,
/* 67 */   PART_TRANSPOSE,
/* 68 */   PART_DETUNE,
/* 69 */   PART_SUS_ENABLE,
/* 70 */   PART_DELAY,
/* 71 */   PART_UNI_NOTE_PRIORITY,
/* 72 */   PART_MTUNE_TAB,
/* 73 */   PART_MTUNE_KEY,
/* 74 */   PART_MIDI_XMIT_CHAN,
/* 75 */   PART_PLAY_MODE,
/* 76 */   PART_PROG_CHANGE_XMIT,
/* 77 */   PATCH_STRUCTURE,
/* 78 */   PATCH_HARD_SYNC,
/* 79 */   CURRENT_WAVE,
/* 80 */   PATCH_PITCH_MACRO,
/* 81 */   PATCH_FILTER_MACRO,
/* 82 */   PATCH_AMP_MACRO,
/* 83 */   PATCH_PAN_MACRO,
/* 84 */   PATCH_ENV_MACRO,
/* 85 */   PATCH_PITCH_BEND_RANGE,
/* 86 */   PATCH_PITCH_RAMP_AMT,
/* 87 */   PATCH_PITCH_RAMP_RATE,
/* 88 */   PATCH_PITCH_VEL_AMT,
/* 89 */   PITCH_SOURCE_1,
/* 90 */   PITCH_SOURCE_1_AMOUNT,
/* 91 */   PITCH_SOURCE_2,
/* 92 */   PITCH_SOURCE_2_AMOUNT,
/* 93 */   FILTER_MOD_CUTOFF,
/* 94 */   FILTER_MOD_TRACKING,
/* 95 */   FILTER_EXCITER_AMOUNT,
/* 96 */   FILTER_MOD_SOURCE1,
/* 97 */   FILTER_MOD_SOURCE1_AMT,
/* 98 */   FILTER_MOD_SOURCE2,
/* 99 */   FILTER_MOD_SOURCE2_AMT,
/* 100 */   GP_ENV_LEVEL_0,
/* 101 */   GP_ENV_LEVEL_1,
/* 102 */   GP_ENV_LEVEL_2,
/* 103 */   GP_ENV_LEVEL_3,
/* 104 */   GP_ENV_LEVEL_4,
/* 105 */   GP_ENV_RATE_1,
/* 106 */   GP_ENV_RATE_2,
/* 107 */   GP_ENV_RATE_3,
/* 108 */   GP_ENV_RATE_4,
/* 109 */   GP_VEL_ENV_AMT,
/* 110 */   AMP_ENV_LEVEL_0,
/* 111 */   AMP_ENV_LEVEL_1,
```

```
/* 112 */   AMP_ENV_LEVEL_2,
/* 113 */   AMP_ENV_LEVEL_3,
/* 114 */   AMP_ENV_RATE_1,
/* 115 */   AMP_ENV_RATE_2,
/* 116 */   AMP_ENV_RATE_3,
/* 117 */   AMP_ENV_RATE_4,
/* 118 */   AMP_MOD_VEL_ENV_ AMOUNT,
/* 119 */   AMP_MOD_SOURCE_1,
/* 120 */   AMP_MOD_SOURCE_1_AMOUNT,
/* 121 */   AMP_MOD_SOURCE_2,
/* 122 */   AMP_MOD_SOURCE_2_AMOUNT,
/* 123 */   AMP_MOD_VEL_ATTACK_RATE,
/* 124 */   AMP_MOD_KBD_DECAY_RATE,
/* 125 */   LFO1_RATE,
/* 126 */   LFO1_INITIAL_AMOUNT,
/* 127 */   LFO1_SHAPE,
/* 128 */   LFO1_SYNC,
/* 129 */   LFO1_DELAY,
/* 130 */   LFO1_FADE_IN,
/* 131 */   LFO1_DEPTH_MOD_SOURCE,
/* 132 */   LFO1_DEPTH_MOD_SRC_AMT,
/* 133 */   LFO1_RATE_MOD_SOURCE,
/* 134 */   LFO1_RATE_MOD_SRC_AMT,
/* 135 */   LFO2_RATE,
/* 136 */   LFO2_INITIAL_AMOUNT,
/* 137 */   LFO2_SHAPE,
/* 138 */   LFO2_SYNC,
/* 139 */   LFO2_DELAY,
/* 140 */   LFO2_FADE_IN,
/* 141 */   LFO2_DEPTH_MOD_SOURCE,
/* 142 */   LFO2_DEPTH_MOD_SRC_AMT,
/* 143 */   LFO2_RATE_MOD_SOURCE,
/* 144 */   LFO2_RATE_MOD_SRC_AMT,
/* 145 */   PAN_VELOCITY_AMOUNT,
/* 146 */   PAN_KEYBOARD_AMOUNT,
/* 147 */   WAVEA_BANK,
/* 148 */   WAVEA_NUM,
/* 149 */   WAVEA_NAME,
/* 150 */   WAVEA_LEVEL,
/* 151 */   WAVEA_TUNE_COARSE,
/* 152 */   WAVEA_TUNE_FINE,
/* 153 */   WAVEA_TUNE_SLOPE,
/* 154 */   WAVEB_BANK,
/* 155 */   WAVEB_NUM,
/* 156 */   WAVEB_NAME,
/* 157 */   WAVEB_LEVEL,
/* 158 */   WAVEB_TUNE_COARSE,
/* 159 */   WAVEB_TUNE_FINE,
/* 160 */   WAVEB_TUNE_SLOPE,
/* 161 */   WAVEC_BANK,
/* 162 */   WAVEC_NUM,
/* 163 */   WAVEC_NAME,
/* 164 */   WAVEC_LEVEL,
/* 165 */   WAVEC_TUNE_COARSE,
/* 166 */   WAVEC_TUNE_FINE,
/* 167 */   WAVEC_TUNE_SLOPE,
/* 168 */   WAVED_BANK,
/* 169 */   WAVED_NUM,
/* 170 */   WAVED_NAME,
/* 171 */   WAVED_LEVEL,
/* 172 */   WAVED_TUNE_COARSE,
/* 173 */   WAVED_TUNE_FINE,
/* 174 */   WAVED_TUNE_SLOPE,
/* 175 */   WAVE_SEQ_NUM,
/* 176 */   WAVE_SEQ_BANK,
/* 177 */   WAVE_SEQ_NAMB,
/* 178 */   WAVE_SEQ_STEP,
/* 179 */   WAVE_SEQ_WAVE_BANK,
/* 180 */   WAVE_SEQ_WAVE_NUM,
/* 181 */   WAVE_SEQ_WAVE_NAME,
/* 182 */   WAVE_SEQ_COARSE,
/* 183 */   WAVE_SEQ_FINE,
/* 184 */   WAVE_SEQ_LEVEL,
/* 185 */   WAVE_SEQ_DURATION,
/* 186 */   WAVE_SEQ_XFADE,
/* 187 */   WAVE_SEQ_LOOP_START,
/* 188 */   WAVE_SEQ_LOOP_END,
/* 189 */   WAVE_SEQ_REPEATS,
/* 190 */   WAVE_SEQ_START_STEP,
/* 191 */   WAVE_SEQ_MOD_SRC,
/* 192 */   WAVE_SEQ_MOD_AMT,
/* 193 */   MIX_ENV_POINT,
/* 194 */   MIX_ENV_RATE,
/* 195 */   MIX_ENV_X,
/* 196 */   MIX_ENV_Y,
/* 197 */   MIX_PERCENT_A,
/* 198 */   MIX_PERCENT_B,
/* 199 */   MIX_PERCENT_C,
/* 200 */   MIX_PERCENT_D,
/* 201 */   MIX_ENV_LOOP,
/* 202 */   MIX_ENV_REPEATS,
/* 203 */   MIX_MOD_X_SOURCE1,
/* 204 */   MIX_MOD_X_SRC1_AMT,
/* 205 */   MIX_MOD_X_SOURCE2,
/* 206 */   MIX_MOD_X_SRC2_AMT,
/* 207 */   MIX_MOD_Y_SOURCE1,
/* 208 */   MIX_MOD_Y_SRC1_AMT,
/* 209 */   MIX_MOD_Y_SOURCE2,
/* 210 */   MIX_MOD_Y_SRC2_AMT,
/* 211 */   COPY_MACRO_MODULE,
/* 212 */   COPY_MACRO_SOURCE_WAVE,
/* 213 */   COPY_MACRO_SOURCE_BANK,
/* 214 */   COPY_MACRO_SOURCE_NUM,
/* 215 */   COPY_MACRO_SOURCE_NAME,
/* 216 */   COPY_MACRO_DEST_ MODULE,
/* 217 */   COPY_MACRO_DEST_WAVE,
/* 218 */   COPY_MACRO_DEST_BANK,
/* 219 */   COPY_MACRO_DEST_NUM,
/* 220 */   COPY_MACRO_DEST_NAME,
/* 221 */   COPY_DEST_PART,
/* 222 */   COPY_DEST_PART_PATCH_BLANK,
/* 223 */   COPY_DEST_PART_PATCH_NUM,
/* 224 */   COPY_DEST_PART_PATCH_NAME,
/* 225 */   COPY_WS_SOURCE_FROM_STEP,
/* 226 */   COPY_WS_SOURCE_FROM_BANK,
/* 227 */   COPY_WS_SOURCE_FROM_NUM,
/* 228 */   COPY_WS_SOURCE_FROM_NAME,
/* 229 */   COPY_WS_SOURCE_TO_STEP,
```

```
/* 230 */   COPY_WS_SOURCE_TO_BANK,
/* 231 */   COPY_WS_SOURCE_TO_NUM,
/* 232 */   COPY_WS_SOURCE_TO_NAME,
/* 233 */   COPY_WS_DEST_BANK,
/* 234 */   COPY_WS_DEST_NUM,
/* 235 */   COPY_WS_DEST_NAME,
/* 236 */   COPY_WS_DEST_AFTER_STEP,
/* 237 */   COPY_WS_DEST_AFTER_BANK,
/* 238 */   COPY_WS_DEST_AFTER_NUM,
/* 239 */   COPY_WS_DEST_AFTER_NAME,
/* 240 */   COPY_WS_DEST_BEFORE_STEP,
/* 241 */   COPY_WS_DEST_BEFORE_BANK,
/* 242 */   COPY_WS_DEST_BEFORE_NUM,
/* 243 */   COPY_WS_DEST_BEFORE_NAME,
/* 244 */   MTUNE_C,
/* 245 */   MTUNE_CS,
/* 246 */   MTUNE_D,
/* 247 */   MTUNE_DS,
/* 248 */   MTUNE_E,
/* 249 */   MTUNE_F,
/* 250 */   MTUNE_FS,
/* 251 */   MTUNE_G,
/* 252 */   MTUNE_GS,
/* 253 */   MTUNE_A,
/* 254 */   MTUNE_AS,
/* 255 */   MTUNE_B,
/* 256 */   CURRENT_MTUNE,
/* 257 */   FX_PLACEMENT,
/* 258 */   FX1_PROG,
/* 259 */   FX2_PROG,
/* 260 */   FX_MIX_3,
/* 261 */   FX_MIX_4,
/* 262 */   FX_MOD_3,
/* 263 */   FX_MOD_4,
/* 264 */   FX_MOD_AMT_3,
/* 265 */   FX_MOD_AMT_4,
/* 266 */   CURRENT_FX,
/* 267 */   FX_PROG,
/* 268 */   FX_FOOTSWITCH_ENABLE1,
/* 269 */   FX_FOOTSWITCH_ENABLE6,
/* 270 */   FX_LFO_SHAPE,
/* 271 */   FX_MOD1,
/* 272 */   FX_MOD2,
/* 273 */   FX_MOD3,
/* 274 */   FX_MOD4,
/* 275 */   FX_MOD5,
/* 276 */   FX_MOD6,
/* 277 */   FX_MOD7,
/* 278 */   FX_MOD8,
/* 279 */   FX_MOD10,
/* 280 */   FX_LFO_RATE1,
/* 281 */   FX_LFO_RATE3,
/* 282 */   FX_LFO_RATE4,
/* 283 */   FX_LFO_RATE5,
/* 284 */   FX_LFO_RATE6,
/* 285 */   FX_LFO_RATE7,
/* 286 */   FX_SPLIT_POINT2,
/* 287 */   FX_SPLIT_POINT3,
/* 288 */   FX_SPLIT_POINT10,
/* 289 */   FX_DELAY_FACTOR7,
/* 290 */   FX_TOP_DELAY3,
/* 291 */   FX_WG_JUCT_MIX10,
/* 292 */   FX_EQ_FREQ_LOW0,
/* 293 */   FX_EQ_FREQ_MID2,
/* 294 */   FX_EQ_FREQ_HIGH7,
/* 295 */   FX_EQ_WIDTH6,
/* 296 */   FX_100_WET_DRY0,
/* 297 */   FX_100_WET_DRY3,
/* 298 */   FX_100_WET_DRY4,
/* 299 */   FX_10_WET_DRY0,
/* 300 */   FX_10_WET_DRY3,
/* 301 */   FX_10_WET_DRY4,
/* 302 */   FX_UPARAM0,
/* 303 */   FX_UPARAM1,
/* 304 */   FX_UPARAM2,
/* 305 */   FX_UPARAM3,
/* 306 */   FX_UPARAM4,
/* 307 */   FX_UPARAM5,
/* 308 */   FX_UPARAM6,
/* 309 */   FX_UPARAM7,
/* 310 */   FX_UPARAM8,
/* 311 */   FX_UPARAM9,
/* 312 */   FX_UPARAM10,
/* 313 */   FX_UPARAM11,
/* 314 */   FX_UPARAM12,
/* 315 */   FX_UPARAM13,
/* 316 */   FX_PARAM0,
/* 317 */   FX_PARAM1,
/* 318 */   FX_PARAM2,
/* 319 */   FX_PARAM3,
/* 320 */   FX_PARAM4,
/* 321 */   FX_PARAM5,
/* 322 */   FX_PARAM6,
/* 323 */   FX_PARAM7,
/* 324 */   FX_PARAM8,
/* 325 */   FX_PARAM9,
/* 326 */   FX_PARAM10,
/* 327 */   FX_PARAM11,
/* 328 */   FX_PARAM12,
/* 329 */   FX_PARAM13,
/* 330 */   FX_DEST_TYPE,
/* 331 */   FX_DEST_PROG,
/* 332 */   FX_DEST_FX_NUM,
/* 333 */   FX_DEST_PLACEMENT,
/* 334 */   FX_DEST_FX1,
/* 335 */   FX_DEST_FX2,
/* 336 */   WAVE_MUTE,
/* 337 */   WAVESEQ_WAVE,
/* 338 */   WAVE_SEQ_LOOP_DIR,
/* 339 */   WAVESEQ_COMPAND_SCALE,
/* 340 */   FOOT_DAMPER_FUNCTION,
/* 341 */   FOOT_DAMPER_POLARITY,
/* 342 */   FOOT_ASSIGN_1_FUNCTION,
/* 343 */   FOOT_ASSIGN_1_POLARITY,
/* 344 */   FOOT_ASSIGN_2_FUNCTION,
/* 345 */   FOOT_ASSIGN_2_POLARITY,
/* 346 */   BANK_COPY_TYPE,
/* 347 */   ENV1_MOD_VEL_RATE,
```

```
/* 348 */   ENV1_MOD_KBD_RATE,
/* 349 */   WS_MIDI_CLOCK,
/* 350 */   VIEW_BANK,
/* 351 */   VIEW_PERF_NUM,
/* 352 */   VIEW_PERF_NAME,
/* 353 */   COPY_FX_SOURCE_BANK,
/* 354 */   COPY_FX_SOURCE_NUM,
/* 355 */   COPY_FX_SOURCE_NAME,
/* 356 */   FX_11_WET_DRY0,
/* 357 */   FX_11_WET_DRY3,
/* 358 */   FX_11_WET_DRY4,
/* 359 */   FX_RAMP5,
/* 360 */   SOURCE_CARD_NAME,
/* 361 */   DEST_CARD_NAME,
/* 362 */   WAVEA_BUS_A,
/* 363 */   WAVEA_BUS_B,
/* 364 */   WAVEA_BUS_C,
/* 365 */   WAVEA_BUS_D,
/* 366 */   WAVEB_BUS_A,
/* 367 */   WAVEB_BUS_B,
/* 368 */   WAVEB_BUS_C,
/* 369 */   WAVEB_BUS_D,
/* 370 */   WAVEC_BUS_A,
/* 371 */   WAVEC_BUS_B,
/* 372 */   WAVEC_BUS_C,
/* 373 */   WAVEC_BUS_D,
/* 374 */   WAVED_BUS_A,
/* 375 */   WAVED_BUS_B,
/* 376 */   WAVED_BUS_C,
/* 377 */   WAVED_BUS_D,
/* 378 */   COPY_PART_SOURCE_BANK,
/* 379 */   GLOBAL_UTIL_DEST_BANK,
```

Parameter numbers greater than 379 are sent as expanded parameter change messages. Parameters 380-404 are ignored by the Wavestation Keyboard.

```
/* 380 */   REMAP_TO_JOY_X,
/* 381 */   REMAP_TO_JOY_Y,
/* 382 */   REMAP_TO_FX_SWITCH,
/* 383 */   PROG_TO_MULTI_FX,
/* 384 */   CHANGE_MULTI_WITH,
/* 385 */   ANALOG_LEV_1,
/* 386 */   ANALOG_LEV_2,
/* 387 */   ANALOG_CHAN_1,
/* 388 */   ANALOG_CHAN_2,
/* 389 */   ANALOG_1_BUS_A,
/* 390 */   ANALOG_1_BUS_B,
/* 391 */   ANALOG_1_BUS_C,
/* 392 */   ANALOG_1_BUS_D,
/* 393 */   ANALOG_2_BUS_A,
/* 394 */   ANALOG_2_BUS_B,
/* 395 */   ANALOG_2_BUS_C,
/* 396 */   ANALOG_2_BUS_D,
/* 397 */   FX_BUS0,
/* 398 */   FX_BUS2,
/* 399 */   ANALOG_BUS_MACRO,
/* 400 */   ANALOG_1_FILTER,
/* 401 */   ANALOG_2_FILTER,
/* 402 */   ANALOG_1_EXCITER,
/* 403 */   ANALOG_2_EXCITER,
/* 404 */   ANALOG_INPUT_DISABLE,
/* 405 */   COMP_CONTROL0,
/* 406 */   LOCAL_XPOSE,     /* Keep right
            before last */
/* 407 */   PARAM_END        /* Must be
            last */
              };
```

NOTICE

KORG products are manufactured under strict specifications and voltages required by each country. These products are warranted by the KORG distributor only in each country. Any KORG product not sold with a warranty card or carrying a serial number disqualifies the product sold from the manufacturer's/distributor's warranty and liability. This requirement is for your own protection and safety.

KORG KORG INC.
15-12, Shimotakaido 1-chome, Suginami-ku, Tokyo, Japan.

© KORG INC.

Printed in Japan
1991 0308 ETH