Loop® IV

For the Gibson/Oberheim
Echoplex Digital Pro

Upgrade Guide

Manual Version 1.02
LoopIV version 1.1
May 14, 2002

*a*urisis *r*esearch

Aurisis Research, LLC • PO Box 40200, Berkeley, CA 94704-4200 • http://www.aurisis.com • info@aurisis.com

MOST IMPORTANT!

At the first starting up of the new software, you have to do a Parameter reset. This resets all of the parameters to the correct default settings for LoopIV. To do the Parameter Reset: Hold down Parameters before you turn on the power, and keep holding it down until you see “Loop 4” scrolling across the screen. Then you can let it go. The Echoplex parameters will all have default values when it reaches the reset state.

Acknowledgements

Aurisis Research would like to thank all of the incredible users who have inspired us over the years with your wonderful music and creative ideas. Many of the new features and enhancements of LoopIV come from your suggestions. Your enthusiasm keeps us going! LoopIV was an incredibly difficult effort, without all of you we would never have finished it.

Several individuals deserve special recognition for their contributions to this project.

Claude Voit – For his constant stream of new ideas that led to many fundamental new features of LoopIV. The amazing new ReAlign synchronization feature was developed with Claude, along with many, many others. His tireless testing and debugging efforts over nearly two years did much bring LoopIV into reality.

Andy Butler – For dreaming up all of his InterfaceModes and creating completely new ways to control the loop in the process, as well as numerous suggestions about the subtleties of many functions. His tireless and thorough testing efforts also helped us to find numerous bugs.

Andy Ewen – For going way beyond the call of duty to make certain the Echoplex hardware stayed in production and the quality remained high.

Andre LaFosse – For his amazing and deep use of Loop and the Echoplex which helped inspire the whole range of SUS and granular functions, among other features. His uncanny knack for finding obscure bugs has been a tremendous help, as well as his quick testing of so many different feature combinations. Andre is famous for finding strange bugs and turning them into features.

David Torn – For his feature ideas, great music, and public support for Looping that has opened so many musicians to the possibilities in this technology. We hope Quantize=8th and LoopWindowing brings you great happiness.

Chris Muir – For constantly nagging us about the need for better MIDI control. He inspired the DirectMIDI control now in LoopIV.

Eberhard Weber – For his inspiration and support, and the idea of tempo LEDs.

David Kirkdorfer – For the MultiIncrease function and his great marketing studies that helped show just how much interest there is in the Echoplex.

Willy Strehler – For creation of the “double click” functions and his quick bug finding abilities. Willy also inspired many of the fundamental features developed in earlier versions of Loop, such as Quantize and the different Sync options.

Ljubo Majstorovic – For his testing and good humor and all the ideas he contributed for the previous versions.

Our beta testers - Claude Voit, Andy Butler, Andre LaFosse, Mark Hamburg, Bret Moreland, Dennis Leas, Jon Wagner, David Torn, Chris Muir, and Markus Reuter – thank you for getting us through the final stretch.

Henry Juszkiewicz and the Gibson Guitar Corporation – for continuing to believe in the Echoplex and keeping it alive all these years. Special thanks to all of the Gibson employees who have cared about the Echoplex and put some of their heart and soul into it.

Our Families and Friends – Thank you for your love and support and for putting up with all the time and energy that we have put into this.

If we’ve somehow forgotten you, please accept our apologies and let us know. We’ll be happy to update the list!

Aurisis Research
Matthias Grob, Eric Obermuhlner, Kim Flint
May, 2002

Contents

Audience for the Upgrade Manual

The LoopIV Upgrade Manual is intended for users who are upgrading their Echoplex from LoopIII to LoopIV. The intent is to explain the new features of LoopIV and how they work. It is assumed the reader is already familiar with existing functions and concepts from LoopIII, so they are generally not re-explained. You should refer to the LoopIII manual when you are unsure about an existing feature of Loop. In general, this is a reference manual and not an applications guide.

You may also wish to consult the Aurisis Research and Gibson Guitar Echoplex web sites for occasional manual updates and other useful information. at http://www.aurisis.com and http://www.gibsonechoplex.com.

Another good source of information is the Looper community at Looper's Delight, http://www.loopers-delight.com. There you can find many other loopers and Echoplex users sharing tips and techniques. There is also an Echoplex FAQ and pages of tips from different users.

New Feature Overview

The following is a brief list of what is new about LoopIV. Please consult the following chapters for details about any given feature.

New Functions and Features

- Half Speed
- SUS Commands – Real Time Granular Looping
- Substitute
- Feedback Display
- Tempo Select
- SyncRecord
- Sync ReAlign
- MultiIncrease
- Record-to-NextLoop
- LoopWindowing
- LoopDividing
- 16 Loops
- New InterfaceModes – Stutter, Expert, Input, Replace, Flip
- Parameter Presets
- Expanded MIDI Control
- DirectMIDI function access
- SUS MIDI commands
- MIDI SUSNextLoop
- MIDI PreviousLoop
- Sysex editing of Parameters and Presets
- MIDIpipe
- MIDI Note Sync Indicators
- Simple Retrigger while Playing
- NextOverdub
- SimpleCopy
- SmartButtons – MIDI "double-clicks"
- QuantizeStartPoint

- DataWheel
- Improved Reverse
- Improved Undo
- Improved Multiply and Insert cross-functions
- 8th/Cycle and sync changes occur in rhythm
- Improved SampleDump

New Synchronization Features

- SyncRecord
- Sync ReAlign
- Local/Global MIDI clock StartPoints
- Tempo
- SongPositionPointer and ContinueSong
- StopSync
- SyncStartPoint
- QuantMIDIStartSong
- MuteQuantMIDIStartSong
- QuantStartPoint
- Improved Sync routings
- 8th/Cycle up to 256

New Parameter Values

- MoreLoops = 16
- SwitchQuantize = Loop, ConfirmLoop, ConfirmCycle
- SamplerStyle = Start
- Quantize = Loop, 8th
- Loop/Delay = Stutter, Expert, Input, Output, Replace, Flip
- RecordMode = Safe
- 8th / Cycle up to 256
- Sync = Out and OutManualStartSong

New Display Features

- Visual continuous control values for Feedback and Volume
- Tempo LEDs
- Sync LEDs
- Command Display

Installation

Installing LoopIV v1.1 EPROMs

1. Before you start, please remember to be careful as you work. Changing the EPROMs is easy, but if you damage your Echoplex it is your problem. Aurisis Research is not responsible or liable for any damage you cause. If you do not feel comfortable following these instructions, find somebody who can help you.

2. You'll need a Phillips screwdriver, a small flat head screwdriver, the LoopIV software EPROMs, and a good surface to work on.

3. To avoid damaging anything with static electricity, touch a grounded metal object to discharge yourself. If you have to stop part way through the following steps, make sure to ground yourself again when you resume work. Make certain the Echoplex is also properly discharged before you begin. The best way to do this is by connecting the power cord to a properly grounded wall outlet, and then disconnecting it.

4. Unplug the power cord from the Echoplex. When changing the EPROMs, be careful not to touch other parts of the circuit, as some areas can hold a charge even after the power is off. When working around electronic devices, always practice the "one hand rule." That means, use one hand to work at a time. If you do accidentally get a shock, it will only travel across your hand.

5. Remove the screws holding the top of the Echoplex. There is one screw on the front at the upper right of the Gibson (or Oberheim) label, two screws on each side, and four screws across the back. Keep track of where each screw came from so you can put it back in the same place later.

6. Pull the top off. Put it aside.

7. The current EPROMs are the two chips in sockets just behind the memory SIMMs. They will be the same size as your new chips and have labels saying either "LD 3.32 ODD Master" and "LD 3.32 Even Master" or "LoopIIIv5.0 Even" and LoopIIIv5.0 Odd." If the labels have come off or it is not clear which are the correct chips, the reference designators on the PCB are U34 and U35.

8. Carefully remove the EPROMs from the sockets with the flat head screwdriver. Do this by inserting the flat part of the screwdriver between the body of the IC and the socket, and gently rocking back and forth to lift the pins out. Switch sides to make sure it comes out evenly. Be very careful so that you don't bend the pins. When it comes loose, lift it out and put it aside. Do the same with the other one. This should not require a lot of force. If the chips do not lift up easily make sure you are working with the correct chips and have the screwdriver tip between the chip and socket.

9. Take the new EPROMs out of the protective packaging. Put the old ones in the package to keep them safe.

10. The EPROM labeled "Even" goes in the socket labeled "Even" on the PCB, U34. The one labeled "Odd" goes in the odd socket, U35. With the front of the unit towards you, the notch on the top of the chip should be towards the left side of the unit, away from the transformer. The text on the label should be right side up. Carefully press the IC's into the sockets, making sure all the pins are going in correctly.

11. Put the top back on, replace the screws in all locations, plug in the Echoplex.

12. Hold the Parameter button down, and turn the power on. You should see "Loop 4" scrolling across the display. At that point you can release the Parameter button. This initializes the parameter memory correctly for LoopIV, and sets all Parameters to default values.

That's it!

Chapter 1: New Loop Functions

Half Speed

Half Speed is a new function that switches the playback speed of the loop to half the normal speed, making it an octave lower and twice as long. Half Speed is a new InsertMode option, making it available from the front panel. Half Speed is also available by MIDI.

When the new value H.SP. is selected in InsertMode, the Insert Button becomes the Half Speed button. Pressing it switches the current loop an octave lower, to half speed. The insert LED turns red and the display says H.SP briefly. Press Insert again and the loop returns to Full Speed. The LED turns green and F.SP is displayed for a moment.

The function is reset to Full Speed with Reset, but it can be selected while still in Reset. This allows you to start a loop in Half Speed with the audio sounding normal, and then switch to full speed. It ends up as double speed, one octave higher!

All other functions work normally in Half Speed. The speed can be switched anytime during playing or Reset, even while in the middle of overdubbing or multiplying! So as you are overdubbing you can switch freely between Half Speed and Full Speed to get interesting octave and speed jumps in the middle of the overdub.

The sound quality is somewhat reduced during Half Speed because the sampling rate for the audio is cut in half. Also note that MIDI piping is slowed down by half speed, so it is possible to see to see slight delays in very dense MIDI streams. See the MIDI chapter below for more details on MIDIpipe.

SUS Commands – Real Time Granular Loops

Another new value on the InsertMode parameter is called SUS. This is short for Sustain.

SUS changes the way in which the Insert and Multiply buttons work. It turns Insert and Multiply into Unrounded functions with Sustain action on the button. In other words, they start when the button is pressed and end immediately when it is released, just like Record or Overdub = SUS always did. When the function ends it does so as if Record had been pressed as an alternate ending to the Insert. This is what we call an “Unrounded” multiply or insert, because instead of rounding off to the next Cycle point it is ended immediately and the loop time is redefined.

With Quantize = OFF, the effect of SUS with Multiply and Insert allows you to splice together fragments of sound into a loop. One use of this is to create short loops and splice short "granular" sounds together in real time by tapping on the multiply or insert buttons as sounds are played into the input. If you hold the button down, the Multiply or Insert goes on as long as you hold it, but if you just tap the button lightly the functions will only be active for as long as the switch is contacting. This can be as short as a few milliseconds, allowing you to splice together very short fragments. Combine InsertMode=SUS with RecordMode=SUS and OverdubMode=SUS, as well as the new MIDI commands to access other functions as a sustain action. (like ReplaceSUS and SubstituteSUS). SUS techniques give exciting new timbres and glitch effects, all created in real-time. Real-Time Granular!

With other Quantize values the SUS versions of Insert and Multiply start and stop quantized. There will always be an Insert or Multiply of at least one time period as determined by the quantize setting (Loop, Cycle or 8th). Even if you quickly tap the button such that it is actually released before the start of the function, you will still get one time period worth of the function. This is very useful when working with short loops where it is important to maintain a rhythmic length. With SUS you can get much quicker Inserts and Multiplies than you could if you had to press the button twice. Note that this quantized behavior is true with other SUS functions, like Replace and Substitute.

With Quantize = CYC it's easy to create rhythmic sequences of sounds when using SUS commands.

With Quantize = 8th a short press of Multiply will change the loop length to one Cycle divided by the value of 8th/Cycle.

Substitute

The New Substitute Function
Substitute has some similarity to the Replace function that we had from LoopIII. With Replace the original loop playback is cut while the replace is done. So while you are playing something new to Replace what was there, you don't hear the old loop. Replace is useful when the new material would clash with what was there, but oftentimes the result not very tight since you don't have any guide to play along to as you are doing the Replace.

With Substitute the original loop playback continues while you are playing the new material. On the next repetition, only the new audio will remain in the loop and the old portion will be removed. This helps keep the groove going while substituting and gives you something to play along to, as well as giving an overlap between the old portion and the new portion for continuity. Substitute is the same as if you were doing an Overdub with the feedback turned down to zero only during the Overdub.

Using Substitute
Substitute can be used in several ways:

1. an Insert button press when InsertMode=Sub
2. a LongMultiply (less accurate, see below)
3. a Record-Insert combination when InsertMode=rhr. (formerly called Rehearse)
4. the Substitute VirtualMIDI button
5. the SUSSubstitute DirectMIDI command

As long as Substitute is active, all playing is repeated once. This can be useful to find the groove to start a loop. Just hold Substitute down as you play, and when you've played something you like let it go!

When Quantize is Off, Substitute is an instant function with Sustain action, same as Replace. This means it is active while the Insert button is pressed down and turns off when you release the button.

When Quantize is on, pressing Substitute down puts the Echoplex into the waiting state until the next Cycle point. Once the substitute starts, releasing it also goes into a waiting state until the next Cycle or loop StartPoint is reached. If you simply tap the Substitute button, it will be active for exactly one Cycle.

There is a problem when you try to use Substitute with the LongMultiply option. During the first 400ms until the switch action is detected as a long press, it's treated like a Multiply. This means the old loop is still present for those 400ms, and only after that it mutes for the substitute. If you have quantize on you will not have this problem, since the long press can happen completely during the ooo waiting phase. When the Cycle point comes Substitute is started directly.

Advanced Use of Substitute
If a Feedback Pedal is connected and you are using some of the new InterfaceModes, Substitute has some extended functionality that makes it even more powerful. StutterMode and ReplaceMode have this capability. While the pedal continues to do Feedback during normal playing, the front panel Feedback knob controls the feedback just for the Substitute function. So you can have different settings for each! If you have the FeedBack knob all the way up, The existing audio is completely preserved as you are adding more. So it turns into Overdubbing. With the Feedback knob turned all the way down, the existing audio completely disappears on the next repetition, so it is the normal Substitute. In between is where it is interesting, because you can choose how much the level of the existing audio should decay each time you do an "overdub" with Substitute. In ReplaceMode the loop output level is also set to 100% during Substitute instead of being set by the Pedal as it is otherwise. See the section on the new InterfaceModes for more details.

SyncRecord

SyncRecord is a variation of Record that is done when a Sync of any type is being received and Quantize is off. Instead of always quantizing Record when a sync is coming in as was done in LoopIII, we now do a kind of "Multiply over nothing" for the unquantized case. This means SyncRecord starts immediately when you press Record, counts the Cycles on the green display, and rounds at the end to fit the loop time defined by the sync. SyncRecord gives freedom from quantization, while still allowing tight synchronization to an external clock source.

During Reset, the Overdub LED turns yellow to indicate that a Sync arrived. When the second Sync point arrives to define the loop length, the display shows the resulting Cycle time. Whenever the Overdub LED is yellow like this, the next Record press will be a SyncRecord.

With SyncRecord, you only need to have received the first sync event to begin Recording. As you are Recording, the Echoplex will continue watching the sync to determine what the right cycle times are.

SyncRecord does make one improvement when Quantize is on. In LoopIII, as a loop was recorded in sync the cycles would not be counted or differentiated. Now the Cycles are tracked and counted properly. This means that if the incoming clock defines a Cycle length of 2 seconds and you let Record continue to 8 seconds, you will see the multiple counter counting from 1 to 4. The Cycle boundaries will be set at 2.0 seconds. In LoopIII it would have just made a big 8 second cycle. This is especially useful now since the Quantize function can quantize to either the Cycle boundaries or the Loop StartPoint.

Tempo Select

Tempo Select is a way to set up the tempo of a loop in Beats Per Minute (BPM) before you record it. Once you set the BPM, the basic loop length is determined by the 8ths/Cycle parameter. You can also think of this as setting the loop time ahead of recording the loop.

The tempo is set with the FeedBack knob or by MIDI. While still in reset and before a loop is recorded, you enter the Tempo state with a press of the Undo button. From there you can set the Tempo.

After setting the tempo the loop can be recorded. When you press Record, the Echoplex actually does a SyncRecord to the clock tempo that has been set. (See the SyncRecord section above or in the Sync chapter for more details on SyncRecord). This allows you to start the Record at any time. When you press Record again to finish, it will continue to the precise loop time determined by your tempo and the 8ths/Cycle parameter, and end the Record automatically.

Once the tempo is set in the Tempo state, MIDI clock is sent out. This allows you to start a sequencer or drum machine in time with your loop length before you even record the loop! Or similarly, it allows you to start a sequencer at the exact time you *start* recording a loop, instead of when you finish it! Tempo Select gives you a lot more flexibility for working with sequencers and other synchable devices over what was possible with LoopIII.

How to Use Tempo Select

Tempo Select requires that the Sync Parameter be set to Out or OuS. When you have Sync set to Out or OuS, the Undo LED will be green in reset to indicate the Tempo function is available.

To select the Tempo, first press the Undo button in Reset. This will put you into the Tempo Select State. From there you will see the display change and you will have several different commands available from the front panel.

The Tempo Select Display

When you enter the Tempo State, the Undo LED will turn red and the BPM will appear on the LoopTime display. The tempo LEDs will begin flashing to the beat. You will also see that the Record LED is green and the Overdub, Insert, and Mute LEDs will be Orange to indicate they have special functions.

Tempo Select Commands

Record – record a loop in tempo. It will automatically do a SyncRecord to the selected tempo.

Overdub – Press to disregard the Tempo without erasing it. If you press it again later or reenter the Tempo Select function, the tempo returns.

Feedback Knob – sets the Tempo.

Insert and **Mute –** use to fine tune the tempo.

Short press of Undo – locks the tempo and triggers a StartSong message.

Long press of Undo – exits from the Tempo Select state and switches the feature off. Any time a tempo has been set, a Long Undo during reset will clear it.

Setting the Tempo
Select the tempo with the FeedBack knob. You can select a tempo between 26 and 278 BPM. Tempo can also be set by MIDI using the DataWheel continuous controller. (controller #6)

While the knob is being turned, the Tempo is displayed in BPM on the LoopTime display. Once you've stopped turning it for a moment, the resulting loop time is displayed in place of the BPM. The loop time depends on Tempo and the 8th/Cycle parameter. We assume that a beat is a quarter note, so at Tempo 120 BPM and 8th/Cycle=8, the Cycle time results in 2.0 seconds. If 8th/Cycle is 16, you get 4.0 seconds, and so on.

With the feedback knob the tempo is adjusted coarsely, in 2 BPM increments. The Insert and Mute buttons can be used to fine tune the tempo from there. Insert reduces the LoopTime (increase BPM) and Mute increases the LoopTime (reduces BPM). Each press changes the loop time by approximately 3 milliseconds. Fine tuning changes are not shown in BPM, it only changes while the LoopTime is displayed. Unfortunately, the LoopTime usually shows the time in 100ms resolution! So you might not be able to see anything change on the display as you fine tune the tempo, until you have changed it by 100ms. But you can hear it. Since the MIDI clock is being sent out during this time, any device following the clock will be slowly changing in tempo as it follows the fine tuning.

A long press of Undo clears the Tempo and exits from Tempo Select.

Differences Between Sync = Out and Sync = OuS
Tempo Select behaves slightly differently depending on whether the Sync parameter is set to Out or OuS. With Sync = Out, StartSong messages are sent when you start Recording or when you Set the tempo with Undo. When Sync = OuS, StartSong is only sent at user command with the press of Undo in the Tempo Select state, but not sent when you start or stop Recording. If you have recorded a loop without StartSong, you need to do one of the new Quantized StartSong functions to send a StartSong message. This can give you more freedom in controlling when the sequencer starts.

See the Synchronization chapter for more details on the differences between Out and OuS. Details on the Quantized StartSong functions can be found in the Synchronization chapter and in the MIDI chapter.

MIDIclock and StartSongs
MIDIclock is sent out immediately when you enter the Tempo state, but without a StartSong message. Some devices like to have MIDI clock in advance, and for some cases this allows you to get a feel for the rhythm. But to really start things, you need to send a StartSong message!

Recording in Tempo
There are three ways to send the StartSong and get things started:

1. Start up the sequencer before recording any loops
The first option for starting the sequencer is to start it in tempo before recording any loops. After you have entered Tempo Select and set the Tempo, press Undo again. This press of Undo sends a StartSong message to the sequencer and locks in your tempo. We call it SetTempo. The sequencer will receive the StartSong message and start playing at your tempo using the MIDI clock out from the Echoplex, and the Echoplex and sequencer will be aligned from then on. If you don't like the tempo you can press Undo to set it again with the feedback knob or the fine tune buttons. This StartSong is sent if sync is OuS or Out.

If you are not using a pedal for feedback, make sure you set the feedback knob back to where you want it for feedback before recording! Since you have locked the tempo, changing the knob position at this point will not change tempo, only feedback. You may also find the new RecordMode=Safe parameter helpful here.

Whenever you are ready to record your loop, you can simply tap record to begin. You will actually do a SyncRecord. After the second press of record to end recording, the Echoplex will round off the recording to the next sync point as determined by your tempo. Your loop will end at exactly the right length and in time with the sequencer.

2. Start up the Sequencer as you begin Recording your Loop
The second option for starting the sequencer is to trigger it immediately as you start Recording. You can do this by pressing Record directly when the Undo LED is still red, right after you have dialed in the tempo. A StartSong message is sent, and the sequencer will start at the same instant as you start recording your loop. In this case the press of Record is the SetTempo moment. When you press Record again to finish, the Echoplex rounds off the recording to the correct loop time, same as before. For this to work you have to have Sync = Out.

3. Record a Loop and Start the Sequencer Later
The third option lets you Record a loop without starting up the sequencer. This requires the Sync parameter to be set to OuS. Set the tempo in the tempo state, and then record your loop to it as above. With OuS, the StartSong message is not sent out when Recording is started or stopped, so the Sequencer will not start up. When you are ready to start the sequencer, you need to send a StartSong message with the QuantStartSong command executed by pressing Mute and then Multiply while the loop is playing. At the next StartPoint of your loop a StartSong message will be sent automatically and your Sequencer will start. You can also use one of the MIDI StartSong commands which don't necessarily require you to mute your loop first. More details about the quantized StartSong commands can be found in the Synchronization chapter and the MIDI chapter.

SetTempo and Presets
The SetTempo moment when the tempo is locked in is important. At SetTempo the tempo you have defined with the feedback knob is stored as a parameter value in memory. You can then save it as a preset and recall it again later. This allows you to have predefined tempos stored in different Presets and jump to them immediately. See the Presets section for more information on Presets. SetTempo also means that the tempo is remembered if you go out of tempo and then come back. For example, after SetTempo and while still in Reset you press Overdub to disable sync. At that point you can record a loop out of sync if you like. After reset, press Overdub again to re-enable sync, and then press Undo again to go back into the Tempo state. Your old tempo is still there!

Sync=In and Sync=Off
If Sync=IN, the Tempo Select function can not be selected. The Undo LED is actually orange, indicating an alternate function. In this case a different function is available, where StartSong can be sent in Reset with a press of Undo. See the Synchronization chapter for more details on this.

If Sync = Off, Tempo Select is not available and StartSong messages are not sent. The Undo LED is off.

Storing Tempo in Presets
Tempo can be saved in a Preset. Whenever you recall that preset it will immediately come up with that tempo. To do this, you simply save to a Preset while you have a tempo set, and it will be stored. See the Presets section for more information on how to save and recall presets.

When you recall a preset where no tempo has been set, it is just as if this feature did not exist at all. It comes up without any tempo and behaves normally.

When the Tempo state is activated with Undo, it first displays the tempo value in the preset currently loaded.

If there is no tempo saved in the current preset, it defaults to 120BPM. From then on, as soon as the Feedback knob is moved, the new values is activated.

MultiIncrease

Multiply now has a feature to aid in creating very long multiplies, called MultiIncrease. MultiIncrease is also useful when you know exactly how many multiples you want to do in advance. Instead of waiting until the end of the multiply to make the second button press, now you can immediately tap in as many Multiples as you want in the beginning of the multiplying. The Echoplex will automatically complete that many multiply Cycles for you. This same function is also available for Insert, however for simplicity we will just describe it in terms of Multiply.

MultiIncrease is very helpful for situations where you want to have a large number of multiples and you don't want to wait to the very end to remember to press Multiply a second time. This way you can set up in advance how far it will multiply and let it go while you continue playing. MultiIncrease is in addition to the normal Multiply operation, so the traditional use is not affected.

How to Use MultiIncrease
Once you have started multiply, immediately tap the multiply button again to signal you want to end. The Echoplex begins Rounding off the multiply, just as it always has. During the Rounding period, continue tapping Multiply to increase the number of Cycles you want to add. The number of Cycles where Multiply will be stopped is briefly displayed as C<number> while you are tapping them. If you like, you can tap them in very quickly right from the beginning. Or, if you have had multiply going for a while, using MultiIncrease simply adds to the number of multiples you already have.

If you are tapping the Cycles in quickly, it is helpful to remember that the first tap of Multiply is just starting it. The *second* tap is where you start counting the total number of Cycles you will get. This can throw you off when you count the Multiply taps quickly, because you need to tap one extra time than the number of Cycles you want. So if you want 4 Cycles total, you need to tap five times. You might count it *start – 1 – 2 – 3 – 4*.

Example

- Record a loop.
- Tap Multiply 4 times and you get:
  - Tap 1: Start Multiply
  - Tap 2: Stop Multiply, begin Rounding
  - Tap 3: MultiIncrease (Cycles = 2)
  - Tap 4: MultiIncrease (Cycles = 3)
- You've set it to Multiply by 3
- At the third Cycle, the multiply will stop automatically.

MultiIncrease with Quantize=Loop
When Quantize=Loop, MultiIncrease adds entire loops. For example, if the loop consisted of 4 Cycles, MultiIncrease counts C 8, C12, C16, etc.

MoreLoops up to 16

There are 16 loops available now, instead of 9 as in LoopIII. On the Loop display the numbers above 9 are shown with letters, due to the lack of a leading 1. So they go 1, 2, 3…9, A, b, c, d, E, F, G.

New SwitchQuantize Values

SwitchQuantize has three new values that add quantizing to the loop and mix the idea of Confirm with the idea of Quantize.

Loop (SwitchQuant = LOP)
Jumps to the chosen loop at the next loop end. Useful when multiplies and inserts have been done.

Confirm Cycle (SwitchQuant = CCY)
Similar to Confirm. (SwitchQuant=CnF). After a confirming action is done, it additionally quantizes to the next Cycle point.

Confirm Loop (SwitchQuant = CLP)
Similar to Confirm. (SwitchQuant=CnF). After a confirming action is done, it additionally quantizes to the next Loop StartPoint.

An example for Confirm Cycle (CCY):

- Have two loops recorded.
- Press NextLoop, it waits for you to do some action.

- Press the function you want (Record, Overdub, Multiply, Insert, etc...)
- The action will begin in the new loop after the next Cycle point of the current loop.

Confirm Cycle and Confirm Loop also give you an alternate quantize type for operation in the current loop:

- You are in loop number 1
- Press NextLoop until next loop number 1 is displayed
- Any operation will begin according to the Confirm setting, regardless of the Quantize parameter. You can even start Overdub quantized this way!

LoopDividing

The Quantize parameter has an important new value, $8^{th,}$. With Quantize=8^{th}, functions execute at subdivisions of the loop Cycles, giving us LoopDividing.

The 8^{th}/Cycle parameter normally determines how the loop is divided. For example, if 8^{th}/Cycle = 8 the subdivisions are on 8^{th} note boundaries of the Loop time. If 8^{th} = 4, the subdivisions are on quarter notes. With the new values available in the 8^{th}/Cycle parameter, you have a wide range of options for dividing your loop.

The exception is when Sync=In and a MIDI clock is being received. MIDI clock specifically defines 8^{th} notes, so the MIDI clock information is used for Quantizing to 8^{th} notes in this case.

LoopDividing appears simple at first, but offers powerful new techniques when combined with other functions. For example, Replace and the new Substitute function can be used to easily change exactly one eighth note in a loop. Or you can execute Reverse aligned to the nearest quarter note, which feels almost immediate but keeps your loop in tempo as you switch in and out of Reverse.

See the New Parameter Settings chapter for more details on the new options for Quantize and 8^{th}/Cycle.

New SamplerStyle Options

SamplerStyle is a parameter that affects how multiple loops are triggered by MIDI or the NextLoop button. A new option called Start has been added, and several of the other options have been improved.

SamplerStyle = Start (StA)
Start is a completely new value for SamplerStyle. It makes the new loop start from the beginning and play forever. This is true whether the loop is entered with NextLoop or triggered by MIDI.

SamplerStyle = Once (OnE)
Once gained an important new functionality over what it had in LoopIII. It still triggers the loop from the start, plays it once, and then goes to Mute when the loop is triggered by MIDI.

However, with the NextLoop button it now plays the next loop once and then returns to the previous loop automatically. This is very helpful as a way to improvise the form of your music. You could have the 'A' section looping in Loop 1, and at some point decide you want the 'B' or 'C' section to drop in for one repetition before returning to the main loop. With SamplerStyle = One you can do this with one press on NextLoop and let the Echoplex take care of everything for you.

There are a few differences in functionality while the new loop is playing once:

- **Undo**: If you decide you want to stay in the new loop instead of bouncing back, you just have to press Undo while it is playing the single repetition. Instead of bouncing back to the first loop when it reaches the end, it will keep repeating the new one.
- **Mute**: if you press Mute during the second loop, it will Mute and stay in that loop.
- **Insert**: If you press Insert while the second loop is playing, it will retrigger. You can retrigger it as much as you like, and when you let it go to the end it will return to the first loop.

- **Multiply**: Multiply is not available while the second loop is playing.
- **Overdub**: If you turn on overdub while the second loop is playing, it will be assumed that you want to make some change to the new loop and it will not switch back to the first one. If you have SwitchQuantize on and press Overdub while you are still waiting for the first loop to finish, Overdub will be on when you go to the second loop. Again, it will not return to the first loop after it is done, and the overdub is kept.
- **Next**: Next is interesting. If you press NextLoop again while the second loop is playing, you will go to a third loop. When the third loop is finished playing once, it returns to the second loop. When the second loop finishes playing one more time, it returns to the first loop! So you can stack up a sequence of jumps and then return to the beginning, all automatically! There is a limit to this, in that you can't NextLoop through the same loop several times and automatically jump back to it that many times. It will stop the first time it returns to that loop and ignore previous steps in the sequence.
- **Record**: If you press Record while the second loop is playing, you will record a new one. It will continue repeating instead of jumping back to the first loop.
- **AutoRecord**: If the loop you jump to with Once is in reset and AutoRecord is on, you record the B part and immediately jump back to the previous loop when you tap Record to finish. If you press some additional function during recording for an alternate ending (like Multiply or Insert), it will go ahead and do that function and stay in the second loop. There is a limitation with AutoRecord and several loops. If you use AutoRecord to record several loops it will only jump back one loop at the end instead of jumping back through all of them.
- **LoopCopy and TimeCopy**: If you have LoopCopy on or if you engage a LoopCopy by pressing Multiply, Insert, or Overdub when using SwitchQuantize to change loops, the copy will be made into the new loop. When you finish the copy with a press of Multiply (or Insert) it will jump back to the first loop. This is an interesting way to make copies into a new loop and not necessarily listen to it repeat immediately.
- Similarly, if you press any additional function during the quantizing period, it does not switch back to the previous loop. The Echoplex assumes you want to elaborate on the B part.

SamplerStyle = Attack (Att)
The existing SamplerStyle value Attack only makes sense when loops are triggered by MIDI. This gives a keyboard like "play as long as you press" function, but it never made sense with NextLoop. So now when the NextLoop button is pressed (or MIDI- NextLoopButton), it is the same as "run". The loop starts in the same place where you last left it. If you used SamplerStyle=Att with LoopIII to make the new loop restart, you should now use SamplerStyle=STA.

SamplerStyle = Run (run)
Run is unchanged from LoopIII. With NextLoop or with MIDI triggers, you enter the loop at the last point where it was left.

Record-to-NextRecord

Many users requested the Record-to-NextRecord function. With multiple loops set up, you can now end recording with a tap of NextLoop and jump immediately to the next loop.

If Sync is being used, the recording will be rounded first and it will jump quantized at the Sync StartPoint.

If AutoRecord is on, the Echoplex immediately continues recording in the new loop. This is especially useful for filling the loops with the various parts of a song while playing continuously. You just keep pressing Next as you play! Similarly you can have LoopCopy set to Sound or Time, and it will automatically copy from one loop to the next as you press NextLoop.

When MoreLoops is 1, ending Record by pressing Next does a stop Record and begins playing the loop, just as if the recording had been ended with another press of Record. Note however, that when MoreLoops = 1, NextLoop becomes a retrigger button, so Record-to-Next can be an interesting way to immediately go into stutters of your loop.

LoopWindowing

LoopWindowing originally started as an obscure bug in the LoopIIIv5.0 software. People liked it so much they insisted we not fix it, and instead turn it into a feature! The “bug” has now been cleaned up to work predictably and in a consistent manner with other functions. So now it really is an interesting function that we call LoopWindowing.

Understanding LoopWindowing
LoopWindowing lets you define a short segment, or Window, out of a longer loop and let that short segment repeat as a loop. This Window is defined on the fly, in real time. You then have the ability to move that window through the larger loop as it exists in memory. In fact, it is more than just moving the window over the loop as it currently exists, you really move back through the memory, through all of the changes that have been recorded by overdubbing and multiplying and whatever. It stops when you reach the very initial point where the first tap of Record happened. LoopWindowing can give a variety of interesting effects, depending on the size of the Window and how much material is in the memory to Window through. You can even resize the Window on the fly, to capture different sized chunks of memory!

Creating the LoopWindow
The LoopWindow is created by either re-multiplying a loop or doing an Unrounded Multiply. Both of these are standard techniques that were possible in past versions of Loop. Re-Multiplying is done on a loop that has already had Multiplies or Inserts done on it, so you can see the Multiple display counting the cycles. If you press Multiply again on this loop, and then end it somewhere before the end of the loop, you will get a new loop of just that section. In this case it will be neatly rounded off to the previous cycle length. This technique allows you to chop out Cycles from the larger loop. You may want to experiment with setting Quantize to Cycle or 8^{th} as a way to get rhythmically aligned LoopWindows.

Unrounded Multiply is when you start a Multiply on a loop, and then end it with a press of Record. Instead of rounding off the cycle, it will stop immediately and redefine the new loop length at exactly that point. Unrounded Multiply is a great way to change rhythms by chopping out a completely new loop lengths. Using the new InsertMode=SUS function is also an interesting way to create Unrounded Multiples.

Either one of these techniques let you chop out a segment of your loop, either maintaining rhythm or not depending on what you want to do.

Moving the LoopWindow
Once a LoopWindow has been defined, we can move it backwards through the loop memory by pressing Undo. With each Undo press, the LoopWindow jumps back in memory by the size of the window, and then loops over that section. You can continue moving the window backwards to the point where the initial loop was started with the first tap of Record. If a Reverse has been done on the loop, then you can only move it back to the point of the Reverse.

Moving the LoopWindow works in the same way as Undo works, so it is useful to understand the distinction between a ShortUndo and a LongUndo. (Check the original Echoplex manual for more discussion on Undo.) Basically, a long-press of Undo will jump you back a complete LoopWindow length before your current window, and is the most obvious to use. A short-press of Undo sets the LoopWindow to *end* at the spot where you press it and *begin* a LoopWindow length before that.

For example: if you redefine the length of the LoopWindow from 8 seconds to 2 seconds, and then tap Undo at 1.5 seconds, it is only the last .5 seconds that change in that window. The previous 1.5 seconds of the Window remain intact in the new Window after that initial Undo button press, except they will now be coming at the end of the LoopWindow. Your new loop will start .5 seconds *before* the previous LoopWindow StartPoint, and end at the 1.5 second point where you tapped Undo.

Using ShortUndo is more complicated to understand, but is also more flexible. If you want to scroll through different sections of the loop cleanly with ShortUndo, press Undo right at the beginning of the window. This way you will really jump back a whole Window length. If you hit Undo somewhere within the boundaries of the window, you'll find that you get a blend between different memory window sections, with that blend happening at the exact point you hit Undo. So the timing of the Undo button press becomes a powerful tool for playing with the distinction between window fragments. It is especially powerful in rhythmic loops.

Modifying the LoopWindow
You can define new LoopWindow sizes at any time by doing more Re-Multiplies or Unrounded Multiplies, and then move the new LoopWindow over the loop.

Once you have a LoopWindow defined, you can do any other loop function on it that you like. For example, you can overdub new material onto it. Pressing Undo after that will first remove the overdubs, and then begin jumping backwards through memory of the larger loop.

Retrigger while Playing

When MoreLoops = 1, the otherwise unused NextLoop button turns yellow and does a Retrigger function. When you press it, the current loop will retrigger from the start, and then continue looping. This is similar to doing Mute-Undo to retrigger a loop, but without silence from having to mute first! This is only available when you have one loop set up in MoreLoops, since the NextLoop button obviously changes loops otherwise. There are also new MIDI commands available for Retriggering loops, so if you need to Retrigger when you have multiple loops set up you should consider those. Details are in the MIDI chapter.

InterfaceModes – New Loop/Delay Parameter Values

The Loop/Delay parameter determines how feedback, loop input volume, and loop output volume are controlled during various states. The parameter affects how you interact and control the loop, and different settings will be more or less useful to different players and different styles of looping. We call these InterfaceModes. Basically, InterfaceModes reroute the control signals from the feedback knob on the front panel and the feedback pedal input on the back, and determine when they are active and which parameters they control. In some cases these settings end up affecting Insert in interesting ways as well.

There were only three settings before – Loop, Delay, and Out. Loop has always been the default setting and most people use it. DelayMode is there to give a familiar style of operation to people accustomed to using delays. OutMode was only available if a pedal was inserted in the Feedback Pedal jack, and is really just like LoopMode but with Loop Output level controlled by the pedal while feedback was controlled by the knob.

Now we have added several new options to allow new ways to interact with the loop, for a total of eight. Four of the InterfaceModes are available at any time, and four require a pedal to be inserted in the Feedback Pedal jack. Those four are not visible in the parameter selection unless the pedal is connected.

These InterfaceModes are really expert functions, for experienced users to find subtle new ways to interact with loops. For newer or less experienced users, we recommend that you stay with LoopMode until you feel ready to experiment with the other InterfaceModes.

LoopMode (Loop/Delay = LOP)
LoopMode is the default setting for the Loop/Delay parameter, and is the most common way of using the Loop. This is the InterfaceMode we recommend people to start with, and most people stay with it. In LoopMode feedback control is always active, whether overdubbing or not. Feedback is controlled by the front panel knob if there is no pedal inserted, or by the pedal if it is there. Input or Output levels are fixed all the way on or off depending on the function, so these are being set for you according to what you are doing.

State	Feedback (Pedal/NoPedal)	Loop Input	Loop Output
playing	Pedal/Knob	0	100%
overdubbing	Pedal/Knob	100%	100%
substituting	0	100%	100%
Recording	NA	100%	0
multiplying	Pedal/Knob	100%	100%
inserting	0	100%	0
replacing	0	100%	0
mute	100%	0	0

StutterMode (Loop/Delay = Stu)
StutterMode is just like LoopMode, but Insert works as a "Single Cycle Multiply". When you have done a multiply and have several Cycles in a loop, pressing Insert will insert repetitions of the next Cycle. As the inserts are made you can overdub a longer phrase over the repetitions of the Cycle, with the results inserted into the loop when you press Insert again. If you press Undo instead, the loop will return to its original form. Using Insert-Undo like this lets you alter the flow of a loop by having one of the Cycles Stutter in a way similar to a skipping CD, and then return to the original. This can make very interesting results when working with very short Cycles, and that is why it is called StutterMode.

Stutters can be done into a new loop as a copy function. With multiple loops set up in MoreLoops, and SwitchQuantize on, pressing Next-Insert will do the stutter into a new loop. You can overdub on this as it stutters, and keep it with another press of Insert. Pressing Undo sends you back to where you were in the previous loop. Copying a stutter is a good way to preserve the original loop while making stuttered variation out of a fragment from it.

If you perform a very large number of repetitions of Insert and Undo button-presses to trigger and cancel the SingleCycleMultiply, you may eventually notice bits of the loop being erased by the Undo presses as well. This is a result of the way the Echoplex processes its memory. If you're planning to do heavy Insert + Undo button combinations with SingleCycleMultiply, you should be aware of this, and consider copying your loop via NextLoop before doing intensive stutter work, so you can return to the original loop fully intact if you wish.

Another trick you can do to avoid the loss of the overdubs is to first fill a bit of memory reserve by letting the loop repeat a few times without AutoUndo. (without the left green AutoUndo dot LED blinking.) You can do this by reducing Feedback a little bit, say to 120 – 125. That is small enough that the fading will not be obvious over a couple of repetitions, but you will force the Echoplex to copy the loop a few times into new memory. Obviously by doing this you lose some of the older stuff in memory, which you will note if you later want to go backwards with Undo. The reasons why this works are very complicated, but suffice to say that you will not find bits of your loop disappearing when doing heavy stuttering effects!

Also note that Substitute gains more advanced control in StutterMode. If you have a Pedal inserted for Feedback control, the pedal controls the Feedback during normal use and the knob setting is not used. However, during Substitute the knob becomes active for Feedback control. This lets you have two different Feedback settings between normal playing and Substituting. If you do not have a pedal inserted, Substitute operates the way it normally does in LoopMode and has feedback set to 0 while active. See the Substitute section to learn more about this function.

State	Feedback (Pedal/NoPedal)	Loop Input	Loop Output
playing	Pedal/Knob	0	100%
overdubbing	Pedal/Knob	100%	100%
substituting	Knob/0	100%	100%
recording	NA	100%	0
multiplying	Pedal/Knob	100%	100%
inserting	Pedal/Knob	100%	100%
replacing	0	100%	0
mute	100%	0	0

DelayMode (Loop/Delay = DEL)
DelayMode operation is like a traditional delay, and is useful for people familiar with that style of looping. In a traditional delay, the input to the delay line is always open and feedback is always being applied. When a “Hold” button is pressed, the input to the delay is closed, and the feedback is set to 100%. So in DelayMode during normal playing, the Loop Input level is always open and feedback is controlled by the front panel knob. The Overdub button becomes the “hold” button, so when you press it the feedback is set to 100% for infinite repeats and the input is closed so that nothing new is added to the loop. This is different from the LoopMode style, where feedback is always available to control whether overdubbing or not.

In DelayMode, the foot pedal controls the Loop Input level, which is useful as a way to do volume swells into the delay line.

One improvement has been made to DelayMode from LoopIII. The "Hold" function accessed by pressing Overdub also works while Multiplying and while the loop is Muted, which is more consistent.

State	Feedback	Loop Input	Loop Output
playing	Knob	Pedal	100%
overdub (hold)	100%	0	100%
substituting	0	Pedal	100%
recording	NA	Pedal	0
multiplying	Knob	Pedal	100%
multiplying(hold)	Knob	0	100%
inserting	0	Pedal	0
replacing	0	Pedal	0
mute	100%	Pedal	0
mute(hold)	100%	0	0

ExpertMode (Loop/Delay = EXP)
ExpertMode uses the pedal for Feedback during play and the front panel FeedBack knob for feedback during Overdub, Multiply, and Substitute. This allows you to have different feedback settings between playing and overdubbing. When there is no Pedal connected, the Feedback during play is always max (100%).

State	Feedback (Pedal/NoPedal)	Loop Input	Loop Output
playing	Pedal/100%	0	100%
overdubbing	Knob	100%	100%
substituting	0	100%	100%
recording	NA	100%	0
multiplying	Knob/100%	100%	100%
inserting	0	100%	0
replacing:	0	100%	0
mute	100%	0	0
SamplePlay	100%	0	Pedal

InputMode (Loop/Delay = In, Pedal Only)
InputMode behaves just like LoopMode except that a connected Feedback pedal controls the input volume in the states where the input is open. Feedback is only controlled by the front panel Feedback knob. This InterfaceMode does not exist without a Pedal connected.

State	Feedback	Loop Input	Loop Output
playing	Knob	0	100%
overdubbing	Knob	Pedal	100%
substituting	0	Pedal	100%
recording	NA	Pedal	0
multiplying	Knob	Pedal	100%
inserting	0	Pedal	0
replacing	0	Pedal	0
mute	100%	0	0

OutMode (Loop/Delay = Out, Pedal Only)
OutMode is similar to LoopMode, except the pedal controls the Loop Output level instead of feedback. The feedback is always set from the front panel knob. This InterfaceMode does not exist without a Pedal connected.

State	Feedback	Loop Input	Loop Output
playing	Knob	0	Pedal
overdubbing	Knob	100%	Pedal
substituting	0	100%	Pedal
recording	NA	100%	0
multiplying	Knob	100%	Pedal
inserting	0	100%	0
replacing	0	100%	0
mute	100%	0	0

ReplaceMode (Loop/Delay = rPL, Pedal Only)
In ReplaceMode the pedal controls Loop Output Volume and Feedback simultaneously. The result is similar to LoopMode, except that you hear the reduction for feedback immediately instead of on the next loop pass. This makes it easier to "sculpt" the loop using the feedback control. If you have Overdub on, the pedal serves as a Replace function with smooth level control. So you can add new material and smoothly drop out the loop underneath you as it is overdubbed, allowing for a smooth real-time replace. The disadvantage of ReplaceMode is that if the pedal is in the toe-up position, the loop output is also zero, so it is not possible to create loops with only single repetitions. It is also less interesting for any loops relying on reduced feedback settings, since the output is affected. This InterfaceMode does not exist without a Pedal connected.

State	Feedback	Loop Input	Loop Output
playing	Pedal	0	Pedal
overdubbing	Pedal	100%	Pedal
substituting	Knob	100%	100%
recording	NA	100%	0
multiplying	Pedal	100%	Pedal
inserting	0	100%	0
replacing	0	100%	0
mute	100%	0	0

FlipMode (Loop/Delay = FLI, Pedal Only)
FlipMode is an unusual and interesting InterfaceMode, in that the pedal controls both Loop Input and Feedback simultaneously. The interesting thing is that Feedback on the pedal is reversed! When the pedal is all the way in the toe-up position, the Loop Input level is zero and the feedback is at 100%. When the pedal is all the way in the toe-down position, the loop input is at 100%, but the feedback goes to 0. In use this is like a Hold pedal, but with a more fluid action. Or you can think of the pedal as a “soft replace” since operating the pedal lets you smoothly crossfade a replacement section into your loop.

To get the hang of this InterfaceMode, use loops of about 1 second, keep the pedal in the toe-up position most of the time, and turn the front panel Feedback knob to zero. In Overdub the Feedback is taken from the front panel knob (reversed!), so once you've made an interesting loop by crossfading in Play, you can keep it by going to Overdub. You can then do overdubs onto it using the pedal to control the input volume. By setting the front panel feedback knob you can make 'Overdub' into a simple delay, which can be used as a contrast to the unusual crossfade effect. Multiply allows you to continue to crossfade over repetitions of your loop. Insert is a Single Cycle Multiply (as described under the StutterMode), so you can Overdub onto repeats of the next Cycle in the loop. (Remember you can hit Undo to end the Insert and not keep the stutters in the loop). Mute allows you to build up a crossfaded loop without hearing it and

then bring it in at once. To record an ordinary loop in this InterfaceMode, put the pedal in the toe-down position and end Record with Overdub. This InterfaceMode does not exist without a Pedal connected.

State	Feedback (reversed)	Loop Input	Loop Output
playing	Pedal	Pedal	100%
overdubbing	Knob	Pedal	100%
substituting	0	Pedal	100%
recording	NA	Pedal	0
multiplying	Pedal	Pedal	100%
inserting	Pedal	Pedal	100%
replacing	0	100%	0
mute	Pedal	Pedal	0

Using the New InterfaceModes with a Stereo Echoplex Setup
Many of the new InterfaceModes require a pedal connected to the feedback pedal jack in order to be available at all, and use that pedal as a key part of their functionality. This causes a problem with the traditional Echoplex Stereo setup, where a pedal is only connected in the Master side, and all value changes made with it are sent to the Slave Echoplex by MIDI. With the new InterfaceModes, the slave in such a setup will not have a pedal inserted and it can not be set to the new InterfaceModes. It will only cycle through the first four, while the master cycles through all 8. (This was also true in LoopIII and found to be unsolvable, but we only had OutMode where it happened so it didn't affect many people.)

There are a few ways to work around this problem.

One way is to use a stereo volume pedal for the feedback pedal jack. Connect the two channels of the pedal to the two Echoplexes, and then each will have a pedal inserted with the control coming from the same place. Both can be set to all InterfaceModes, and be controlled appropriately. If you are really picky, you may find that your pedal is not exact between channels, so you may find values are not set exactly the same between the two Echoplexes. Fixing this will either mean finding a pedal that is better matched, or soldering a wire between the two potentiometer wipers inside your pedal to force them to have the same voltage. (This makes the pedal useless as a true stereo volume pedal, so make sure you understand what you are doing before attempting such a modification.)

A second approach is to use a single mono pedal connected to both units with a Y connector. For this to work, the potentiometer in the pedal must be half the resistance of the pedal requirement for a single Echoplex. This means it will have to be approximately 10Kohms or greater, but you may need to experiment a bit to find a pedal that uses the full range in the best way. For some Echoplex units, values as low as 5Kohm work better.

A third way which is less useful is to insert a dummy connector into the Feedback Pedal jack of the slave, without connecting it to anything. Then you will be able to at least set the slave to any of the InterfaceModes and use some of their functionality, but on the slave you will not be able to control some parameters that the pedal controls in some of the InterfaceModes. Any Feedback settings controlled by the pedal will be sent by MIDI from the Master, so InterfaceModes that use the pedal for Feedback will work fine. But Input and Output level will not be transmitted, so any InterfaceMode that uses the pedal for those will not work very well this way for Stereo.

NextOverdub and SimpleCopy

Next-Overdub with SwitchQuantize still engages Overdub in the next loop as it did in LoopIII, but if this next loop is empty (reset), it automatically creates a full copy of the previous loop and lets you Overdub as it copies. It does not continue adding multiples as the normal copy function does. This function terminates by itself at the end of the loop. We call this function SimpleCopy**.**

Double-Click – SmartButton Presses

Double-Click Copying
When you are triggering loops with MIDI and SwitchQuant is on, repeating (double-clicking) the same note in the lame duck period will do a copy into the next loop. This is similar to using Next-Multiply, but is much quicker and simpler to use from a keyboard.

Double-Click Record
If the current loop is in reset, and this double-click note is for the current loop, we Record onto it.

Double-Click Multiply
If we have a loop playing and we double-click the current loop, we do a Multiply.

Double-Click StopRecord and StopMultiply
If Record or Multiply is going and we press the same note again, we end the Record or Multiply and do nothing else.

Long Press Reset
If you do a long press on the note for the Loop you are currently in, the loop is reset. This is not the case when SamplerStyle = Att. In that case the currently loop is just retriggered and played as normal for SamplerStyle=Att.

With this feature you have most functions under control simply by connecting a MIDI keyboard or other MIDI controller programmed with the notes from 84 to 99.

MIDI pipe

All incoming MIDI commands received at the MIDI In port are monitored and selectively sent to the MIDI Out port, depending a bit on the state of the Echoplex. We call this “Piping”. MIDIpipe makes it possible to have multiple Echoplexes chained together and easily switch between using them independently or together as stereo pairs. It also makes it simple to work with multiple devices sharing the same MIDI clock. It is similar to a MIDI merge function, except with Piping the Echoplex is intelligently deciding which things should be piped to the output or not. A Merge function would send everything through, which can cause major problems in some situations. MIDIpipe intelligently prevents such problems.

MIDIpipe works with very low latency, so you will not notice any significant difference between a command that is piped and one that went direct.

MIDI Sysex commands are also piped immediately, which is convenient for controlling multiple Echoplexes from a PC using a sysex librarian utility. Unfortunately, while piping Sysex commands the audio is stopped due to the complexity of handling the sysex commands in real time. This is not audible for short commands like changing a single parameter, but you may hear it for long Sysex strings.

MIDI clock is piped when Sync = In or Off. It is not piped when Sync = Out or OuS, because the Echoplex will also be generating MIDI clock internally. If they both went out you would have double clocks, so MIDIpipe prevents that.

StartSong, StopSong, and Continue messages are piped in all cases, so an external controller at the beginning of the MIDI chain can send commands to a sequencer that is getting clock from the Echoplex. When the Echoplex is piping Start, Stop, and Continue messages it intelligently checks whether it has already sent the message itself within the last 10ms, and does not pipe if it has. This prevents multiple Echoplexes in parallel from stacking up the StartSong messages when they are all Recording simultaneously.

Echoplex functions that normally send out a MIDI command do not duplicate the command if it is being piped. For example, if a MIDI command is received for Record, the command is piped to the output and used internally to start the Record function. The Record function does not then send another MIDI command for Record as it normally would, so the commands do not get doubled by piping.

QuantStartPoint

When an external clock is present and Sync = In, a long press on StartPoint will reset the internal Loop StartPoint to the Global StartPoint defined by the clock. See the Synchronization section for more details.

Load and Dump

The SampleDump function has new features added that allow it to work with a much wider variety of external devices. See the SampleDump chapter for more information.

There is a whole new system for Sysex parameter editing, allowing you to save and recall all of the parameters, individual Parameter Presets, or individual parameters in a given preset. This is described in detail in Parameter Editing chapter and in the Sysex chapter.

Chapter 2: Function Improvements

Many existing functions have been enhanced or improved over the way they worked in LoopIII.

Improved Reverse and Undo while Reverse

It is now possible to Undo while in Reverse! Undo during Reverse now works normally back to the point where Reverse happened. So any overdubs you do while Reversing can be Undone, which was not possible in LoopIII. However, it is still not possible to Undo past a reverse, since memory really does get used in the other direction and overdubs made prior to reverse get destroyed by ones made after the reverse.

Since Undo is now available in Reverse, the Reverse indicator is no longer the Undo LED and pressing Undo no longer puts the loop back in Forward. That never made much sense anyway, so now Reverse is indicated by the Insert LED.

Record-Undo is also possible in Reverse now. So if you start Record by mistake while in reverse, pressing Undo returns you to where you were. In LoopIII that did not work. As a consequence, starting Record does not automatically force you to be Forward anymore, but leaves it in reverse.

All of this means that Reverse and Forward are really equal now, with the exception that the green Multiple display counts backwards when you are in Reverse.

Multiply-to-x and Insert-to-x CrossFunction Improvements

Rounding for CrossFunctions
In LoopIII, the Multiply-to-x and Insert-to-x CrossFunctions always created an UnroundedMultiply or UnroundedInsert. In other words, if you had Multiply or Insert going and ended them with a different function, they stopped immediately with the loop length redefined to that point and then went to the new function. Now those actions are rounded, which is more consistent with the typical operation of Multiply and Insert. This means when you end a Multiply or Insert with a different function, they round off to the next Cycle point and then begin that function, similar to if you had ended them with a second press of Multiply or Insert. The Cycles created by Multiply and Insert are not lost as they were before.

Forcing UnroundedMultiply while Rounding
UnroundedMultiply can now be executed while rounding. This means that in addition to doing an UnroundedMultiply by pressing Multiply to start and ending with Record, you can press Multiply to start, then Multiply again to finish multiplying and start rounding, and then press Record while it is rounding to force it to stop Unrounded. This is especially interesting when you use alternate functions to end Multiply. Since ending Multiply with an alternate function now does a rounded ending, pressing Record after that forces it to go into that function immediately and end the multiply Unrounded.

For example, you could press Multiply to multiply your loop out as you add something over it, press Reverse to end the multiply and start it rounding, and then press Record to have it immediately start Reversing with the loop length redefined to that point.

Or, you could chop out a short reversed snippet of your current longer loop in a new loop. With SwitchQuant on, you press Next-Multiply-Reverse-Record to create a reversed snippet. The Next-Multiply begins a copy of the current loop into the new loop (which is really the same as a multiply into the new loop), The Reverse starts it rounding with the Reverse command armed, and the Record executes it immediately and redefines the new loop at that length.

The Hidden UnroundedMultiplyReverse Function
In LoopIII the limitation with Multiply cross functions that forced us to do them Unrounded resulted in one function that some users liked. If Mutliply was going and overdubs were being made over the top, ending with a single button press of Reverse immediately changed the length of the loop to that point and instantly put the resulting UnroundedMultiply into reverse. This was an interesting use, but after we made the improvements that allowed us to do Multiply cross functions Rounded it was only possible by doing two button presses. (With two button presses, you do it like this: end Multiply with Reverse, then press Record during the Rounding period to set the new loop length and immediately have it in reverse.)

So for the user who lost this favorite function, we found a way to give it back to him. It is a little bit obscure in operation, so bear with us. Set MoreLoops to some number greater than 1. Set SwitchQuantize on to Confirm. (any quantizing value works, but it is easiest with Confirm.) Record your basic loop in Loop 1. Press NextLoop. You will go into the "Confirming" state, where it waits for another function press before it goes into the next loop. Press Multiply down and hold it down. This will immediately put you into the next loop, copying the audio from the first loop. Overdub as much as you want over multiples of your first loop. When you are ready, release Multiply. The loop in Loop 2 will immediately stop copying, reset the loop length to that point, and go into Reverse! So there is the old UnroundedMultiplyReverse function, now properly called NextUnroundedSUSMultiplyReverse.

When InsertMode is equal to SUS, this function is disabled in favor of the usual UnroundedMultiply SUS function.

Undo Improvements

Undo can now be executed even when the Undo LED is not green. It will be executed as soon as the Undo LED lights up, so you can easily Undo the maximum possible without struggling to press Undo at the right moment. This is an improvement over LoopIII, where you could have a loop with only a short area containing an overdub that was undoable. The Undo LED turns on and off in such a case to indicate when you are in the Undoable section. In LoopIII you could only tap Undo during that time for it to work. Now you can tap it any time and it will be done for you.

Undo now checks whether there may be a whole loop length in memory without changes. If there is, Undo acts twice when you press it. This eliminates the frustrating cases where you sometimes pressed Undo and it appeared that nothing had happened. So you had to press Undo twice to get rid of a bit you had just listened to. This makes Undo feel a lot more responsive for some users who had difficulty with it before.

These improvements for Undo are true for both the Short-Press and Long-Press versions of Undo.

Exiting from SamplePlay to Play with Undo

SamplePlay is when a loop has been triggered like a sampler, where a trigger starts the loop at the beginning and it plays once and goes back to mute. Repeated triggers will retrigger the loop. In LoopIII pressing Undo during a SamplePlay just immediately stopped it and put you into mute. Another button press was required to go back to the normal play state, which was never very convenient.

In LoopIV this has been improved. Pressing Undo during a SamplePlay now puts you seamlessly back into Play, so your loop keeps going instead of stopping at the end. This is really useful if you are doing a lot of retriggers for stutter effects, and then finally decide to let the loop keep playing. You just have to press Undo and it seamlessly continues!

Next-to-X CrossFunction Undo Improvement

When Undo is pressed while overwriting a loop with Next-Record, Next-Multiply, or Next-Insert, the target loop is now recovered.

In LoopIII if you did a Next-Multiply to copy loop 1 into loop 2, and then changed your mind and pressed Undo, it correctly returned to loop 1 but left loop 2 in Reset. The previous contents of loop 2 were lost. Now it is correctly preserved.

Simultaneous Overdub

Overdub can now can be held on in a sustain fashion with a long press while simultaneously pressing other buttons to execute other functions. This is true whether Overdub is activated from the front panel buttons, from the foot pedal, or with a momentary switch in the Overdub jack. In LoopIII this did not always work. Overdub would sometimes get stuck on when another button was pressed.

For example, now you can keep Overdub long-pressed and then press Reverse simultaneously to go in and out of Reverse while Overdubbing.

Simultaneous Overdub is useful when using Overdub as a SUS function. Note this only works with the Overdub button, and no other functions. Also, it does not work to do long-press functions on other buttons while holding Overdub. They will be treated as short-presses.

8ths/Cycle and Sync Parameter Changes Occur in Rhythm

Now when a loop is playing and the 8ths/Cycle or Sync parameters are edited, the change of value is only activated at the first Loop StartPoint *after* you come out of the Parameter editing state. At that point you jump directly to the new selected value. This means the value change occurs only while back in the playing state, and only at a rhythmically sensible point. This helps eliminate any confusion when working with a synchronized sequencer and makes for much smoother transitions into new time signatures.

Try changing 8ths/Cycle with sync = Out and a sequencer slaving to the clock. You control the sequencer's tempo in relation to your loop!

Also, the 8ths/Cycle parameter has new values, allowing you to have up to 256 8th notes in a Cycle. This gives a lot more flexibility for how the Echoplex synchronizes to external devices.

When editing, the most important values come first to make them easy to select: 8,4,2,6,12,16,32,64,128,256, then it goes on with 1,2,3...96. Note that with the new data wheel feature, the top of the knob range ends at 54 instead of 96. This was done because we found it was easier to set the more typical values when the knob resolution was limited a little bit. To reach the values between 54 and 96 you simply use the front panel button to continue incrementing the number in the usual way. A long-press while editing the Parameter returns you to the initial value of 8.

Chapter 3: Display

The Display has been enhanced in a variety of ways to make operation easier and give more information to the user.

Visual Tempo and Sync Guide

It is often difficult in looping to feel the length of a loop before there are any good rhythmic clues recorded into it. This can make it frustrating to overdub new material that is intended to be in rhythm. To aid in this a new visual tempo guide has been added to help the user find the tempo.

Timing LED: blinks with the sub-Cycles (8th notes) defined by Parameter 8ths/Cycle and the global clock.

Keys LED: blinks at local Cycle StartPoints.

MIDI LED: blinks with the local Loop StartPoint. (only shown if multiples established)

Multiple Right Dot: Global MIDI StartPoint (1 of the sequencer). only shown if not aligned.

Multiple Left Dot: Sync correction happens, as follows:

bright: sync came early (jumps back almost the whole loop)
this means the external sequencer was a little fast

faint: sync came late (jumps a little)
this means the external sequencer was a little slow

Loop Green Dot: AutoUndo executed (loop was not changed that pass.)

When Sync = In and an external clock source is present, the Tempo LEDs will initially reflect the StartPoints defined by the external clock.

If the loop is shifted out of alignment with the external clock, the Tempo LEDs will then reflect our internal loop StartPoints. The Global MIDI StartPoint LED (lower right dot on the Multiple display) will then blink in time with the StartPoints of the external clock. This gives a visual indication of how the loops are aligned. When a ReAlign is done to bring them back together, the Global MIDI StartPoint LED stops blinking.

The LED for 8th note sub-cycles counts on 8th notes determined by the global clock. The cycle LED, on the other hand, blinks at the local StartPoints. If multiple loops are used and the loops are switched unquantized, it is possible to see these move out of alignment with each other. This can be a little disconcerting, but it can also be helpful as a reference of where the global clock is in relation to your cycle points as you switch loops.

If the tempo is above 400 BPM, the 8th note sub-cycle LED stops blinking since it becomes useless as a visual indicator at such speeds. This is also the point where MIDI clock is no longer sent for similar reasons. You can still make loops as short as you like.

The Multiple Left Dot showing the Sync correction can be useful for tuning the tempo on a sequencer to match with an existing loop on the Echoplex. By watching the frequency and intensity of this LED you can quickly speed or slow the tempo of the sequencer to match loop in the Echoplex, at which point the dot disappears. This technique allows you to start a loop without the sequencer, then start the sequencer and tune its tempo to match.

SyncTime

After the second Sync pulse is received, or a sync is established by MIDI clock, the resulting Cycle time is displayed on the LoopTime display. This only appears while in Reset.

Feedback and Continuous Controller Display

Changes to FeedBack are displayed briefly on the LoopTime display while they are being changed. The value appears as a red number (0 - 127) in place of the loop time. The display shows the change whether it is made by the front panel knob, foot pedal, or through MIDI. MIDI Loop volume control is displayed in the same way. This is very helpful in controlling these values, since it is often difficult to tell exactly what you have set when using a foot pedal. In the case of feedback you don't know the result until the next repetition of the loop, which can be frustrating if you didn't really set it where you wanted. The visual display makes this much easier to manage.

Command Display

Several new functions that do not have their own obvious LED indicator are displayed briefly with some letters on the red display. These are:

rE	Reverse
Fd	Forward
H.SP	HalfSpeed
F.SP	FullSpeed
LOA	Load Preset
SAF	Save Preset
RES	Revert Preset to default
AL	ReAlign
St.S	QuantMIDIStartSong
S.Pt	StartPoint sent
cS.P	QuantStartPoint
S.Un	Short Undo
L.Un	Long Undo

These become especially useful now with the new MIDI commands that can directly access so many functions.

Memory Size Display

The size of the memory is only shown for a short time after startup and after GeneralReset. To see it again, just press Multiply in Reset.

Chapter 4: New Parameter Settings

Loop/Delay Parameter Settings

Changes to the Loop/Delay Parameter:

LoopMode	LOP
StutterMode	Stu
DelayMode	dEL
ExpertMode	EXP
InputMode	In
OutMode	Out
ReplaceMode	rPL
FlipMode	FLI

See the New Functions Chapter for more details about these.

Quantize Parameter Settings

Two new values are added for Quantize, Loop and 8th.

Quantize = Loop
This quantizes a function to the StartPoint of the entire loop, as opposed to quantizing to the next Cycle point if multiply or insert has been used.

Quantize = 8th
This quantizes functions to the subdivision of a Cycle, normally determined by the 8th/Cycle (or Beat) parameter and the loop length. The exception is when Sync=In and a MIDI clock is being received. MIDI clock specifically defines 8th notes, so the MIDI clock information is used for Quantizing to 8th notes in this case. If the loop is too short it does not quantize at all. This happens when the tempo is above 400 BPM, and similarly means no MIDIclock is generated and the Timing LED does not flash. See the LoopDividing section in the New Loop Functions chapter for more details.

Quantize = CYC
This is the same as what used to be Quantize = On. This is now the setting for quantizing functions to the next Cycle point.

Quantize = Off
Same as before, this means there is no quantizing and all functions are activated immediately.

8ths/Cycle parameter (8ths/Beat on older units)

The 8ths/Cycle parameter has new values, allowing you to have up to 256 8th notes in a Cycle. This gives a lot more flexibility for how the Echoplex synchronizes to external devices.

When editing, the most important values come first to make them easy to select: 8,4,2,6,12,16,32,64,128,256, then it goes on with 1,2,3...96. Note that with the new data wheel feature, the top of the knob range ends at 54 instead of 96. This was done because we found it was easier to set the more typical values when the knob resolution was limited a little bit. To reach the values between 54 and 96 you simply use the front panel button to continue incrementing the number in the usual way. A long-press while editing the Parameter returns you to the initial value of 8.

When a loop is playing and 8ths/Cycle is edited, the change of value is only activated at the first Loop Start *after* you come out of the Parameter editing state. See the Synchronization chapter for more details.

Sync

Changes to the Sync Parameter:

Value	Display	Description
Off	Off	no Sync
OutManualStartSong	OuS	sends Clock, but does not send StartSong at Record
SyncOut	Out	sends Clock plus StartSong and StopSong messages
SyncIn	In	receives Sync

See the Synchronization chapter for more details.

RecordMode = Safe

This is a new value for RecordMode, and is just like RecordMode = toggle except that after a Record the feedback is always set to 100%. This will be regardless of the feedback knob or pedal position. When the feedback is changed the Echoplex starts to respond as normal. This setting is useful for people who change feedback but then tend to forget to set it back to 100% before recording a new loop. This can be frustrating if you recorded something perfectly and then a little while later realize it is gone because you left the feedback down. RecordMode=Safe is meant to protect you from that.

The disadvantage is that if you want to start a loop with the feedback down, you can't do it with Safe. In that case you would probably just continue to use RecordMode = Toggle.

RecordMode = SAF is disabled during DelayMode.

InsertMode

Changes to the InsertMode parameter:

Insert Only	Ins	same as before
Sustain	SUS	new sustain action for Multiply and Insert
Rehearse	rhr	same as before
Replace	rPL	same as before
Substitute	Sub	new function, similar to Replace
Half Speed	h.SP	new function, changes speed of the loop
Reverse	rEV	same as before

See the New Functions chapter for more information on the new functions in InsertMode.

MoreLoops

There are 16 loops available now (instead of 9 as in LoopIII). On the Loop display the numbers above 9 are shown with letters, due to the lack of a leading 1. So they go 1, 2, 3…9, A, b, c, d, E, F, G.

LoopTrig

The default value for LoopTrig is now 84 to make room for the expanded MIDI commands.

SwitchQuantize

Changes to the SwitchQuantize parameter:

Off	off	
Confirm	CnF	waits for an additional button press to jump
CycleQuantize	CYC	jumps at the next Cycle end
ConfirmCycle	CCY	New: behaves like Confirm but additionally quantizes to Cycle
LoopQuant	LOP	Jumps at the next loop end
ConfirmLoop	CLP	behaves like Confirm but additionally quantizes to loop end

See the New Functions chapter for more details.

SamplerStyle

Changes to the SamplerStyle parameter:

Run	run	same as before.
Start	StA	New: Starts the next loop from the beginning. see New Functions chapter for details.
Once	One	New automatic return function, see New Functions chapter for details.
Attack	Att	same as before with MIDI notes. Acts like “run” with NextLoop button.

See the New Functions chapter for more details.

Chapter 5: Synchronization

SyncRecord

SyncRecord is a variation of Record that is done when a Sync of any type is being received and Quantize is off. Instead of always quantizing Record when a sync is coming in as was done in LoopIII, we now do a kind of "Multiply over nothing" for the unquantized case. This means SyncRecord starts immediately when you press Record, counts the Cycles on the green display, and rounds at the end to fit the loop time defined by the sync. This gives freedom from quantization, while still allowing tight synchronization to an external clock source.

During Reset, the Overdub LED turns yellow to indicate that a Sync arrived. When the second Sync point arrives to define the loop length, the display shows the resulting Cycle time. Whenever the Overdub LED is yellow like this, the next Record press will be a SyncRecord.

With SyncRecord, you only need to have received the first sync event to begin Recording. As you are Recording, the Echoplex will continue watching the sync to determine what the right cycle times are.

When Quantize is on SyncRecord does give us one improvement. In LoopIII, as a loop was recorded in sync the cycles would not be counted or differentiated. Now the Cycles are tracked and counted properly. This means that if the incoming clock defines a Cycle length of 2 seconds and you let Record continue to 8 seconds, you will see the multiple counter counting from 1 to 4. The Cycle boundaries will be set at 2.0 seconds. In LoopIII it would have just made a big 8 second cycle. This is especially useful now since the Quantize function can quantize to either the Cycle boundaries or the Loop StartPoint.

Global/Local MIDIclock

There is a new Global/Local MIDIclock system that allows the slaved Echoplex to lock up with an external sequencer without drifting, even if the loop start point on the Echoplex is moved by one of the following de-aligning functions:

- StartPoint
- Sample triggering
- Mute-Undo (restart loop)
- Reverse
- HalfSpeed
- Next (Each loop keeps its own local MIDIclock counter)
- stopping and restarting the sequencer or drum machine

This lets you shift the loop freely away from the downbeats of an external sequencer for interesting rhythmic effects. All the while, the Echoplex keeps track of the external sequencer's downbeat and clock as a "Global" clock, and the local Loop's StartPoint as a "Local" clock. This allows for incredible new capabilities in the Echoplex to shift loops out of alignment from each other without losing sync, and then ReAlign them perfectly with each other again at will!

After de-aligning the loop, the original alignment of the Loop StartPoint with MIDI beat 1 can be restored with the new ReAlign functions or with reset. ReAlign is available as the Mute-Multiply combination on the front panel, or directly by MIDI. See ReAlign section below for more details.

All the following functions serve to bring the loop together with the sequencer again, once the loop has been de-aligned by one of the functions above:

- Reset (start a new loop with the StartPoint at MIDI beat 1)
- ReAlign (Restart the current loop at the next MIDI beat 1)
- MuteReAlign (same as ReAlign, but the loop is muted while waiting to ReAlign)
- QuantStartSong (stop the sequencer and restart it at next Loop StartPoint)
- MuteQuantMIDIStartSong (same as QuantStartSong but loop is muted while waiting)

Each of those functions is available separately from DirectMIDI (in any state, not just Mute.)

From the front panel the traditional cross function Mute-Multiply offers some of those functions, depending on the Sync parameter and Sync events coming in to the Echoplex.

ReAlign

The new ReAlign Function
ReAlign allows the loop to be aligned again with an external loop or sequencer that it was in sync with, after it has been brought out of alignment by the use of functions like Reverse, Triggering, HalfSpeed, etc. This is very useful to allow rhythmic variations by shifting loops to run out of phase with each other, and then perfectly recover to have them back in perfect sync. It is also very helpful when composing with a sequencer, where you frequently need to stop the sequencer and restart it. Being able to restart them in sync becomes necessary. ReAlign works when the Echoplex is the clock master or the clock slave.

ReAlign makes use of the Global-Local MIDIclock system noted above. The Echoplex uses that to keep track of the "Global StartPoint" defined by an external sync source and our "Local StartPoint" as they are split apart. ReAlign essentially gives you a variety of simple ways to bring them back together again.

Accessing ReAlign Functions from the Front Panel with Mute-Multiply
Mute-Multiply is the button combination to do ReAlign from the front panel. When using this combination, the loop is first put into Mute, and then the ReAlign is armed by pressing Multiply. When the next Global StartPoint comes, the ReAlign is executed. (There are also several new MIDI commands for ReAlign, detailed below.)

What exactly happens at that point depends on the setting of the Sync parameter and whether or not we have clock. The Display will show what specifically happens, whether it is ReAligning our loop to an external sync, or sending out a StartSong to an external sequencer. A ReAlign of the local loop is displayed as "AL" and sending a StartSong is "St.S." The Multiply LED will be red during this waiting time, to indicate the ReAlign is armed.

If you want to cancel ReAlign after it is armed, another press of Multiply switches it off and we don't wait for anything anymore. This is helpful if you find yourself stuck there with no sync coming.

Here are the different cases that can happen with the Mute-Multiply state and various sync conditions:

Sync=In:

Sync Event	Function	Description
BeatSync	TriggerSample	The loop is triggered to play once from the start. Repeated pulses on BeatSync retrigger the loop for stuttering effects. (same as in LoopIII.)
BrotherSync	MuteReAlign / No reaction	The loop is triggered out of Mute to play from the start and continue playing. This ReAligns the loop while BrotherSyncing to other Echoplexes. If a MIDI clock has been received BrotherSync is ignored.
Global MIDI StartPoint	MuteReAlign	If MIDI clock is already being received when Mute-Multiply is pressed, the Echoplex waits for the global MIDI StartPoint and then retriggers the loop out of Mute.
MIDI StartSong	MuteReAlign	If no MIDI clock is present and a MIDI StartSong is received, followed by MIDI clock, the loop triggers immediately from the start and continues playing in sync.
Local StartPoint	N/A	No reaction to Local StartPoint

Sync=Out:

Sync Event	Function	Description
BeatSync	N/A	No reaction to BeatSync
BrotherSync	MuteReAlign	The loop is triggered out of Mute to play from the start and continue playing. This ReAligns the loop while BrotherSyncing to other Echoplexes.
Global MIDI StartPoint	N/A	MIDI clock input is ignored when Sync=Out
MIDI StartSong	N/A	MIDI StartSong input is ignored when Sync=Out
Local StartPoint	N/A	No reaction to Local StartPoint when Sync=Out

Sync=OutManualStartSong (OuS):

Sync Event	Function	Description
BeatSync	N/A	No response to BeatSync
BrotherSync	N/A	No response to BrotherSync
Global MIDI StartPoint	N/A	MIDI clock input is ignored when Sync=OuS
MIDI StartSong	N/A	MIDI StartSong input is ignored when Sync=OuS
Local StartPoint	MuteQuantStartSong	A StartSong is sent at the next Loop StartPoint so the sequencer aligns to the Echoplex. The Loop comes out of Mute at the same time so both start together, in sync.

Accessing ReAlign Functions with MIDI

There are several new MIDI commands in support of ReAlign, which give us much more flexibility than we have from just the front panel. With MIDI the ReAlign commands can be accessed at any time with single button press, and don't necessarily require going into Mute first. (See the MIDI commands section for more info on other new MIDI commands.)

QuantMIDIStartSong (source# + 40)

QuantMIDIStartSong waits until the next Loop StartPoint, and then sends a StartSong message out the MIDI port. The Loop continues playing the whole time, and "St.S" is displayed during the waiting period to indicate what is happening. QuantMIDIStartSong is useful when the Echoplex is the clock master and the external sequencer has been stopped. Now the sequencer can be restarted right in time with the loop. QuantMIDIStartSong can be executed at any time, independent of the Sync parameter setting. This allows you to always have a way to send a StartSong message in sync with the loop StartPoint, even if the Echoplex is not the clock master.

MuteQuantMIDIStartSong (source# + 41)

When the MuteQuantMIDIStartSong command is received, the Loop is muted instantly and then waits until the next StartPoint. The display will show "St.S." When it reaches the StartPoint a MIDI StartSong message is sent out the MIDI port to start up the sequencer and the Loop comes out of Mute. This is similarly useful for when the Echoplex is the clock master and we need to restart the sequencer. In this case, both the loop and the sequencer can be muted and brought back on together. MuteQuantMIDIStartSong works the same as the Mute-Multiply combination from the front panel when Sync is set to OuS, but provides a more direct access since it only requires a single command instead of the two button combination. Unlike the front panel function, MuteQuantMIDIStartSong works for any setting of the Sync parameter and can be executed at any time. This allows you to always have a way to send a StartSong message in sync with the loop StartPoint, even if the Echoplex is not the clock master.

MIDIReAlign (source# + 38)

MIDIReAlign executes ReAlign with a single MIDI command. The loop is not muted prior to the ReAlignment. If Sync = In, the display shows "AL" until the Global StartPoint defined by the sequencer's clock arrives, at which point the loop is retriggered automatically from its StartPoint. When Sync = Out, the loop is retriggered when a BrotherSync is received.

MIDIMuteReAlign (source# + 39)
MIDIMuteReAlign is exactly like using ReAlign from the front panel, except it directly does the function without requiring double button combinations. When the command is received, the loop is muted and "AL" appears on the display. If Sync = In, when the next Global StartPoint arrives the loop is retriggered back in time with the external sequencer. When Sync = OUS, the loop is retriggered when a BrotherSync is received.

BrotherSync and ReAlign
As shown in the tables above, ReAlign also works when BrotherSyncing two or more Echoplexes together. When one Echoplex has had it's StartPoint shifted off from the other, ReAlign can bring them back together. Send a ReAlign command to one of the units and it will wait for a Sync with "AL" on the display. When the BrotherSync pulse comes from the other Echoplex, it will retrigger its loop so they are both back in alignment with each other. BrotherSync is usually done with Sync = Out, so ReAlign to BrotherSync is available there as shown in the table above.

There can be some confusion when De-Aligning and ReAligning to BrotherSync with more than two Echoplexes. If you have two Echoplexes in BrotherSync and shift one unit's StartPoint away from the other, you will have them both sending pulses on the BrotherSync line at different times. Those two are able to tell which pulse is theirs and which is from the other unit, so they can stay in sync with each other just fine. They can be ReAligned later at the user's command. However, if you try to add a third unit to the sync it will see the two different pulses coming and be unable to tell the correct loop StartPoint and loop length. It will not be able to SyncRecord to the others correctly. The best way to deal with this is to have the third unit record a loop in sync before having any of them shift their StartPoints. Then it will be in sync already and not have to worry about joining the sync later. Or ReAlign the other two first before the third unit joins.

An alternative would be to use the QuantStartPoint command on the unit that has been shifted, so that it will reset its StartPoint to the next BrotherSync coming in from the other Echoplex. Then they will both have their StartPoints in the same spot and be sending BrotherSync pulses at the same time so there will not be any confusion for the third unit. Of course this means ReAlign can no longer be used on the second machine to put it back in the original alignment with the first.

StopSync

If a sync has been received, pressing Overdub in Reset switches off reception of incoming Sync events. This is called StopSync**.** The Overdub LED turns red to indicate that sync reception is disabled. StopSync is useful if you want to disregard sync for the next Record.

Another Overdub press in reset or GeneralReset makes the Echoplex receptive to Sync again. This is called ContinueSync. The Overdub LED is turned off until the next sync comes. So after doing a Reset you continue in sync with the incoming clock, but with an Overdub in Reset you can then Record a new Cycle length, unrelated to the external clock. (This is mainly useful for working with other devices that send clock all the time, like Yamaha products.)

StopSync also disables Tempo, so you can temporarily escape from a Tempo you have defined and then return to it.

SyncStartPoint

Pressing Undo in Reset when Sync = In defines a new Global sync start point just like switching on a MIDI clock or sending a StartSong to the Echoplex does. The Global MIDI clock counter is also restarted at this point. Reset does not do this.

When it is pressed, we also send out a StartSong message. This allows you to have a clock source upstream of the Echoplex, and be able to stop a downstream device and restart it in alignment with the Echoplex StartPoint.

If the Echoplex is in a state where we are waiting for a sync, pressing Undo stops it from waiting so recording can be done normally.

SyncStartPoint is also useful when working with units that send clock all the time. (like Yamaha).

Tempo Select

There is a new way to define the tempo of a loop in advance in BPM. This is done by pressing the Undo button in Reset to enter the Tempo Select state and then using the FeedBack knob to set the tempo before a loop is recorded. When this is done a MIDIclock can be sent out and a sequencer or drum machine can be started in tempo. Then a loop can be SyncRecorded to it so that everything matches the preset tempo.

For more details on the new Tempo Select feature see the description under the New Loop Functions section.

StartSong, StopSong, and Continue

A new value for the Sync parameter allows the user to select how MIDI StartSong and StopSong messages are sent when the Echoplex is the clock master. This choice can be very useful in different circumstances when controlling external sequencers. For example, some sequencers need to have the clock sent in advance to set the tempo, with the StartSong sent later. In other cases you may want to have the StartSong sent immediately with the MIDI clock when the loop is recorded so that a sequencer starts right up with the clock.

The possible Sync parameter settings are as follows:

Sync = Out – StartSong and StopSong messages are sent in most instances. MIDI clock is also sent based on the Cycle length of the loop and the setting of the 8^{th}/Cycle parameter.

Sync = OutManualStartSong (OuS on the display) – MIDI clock is sent, but no StartSong or StopSong messages unless the user specifically commands it.

Sync = In – MIDI clock, StartSong, and StopSong messages are only received, not sent. They are piped through.

Sync = Off – MIDI clock, StartSong, and StopSong messages are neither sent or received. They are piped through.

The exception is for one situation where we send StopSong in both Out and OuS cases: When the clock is stopped at Reset we send a StopSong. This avoids the case of the sequencer or drum machine waking up in the middle of a pattern when the clock happens to get restarted again.

The following lists all cases where StartSong and StopSong messages are sent, depending on the state of the Sync parameter:

If Sync = Out
We send StartSong at:
- StopRecord (finishing a loop record)
- Start SyncRecord (in case we come from Tempo Select)
- SetTempo in Tempo Select
- UnroundedMultiply
- UnroundedInsert
- UnMute (if MuteMode = STA)
- Mute-Undo trigger (if MuteMode = Cnt)
- Next (if SamplerStyle = STA)
- ReAlign
- MuteReAlign
- MIDIReAlign
- MIDIMuteReAlign
- QuantMIDIStartSong (only at next loop start)
- MuteQuantMIDIStartSong (only at next loop start)
- StartPoint
- MIDI StartPoint
- Undo Record (only at next loop start)

We send StopSong at:
- Reset

GeneralReset
Start Record
Mute (if MuteMode=STA)

If Sync = OutManualStartSong (OuS)
We send StartSong at:
QuantMIDIStartSong (only at next loop start)
MuteQuantMIDIStartSong (only at next loop start)
Mute-Mutliply ReAlign

We send StopSong at:
Reset
GeneralReset
Start Record

If Sync = IN
We send StartSong at:
Undo in Reset
QuantMIDIStartSong (only at next loop start)
MuteQuantMIDIStartSong (only at next loop start)

We send StopSong at:
never

Commanding a StartSong when Sync=In

StartSong can be sent in Reset with a press of Undo. This can be useful if you have another source of sync upstream of the Echoplex, but have stopped something downstream from it. Pressing Undo on the Echoplex is a convenient way to send a StartSong message to the downstream device and start it up. This also restarts the internal clock counters used to keep track of the “beat 1” of the external sequencer. So if you get off from the sequencer somehow, or if you want to have a different point in the sequence considered as Beat one, tapping Undo lets you redefine the downbeat.

QuantMIDIStartSong

There is a new MIDI command "Send MIDI StartSong at next loop start", or simply QuantMIDIStartSong. When this is executed, we send a MIDI StartSong message at the next StartPoint of the loop. The display shows "St.S" momentarily when QuantMIDIStartSong is executed. This command is very useful for restarting a sequencer or drum machine so it is in time with the loop. The function can be executed at any time, independent of the Sync parameter setting. QuantMIDIStartSong allows you to always have a way to send a StartSong message in sync with the loop StartPoint, even if the Echoplex is not the clock master.

Unfortunately we did not find a reasonable way to do a Quantized StartSong from the front panel without muting first. Therefore QuantMIDIStartSong is only available as a MIDI command. The MIDI location for QuantMIDIStartSong is source# + 40. See the MIDI command section for more information on how to use it.

MuteQuantMIDIStartSong

MuteQuantMIDIStartSong is a variation of QuantMIDIStartSong that first mutes the loop until StartSong is sent. This is useful for having the loop drop out and then everything start up together and in time. MuteQuantMIDIStartSong is also equivalent to using the Mute-Multiply combination from the front panel. The function can be executed at any time, independent of the Sync parameter setting. This allows you to always have a way to send a StartSong message in sync with the loop StartPoint, even if the Echoplex is not the clock master. The MIDI location for MuteQuantMIDIStartSong is source# + 41. See the MIDI command section for more information on how to use it.

SongPositionPointer and ContinueSong

SongPositionPointer and ContinueSong messages are received according to the MIDI spec when Sync = In. Since we cannot reproduce the entire sequence of how the loop was built throughout the song, we just put the actual loop in the correct timing with any position of the song, assuming that the loop length and the sequencer timing stayed the same.

The positioning happens through ReAlign after the sequencer has been continued. After the sequencer has been stopped, put the Echoplex into the ReAlign state. It will wait for a sync event. Press Continue on the sequencer. It will send a SongPositionPointer message to the Echoplex to indicate where in the sequence it is starting from. It will then send a ContinueSong message and begin sending MIDI clocks again. The Echoplex will use the SPP information to determine where the Global MIDI StartPoint is. When the next Global StartPoint occurs, the Echoplex will trigger the loop at the beginning so that it is back in sync with the sequencer.

QuantStartPoint

If Sync=IN and an external clock is present, a long press on StartPoint moves the internal loop StartPoint to the next Global MIDI StartPoint defined by the external clock. (the “Beat 1” of the sequencer). The internal sync counters are realigned to the sequencer’s beat 1.

The result is the same as if you had built the current loop from the beginning with all operations quantized to the external clock. This is another way to “ReAlign” the loop to MIDI, although you are really redefining the StartPoint according to the external sync instead of retriggering the loop to it. It is especially interesting if you start with a non-rhythmic loop, then bring the drums in later to define the rhythm and sync for further development of the loop.

QuantStartPoint function is also available as a DirectMIDI command.

AutoStartPoint

The Echoplex maintains sync to an external clock by retriggering the loop at the Global MIDI StartPoint defined by the clock. An old problem is that true sync cannot be maintained when either FeedBack is reduced or Overdub is on, because an early arriving sync would erase the change made in that pass of the loop or cause glitches to record in the loop. We have to “free run” during that time to prevent such problems. There is no fix for this really, as it is a fundamental issue of how sync works in the Echoplex. Once Overdub is turned off and/or FeedBack brought back up, true sync returns.

We developed a trick for the case where the user reduces FeedBack a lot (a total change of the song for example). He probably does not really care to have his old fading loop stay in sync with the clock anymore, but he will want to stay in sync with whatever new things he overdubs. So when FeedBack is reduced enough and the Echoplex detects that the sync has drifted considerably off, we do an automatic StartPoint function. This sets the internal StartPoint of our loop at the time of the Global MIDI StartPoint defined by the external clock, and that point is then used for sync. Then when you bring FeedBack up again the StartPoint of the Loop and the external sequencer will be very close together and syncing resumes easily.

8th/Cycle (8th/Beat on older units)

The 8th/Cycle parameter has new values, allowing you to have up to 256 8th notes in a Cycle. This gives a lot more flexibility for how the Echoplex synchronizes to external devices.

When editing , the most important values come first to make them easy to select: 8,4,2,6,12,16,32,64,128,256, then it goes on with 1,2,3...96. Note that with the new data wheel feature, the top of the knob range ends at 54 instead of 96. This was done because we found it was easier to set the more typical values when the knob resolution was limited a little bit. To reach the values between 54 and 96 you simply use the front panel button to continue incrementing the number in the usual way.

Another new feature: If a loop is playing, a change of the 8th/Cycle value is only activated at the first Loop End *after* you come out of the Parameter editing state. At that point you jump directly to the new selected value. This means the value change occurs only while back in the playing state, and only at a rhythmically sensible point. This helps eliminate any confusion when working with a synchronized sequencer and makes for much smoother transitions into new time signatures.

Try changing 8th/Cycle with sync = out and a sequencer slaving to the clock. You control the sequencer's tempo in relation to your loop!

Improved Sync Routings

Some improvements have been made in the routing of synchronization signals.

BeatSync and Sync=IN
If a BeatSync pulse is received during Reset while Sync=In, MIDIclock is sent out at the corresponding tempo but ignored if received at the MIDI In jack. The purpose of this is to receive a HW sync at the BeatSync of one unit and send the sync on to others as MIDIclock. This worked in LoopIII, but there was the possibility to get the units confused when both MIDI clocks and BeatSync were inputs to the first unit.

MIDI Clock in and BrotherSync
If a MIDI clock is received, BrotherSync is ignored

MIDIclock out and Sync = IN
MIDIclock is not sent anymore when Sync = In (as was true in LoopIII). This eliminates potential confusions with multiple clocks.

Clock Piping
Incoming MIDI clocks are always piped through to the output with a maximum delay of 2ms. This makes it easy to chain together several Echoplexes with the same source clock as sync, while still maintaining other interesting control options between them via MIDI. Clock is only piped when Sync = In or Off. Clock is not piped when Sync = Out or OuS.

Chapter 6: MIDI Control

DirectMIDI and VirtualButtons

There are two kinds of MIDI commands now, VirtualButtons and DirectMIDI:

VirtualButtons
The existing MIDI commands from LoopIII are now called VirtualButtons, because they do exactly what the corresponding front panel button does. You can press the button virtually by MIDI, and do things like long presses and short presses and cross functions. If the ControlSource parameter is set to Notes, then a NoteOn message is the same as pressing the button, and a NoteOff message is the same as releasing the button. If the ControlSource Parameter is set to Continuous Controllers, a controller with a positive value is the same as pressing a button, and a controller message with a value of 0 is the same as releasing the button. This press and release concept is important to remember when programming a MIDI controller. When using VirtualButtons you have to make sure you send both the NoteOn and the NoteOff, so that you both press *and* release the button!

The virtual buttons emulate the front panel interface in MIDI. However, there are now more VirtualButtons than front panel buttons because we offer all the InsertMode options simultaneously from MIDI as if they were separate buttons. The MIDIInsertButton is the exception, it still does what is selected in InsertMode for consistency. VirtualButtons allow you to use MIDI to take advantage of the economical design of the front panel interface, which lets you get a lot of functionality out of a few buttons.

DirectMIDI
There are a lot of new MIDI commands that are called DirectMIDI. These do new things or reach functions directly that would take several button presses from the front panel. There are some limits to this, in that ReAlign, Half-Speed, and Quantize functions can execute at any time, but the others can only be executed while the loop is playing, overdubbing, or substituting. In many cases DirectMIDI commands only require the NoteOn message, or the single continuous controller message. They do not need the NoteOff. This is not true for the SUS commands. These use the NoteOn message to start their function and the NoteOff message to end it.

Source# and LoopTrig Parameter Defaults Adjusted
To create space for the new commands, the default for the LoopTrig Parameter was moved to 84, while the Source# parameter for the commands stays at 36. If the parameters are changed such that the ranges for LoopTriggers and commands overlap, preference is given to the LoopTriggers. So if you lower the LoopTrig parameter, some of the new DirectMIDI commands may become inaccessible.

Expanded MIDI Commands List

MIDI buttons

Note	source# offset	function	short descriptions (see expanded description below)
G#	-4	8thSync out	a short note out at each 8th note (output only)
A	-3	LoopSync out	a short note out at each Loop StartPoint (output only)
A#	-2	MIDISync out	a short note out at each Global MIDI StartPoint (output only)
B	-1	BrotherSync out	a short note out at each Cycle StartPoint (output only)
C	0	ParameterButton	Virtually presses the Parameter Button
C#	1	empty	
D	2	RecordButton	Virtually presses the Record Button
D#	3	OverdubButton	Virtually presses the Overdub Button
E	4	MultiplyButton	Virtually presses the Multiply Button

F	5	InsertButton	Virtually presses the Insert Button, depends on InsertMode
F#	6	MuteButton	Virtually presses the Mute Button
G	7	UndoButton	Virtually presses the Undo Button
G#	8	NextButton	Virtually presses the Next Button
A	9	ReplaceButton	Virtually presses the “Replace Button”
A#	10	SubstituteButton	Virtually presses the “Substitute Button”
B	11	InsertOnlyButton	Virtually presses the Insert Button, regardless of InsertMode
C	12	SpeedButton	Virtually presses the “HalfSpeed Button”
C#	13	ReverseButton	Virtually presses the “Reverse Button”

New DirectMIDI commands

Note	**source# offset**	**function**	**short descriptions (see expanded description below)**
D	14	SUSRecord	Sustain Action Record
D#	15	SUSOverdub	Sustain Action Overdub
E	16	SUSRoundMultiply	Sustain Action Rounded Multiply
F	17	SUSRoundedInsert	Sustain Action Rounded Insert
F#	18	SUSMute	Sustain Action Mute
G	19	ShortUndo	Immediately execute the ShortUndo function (Undo to end)
G#	20	SUSNextLoop	Sustain Action NextLoop (NoteOn = Next, NoteOff = Previous)
A	21	SUSReplace	Sustain Action Replace
A#	22	SUSSubstitute	Sustain Action Substitute
B	23	SUSToggleReverse	Sustain Action Reverse
C	24	SUSToggleSpeed	Sustain Action HalfSpeed
C#	25	Reset	Immediately reset current loop
D	26	GeneralReset	Immediately reset all loops
D#	27	Exit Parameters	Exit Parameter editing and return to Play state
E	28	SUSUnroundedMultiply	Sustain Action Unrounded Multiply
F	29	SUSUnroundedInsert	Sustain Action Unrounded Insert
F#	30	SUSMute-Retrigger	Sustain Action Mute-Retrigger (NoteOn = Mute, NoteOff=Retrig.)
G	31	LongUndo	Immediately execute the LongUndo function
G#	32	Forward	Go into Forward
A	33	Reverse	Go into Reverse
A#	34	FullSpeed	Go into FullSpeed
B	35	HalfSpeed	Go into HalfSpeed
C	36	PlaySample	Immediately restart the loop and play once
C#	37	ReTrigger	Immediately restart the loop and play forever
D	38	ReAlign	Restart the loop at next Global MIDI StartPoint
D#	39	MuteReAlign	Immediately Mute and restart the loop at next Global MIDI StartPoint
E	40	QuantMIDIStartSong	Wait to next Local Loop StartPoint and then send a StartSong
F	41	MuteQuantMIDIStartSong	Immediately Mute, then wait to next StartPoint and send StartSong
F#	42	StartPoint	Set the StartPoint to the current spot in the loop
G	43	QuantStartPoint	Wait to next Global MIDI StartPoint and then set the local StartPoint there
G#	44	BeatTriggerSample	Mute and wait for BeatSync, then trigger loop to play once. Repeated BeatSyncs retrigger the loop.
A	45	MIDIBeatSync	Virtually send a BeatSync pulse by MIDI

SUS MIDI Commands

The concept of Sustain action, or “SUS” commands is greatly expanded through DirectMIDI. Every function has a SUS version separately available through MIDI, allowing them to be accessed at any time through a MIDI controller

using NoteOn and NoteOff messages. The NoteOn starts the function and the NoteOff ends it. This opens up a whole new range of expressive possibilities with the Echoplex that were never available before! The SUS commands can also be controlled with Continuous Controller messages, where a positive value turns the function on, and a 0 value turns it off. The SUS MIDI commands are detailed in the table above.

SUSNextLoop

SUSNextLoop is an interesting special case of the SUS MIDI commands. With SUSNextLoop, pressing it puts you into the Next Loop and releasing it returns you to the previous loop. In other words, NoteOn puts you into the NextLoop and the NoteOff brings you back. This allows you to bounce in and out of an alternate loop from you main loop. Combining SUSNextLoop with functions like AutoRecord and LoopCopy can give many interesting possibilities for creating alternate loops to bounce in and out of.

PreviousLoop

An interesting feature that falls out of the SUSNextLoop function is a PreviousLoop function. The NoteOff command for SUSNextLoop sends you back one loop. If you only use the NoteOff command for SUSNextLoop, it only sends you backwards through the loops, and becomes the PreviousLoop command! If you set one button on a MIDI controller to only send the NoteOn for SUSNextLoop, and another button to only send the NoteOff for SUSNextLoop, you have a convenient way to go forward and backwards through your loops.

MIDI pipe

All incoming MIDI commands received at the MIDI In port are monitored and selectively sent to the MIDI Out port, depending a bit on the state of the Echoplex. We call this “Piping”. MIDIpipe makes it possible to have multiple Echoplexes chained together and easily switch between using them independently or together as stereo pairs. It also makes it simple to work with multiple devices sharing the same MIDI clock. It is similar to a MIDI merge function, except with Piping the Echoplex is intelligently deciding which things should be piped to the output or not. A Merge function would send everything through, which can cause major problems in some situations. MIDIpipe intelligently prevents such problems.

MIDIpipe works with very low latency, so you will not notice any significant difference between a command that is piped and one that went direct.

MIDI Sysex commands are also piped immediately, which is convenient for controlling multiple Echoplexes from a PC using a sysex librarian utility. Unfortunately, while piping Sysex commands the audio is stopped due to the complexity of handling the sysex commands in real time. This is not audible for short commands like changing a single parameter, but you may hear it for long Sysex strings.

MIDI clock is piped when Sync = In or Off. It is not piped when Sync = Out or OuS, because the Echoplex will also be generating MIDI clock internally. If they both went out you would have double clocks, so MIDIpipe prevents that.

StartSong, StopSong, and Continue messages are piped in all cases, so an external controller at the beginning of the MIDI chain can send commands to a sequencer that is getting clock from the Echoplex. When the Echoplex is piping Start, Stop, and Continue messages it intelligently checks whether it has already sent the message itself within the last 10ms, and does not pipe if it has. This prevents multiple Echoplexes in parallel from stacking up the StartSong messages when they are all Recording simultaneously.

Echoplex functions that normally send out a MIDI command do not duplicate the command if it is being piped. For example, if a MIDI command is received for Record, the command is piped to the output and used internally to start the Record function. The Record function does not then send another MIDI command for Record as it normally would, so the commands do not get doubled by piping.

MIDI Data Wheel

The Feedback knob value is transmitted by the MIDI continuous controller DataWheel (Continuous Controller #6) when in Parameters and Tempo Select. Likewise, the Echoplex receives the DataWheel control for changing the parameter values from MIDI.

Sysex

All parameters can be edited by Sysex. This is a useful way to backup your settings and presets, send song parameters from a sequencer, or control single parameters from a capable MIDI controller. Since the sound is interrupted during this edit, you may hear a pause for a longer sequence of Sysex commands. For a single parameter change the dropout is not audible. All Sysex information is piped so you can set two Echoplexes to different Device IDs and set them up separately, even with MIDI connected simply from MID-In to MIDI-out. See the Sysex section for more details.

MIDI Sync Indicators

MIDI notes are transmitted out at various synchronization points related to the current loop. These are shown at the top of the MIDI command table above. MIDI notes are sent to indicate each Loop StartPoint, Global MIDI StartPoint, Cycle point, and 8^{th} note. The Global MIDI StartPoint note is only sent if it is different from the Local StartPoint. The 8^{th} note is sent at points determined by the 8^{th}/Cycle parameter and the loop length.

The purpose of these notes is to provide a convenient marker point when recording the Echoplex output into an audio sequencing or hard disk recording program. If you also record the midi output you can easily see the StartPoints, Cycle points, and 8^{th} notes of your loops in the track windows.

Another use of these notes is a metronome. You can use them to trigger sounds on a synthesizer or sampler to serve as an audible tempo indicator for your loops.

Chapter 7: Parameter Presets and Editing

Presets

Parameter settings can now be stored and recalled as Presets. There are 15 memory spaces to save sets of all local parameters and Tempo. So for example, if you like to switch Quantize on and off quickly, you could switch it on through the parameter matrix, then save to Preset 1, then switch it off and save to Preset 2. Then keep calling those two Presets with a MIDI Program Change from a MIDI controller while playing. Much easier than editing from the front panel every time!

Selecting Presets
The 15 Presets can be chosen by one of the following methods:

- From within the Preset Editor, accessed with the Reserved parameter
- with the Mute and Insert buttons while in Reset
- with MIDI Program Change commands at any time.
- Sysex

When InsertMode=HalfSpeed, that takes precedence over the Preset control on the Insert button. So only the Mute button is available for changing presets in that case.

Parameters saved in Presets

- Loop/Delay
- Quantize
- 8ths/Cycle
- Sync
- Threshold
- RecordMode
- OverdubMode
- RoundMode
- InsertMode
- MuteMode
- MoreLoops
- AutoRecord
- LoopCopy
- SwitchQuant
- Velocity
- SamplerStyle
- Tempo

Global Parameters not Changed by Presets
The MIDI parameters are set globally and independent of the Presets. So the following are Global:

- MIDI Channel
- ControlSource
- Source#
- VolumeCont
- FeedBkCont
- LoopTrig
- Device ID for Sysex and SampleDump
- Last Preset

MoreLoops is a special case
Normally, when the MoreLoops parameter is changed a General Reset is done automatically. This is necessary because the memory must be reorganized to support the new number of loops, but it unfortunately means current loops are lost. When you change MoreLoops from the front panel we assume you are doing it consciously and are aware that any loops you have going will get reset. However, we found it was very easy to change to a preset while you are playing without realizing that it had a different number of loops stored, and consequently destroy something you are playing.

We developed a simple system to protect against this situation. Whenever you change presets while in a loop with something recorded, any change in the MoreLoops parameter due to the Preset change is ignored. So your loops are not destroyed by the Preset change. If you are in a reset loop when you make the Preset change, we go ahead and implement the MoreLoops change as well and do the GeneralReset. If you changed the preset while playing and you really did want the MoreLoops to change and have everything reset, you will need to reset the current loop and then reload that preset.

Understanding the Playing State "Preset"
Preset 0 is the preset used and edited while playing. These are the Parameters active while you use the Echoplex. When one of the fifteen presets is loaded for use, what really happens is the values in that saved preset are copied into Preset 0 to be used while playing. When you save a preset using the preset editor, what really happens is the values in the Preset 0 playing state are copied into the preset you are saving to.

If you make a change in a parameter value while you are playing, it is only changed in Preset 0, and the Preset you loaded is not affected. This is really helpful because it means you can freely reach over and change something as you play, without affecting any Presets. When you decide you like a setup, you can then go into the Preset Editor and save it. Or you can load up another preset you have saved to get back to a known state. Changes made in Preset 0 are preserved when the power is turned off.

Preset 0 is really the same as the way Parameters worked in LoopIII. So if you do not use Presets at all, everything will be the same as it was before in LoopIII when you make Parameter changes. As you change Parameters, you will actually be editing the Preset 0 location, but you won't even realize it.

We call the playing state "Preset 0" now because that is the actual location in memory where it lives. This will matter to you if you use Sysex or a preset librarian program to edit the parameter values saved in your presets. If you want to edit the active playing state parameters with Sysex, you simply edit the Preset 0 location. See the Sysex chapter for more information on Sysex control.

Selecting Presets from the Front Panel
In Reset, the Mute and Insert buttons can be used to select and load Presets 1 - 15. Mute scrolls up and Insert scrolls down. The preset number is displayed on the LoopTime display as it is selected. Once you stop on one for longer than 400ms it will be loaded. The display briefly shows "LOA" when the preset loads. The Mute and Insert LEDs will be Orange to indicate they are available for preset selection.

There are two important things to note about the front panel preset selection feature. For one, if InsertMode = Half Speed you do not have the Insert button available to scroll down through the Presets. This is because we want to be able to start loops in half speed, so the Insert button is dedicated to selecting half speed while in reset. You will be able to tell this is the case because the Insert LED will be green or red (full speed and half speed, respectively). It is Orange if it is available for changing presets. You will also be able to tell if you press Insert anyway, because the display will show H.SP or F.SP instead of the preset number.

The second important thing to note is the front panel Preset selection is not available at all *until* you have saved your first preset! This was done to make the new user interface friendlier to new users. We didn't want somebody to accidentally press the button, select a preset, and lose any other parameter changes they had made before they understood how the preset feature worked. So until they've read this section and learned how the presets work, we save them from that mistake. In any case, there is no point in loading presets if you haven't saved any, so we don't clutter the interface with it.

Selecting Presets with MIDI Program Change
Selecting presets with MIDI is very straight forward. MIDI program change messages 1 – 15 select presets 1 – 15. Remember, MoreLoops value changes are not made unless we are already in Reset.

If you have edited parameters on the fly and want to return to what you had, MIDI program change 16 reloads the original saved preset.

Preset Editor

Accessing the Preset Editor
The Preset Editor is accessed by the "Reserved" parameter. This is in the "Switches" (or "Keys") row, under the NextLoop Button. When you press it you are in the Editor for the Presets. There, all the other buttons get a new function.

Preset Editor Commands

Mute	counts up Preset#
Insert	counts down Preset#
Multiply	Sets selected Preset and Preset 0 to factory default
Overdub	long press saves the playing state settings (Preset 0) to the preset displayed
Record	loads the preset displayed to the playing state setting (preset 0) for use.

Preset Editor Display
The Display shows the Preset number as "Pr #". No dot after "Pr" indicates that this Preset is the last one that has been loaded into the Playing state Preset 0. If there is a dot after "Pr", it is not the one loaded. When the display shows "PrE" it means that the Preset 0 parameter settings have been modified since the preset was loaded to them. With the display indicators it should be easy to tell in the editor which preset you are currently using, and if there are new changes made that you may want to save.

Time Required for Saving Presets
Please note that the saving of a Preset to the non-volatile memory in the Echoplex takes about 400ms. if the unit is power cycled before that, it may come back on with different parameter values than expected. Make certain you wait a second after saving for the Parameters to be fully saved before turning off the power.

Sysex Dump/Load

Parameter sets can be dumped and loaded from an external MIDI device for backups or librarian functions. See the Sysex section for more details.

Sysex Individual Parameter Editing

All parameters can be individually edited by Sysex. Sysex editing is useful for backing up your settings and presets, sending song parameters from a sequencer, or controlling single parameters from a capable MIDI controller. Since the sound is interrupted during this edit, you may hear a pause for a longer sequence of Sysex commands. For a single parameter change the dropout is not audible. All Sysex information is piped so you can set two Echoplexes to different Device ID's and set them up separately, even with a simple MIDI-In to MIDI-Out connection. Make certain you have set the Echoplex device ID to something. If the Device ID is left at 0, sysex will not be received. See the Sysex section for more details.

DataWheel

The parameters that have many values can be selected with the FeedBack knob on the front panel while that parameter is being viewed. The FeedBack knob becomes the DataWheel. Select the parameter as usual and then turn the Feedback

knob to edit the value instead of pushing the button repeatedly. Parameters with less than 127 values have them spread over the range of the knob. Note this only works from the actual knob on the front panel. A pedal connected to the feedback pedal jack does not have this function.

DataWheel works with:

- 8th/Cycle
- MIDI channel
- Source #
- LoopTrig #
- VolumeCont
- FeedBkCont
- MoreLoops

Since the DataWheel is the Feedback knob in the playing state, the Echoplex makes some effort to avoid conflicts between feedback settings and DataWheel settings. The value for feedback is stored and maintained when you enter parameters. Using the feedback knob as the DataWheel for editing a Parameter value does not change the actual feedback setting. When you leave Parameters, feedback is still set the same as you left it. However, you should still be careful because the knob will now be in a different position. The next time you change the feedback with the knob the value will jump to the knob position. Usually it is best to remember setting it back where you want before returning to play mode. The Echoplex helps you with this if you forget, by providing a short time gap after you start moving the knob before the knob is read. So if you turn the knob quickly you can return to the value you want without getting a strange feedback setting.

DataWheel Continuous Controller

The Feedback knob value is transmitted by the MIDI continuous controller DataWheel (Continuous Controller #6) when in Parameters. Likewise, the Echoplex receives the DataWheel control for changing the parameter values from MIDI.

Chapter 8: Sample Dump

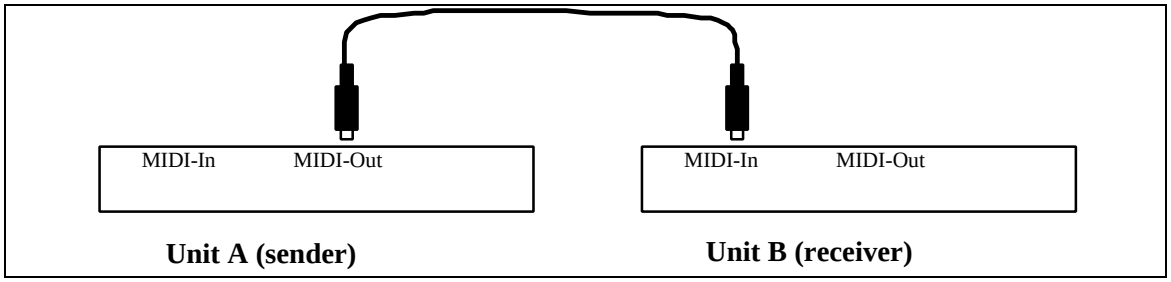
Introduction

The Echoplex uses MIDI sample dump to transfer the loop you recorded to another Echoplex, sampler, or even your computer. MIDI Sample Dump is an extension of the MIDI standard which has been used by many samplers in the past 20 years. It uses Sysex to transmit the audio data. Unfortunately, not all samplers implement sample dump exactly the same way. To give you the possibility of transferring samples between a wide range of samplers we have provided new ways of changing parameters of the MIDI sample dump in the Echoplex.

The following chapters give a short introduction into some details of the MIDI sample dump that may help you.

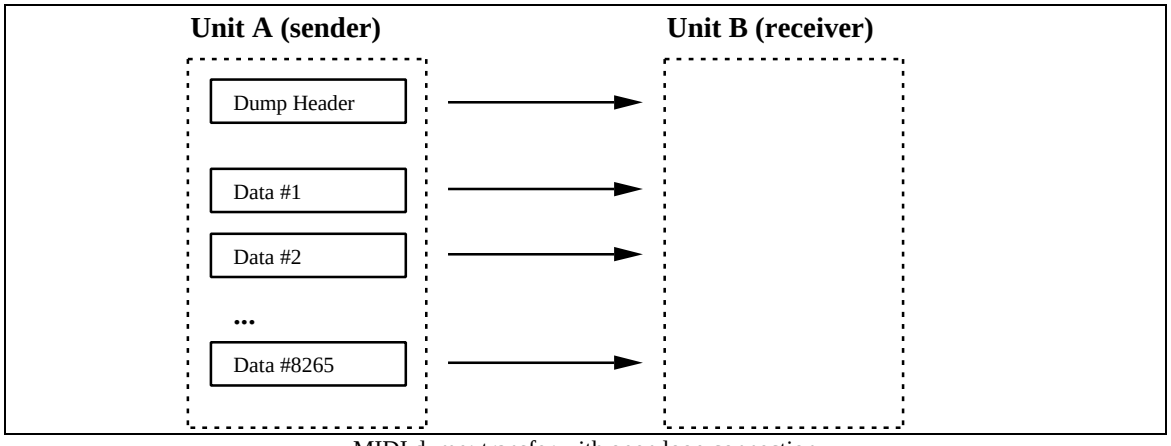
General Sample Dump

Connections
The minimal MIDI patch is to connect the MIDI-Out of the sending unit to the MIDI-In of the receiving unit.



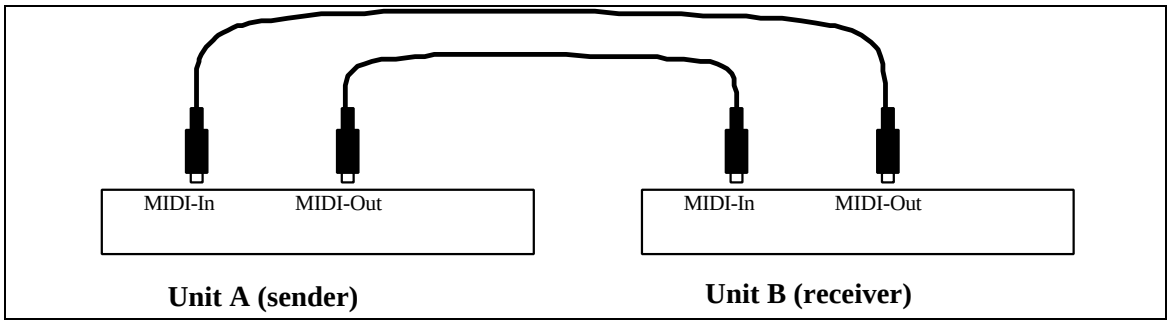
MIDI Dump: open loop connection

Unit A sends a header first, containing general information about the sample to be sent. Then it divides the sound data into packets and sends one packet at a time.



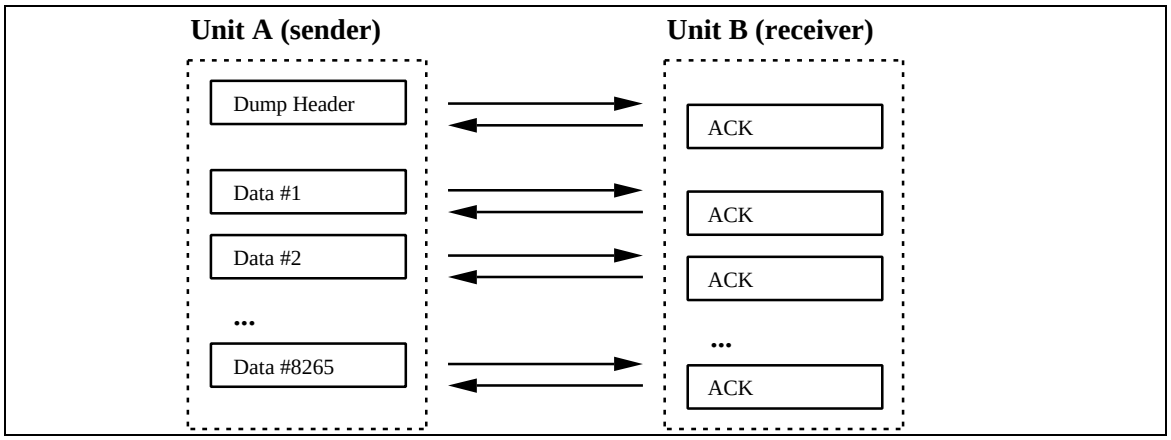
MIDI dump: transfer with open loop connection

If you have two MIDI cables you should connect the two units in both directions.



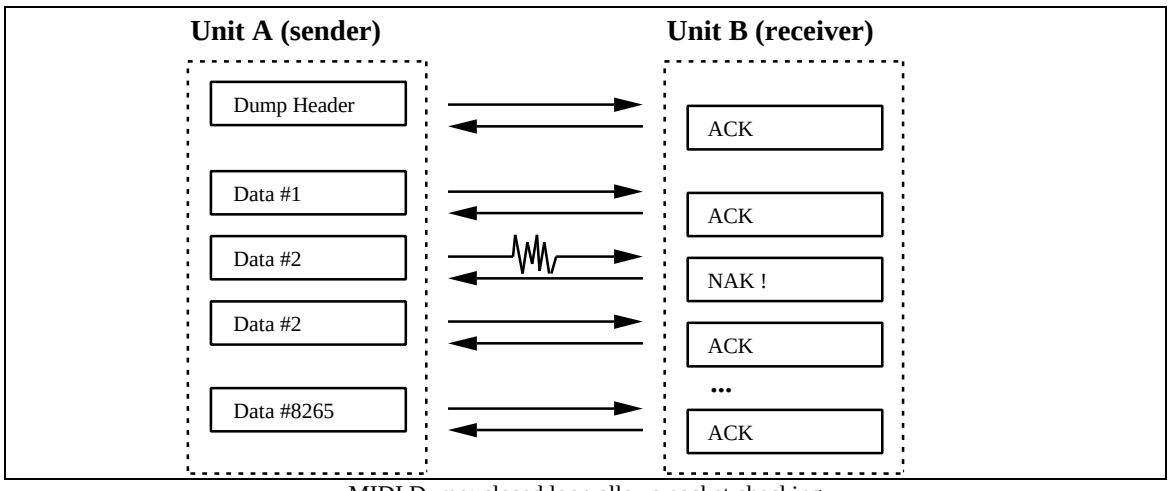
MIDI dump: closed loop connection

This allows unit B to acknowledge each package as it arrives. This is faster and safer than the open loop method.



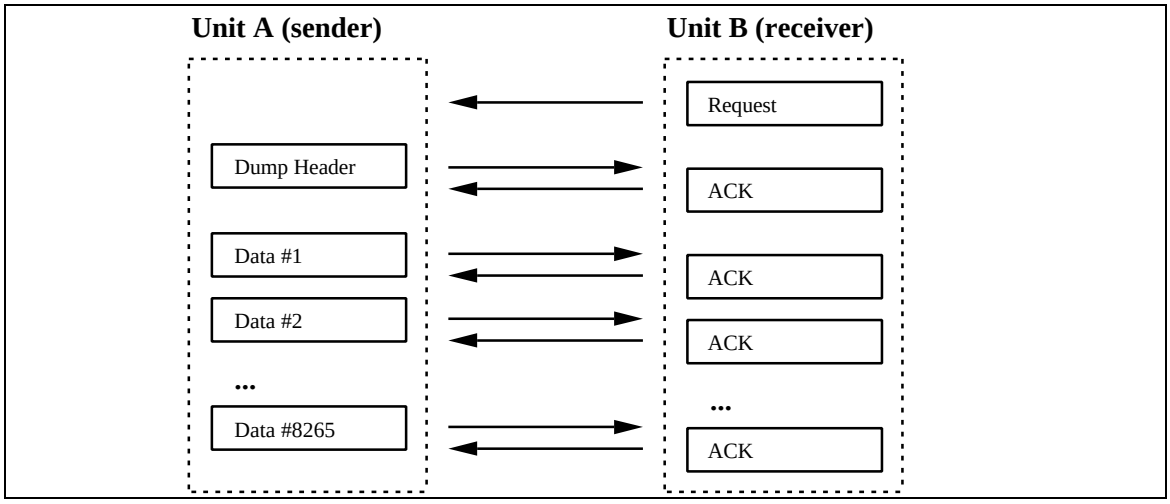
MIDI Dump: transfer with closed loop

This gives the receiving unit B an opportunity to check each incoming packet and ask for a retransmit if necessary.



MIDI Dump: closed loop allows packet checking

Who starts sending?
The previous section showed the case where the sample dump was initiated from the sending unit A. This is not always possible or easy. Some samplers don't even have a sample dump button or menu command. They completely rely on the other unit to initiate the transfer. This means the receiving unit has to send a sample dump request first. The sending unit then reacts by sending the desired dump.



MIDI Dump: request initiates dump

Of course, this works only with a closed loop connection.

Sample Number
The Sample Dump header and request contain several pieces of information about the sample. The most important one for the user is the sample number.

A sampler can store many samples. These samples are usually indexed. In a MIDI sample dump the sender tells the receiving unit what sample number it is sending. Correspondingly the receiving unit can request a specific sample number. The sample number can range from 0 to 16383. Usually not all numbers are accessible to the user.

Unfortunately some samplers start numbering the samples with 0 and others with 1, while still others show you a 1 but actually send them as sample number 0! You have to find out what numbering system your sampler uses. If you can't find it in your manual you have to try it out. Send a sample as #10 and see whether it arrives as #9, #10 or #11.

See below for information on how sample numbers are used in the Echoplex.

Device ID
The device ID of Sysex messages (sample dumps are Sysex messages) are used to make sure that the message is received only by the device that you wanted to send it to.

Two units have to have the same ID in order to talk with each other.

The name "Device ID" is somewhat misleading. "Sysex channel" would be more appropriate. Think of it as the equivalent of the MIDI channel. But be careful, MIDI channel and Device ID don't have to be the same value!

It is possible to have the MIDI notes go through channel 1 to one unit and the MIDI sysex through device ID 114 to another unit.

Device IDs can be in the range from 0 to 127.

A device ID of 127 has a special meaning. If you set the receiving unit to 127, all device IDs are accepted. If you set the sending unit to 127, all receiving units should therefore accept your SYSEX data. This is called broadcasting. See below for information on how device IDs are used in the Echoplex.

Echoplex Sample Dump

Connections
The Echoplex allows you to transfer the sound data through open loop connections (one cable) and closed loop connections (two cables). If you have a choice always use closed loops. The transmission will be approximately twice as fast.

Who starts sending?
The Echoplex can either initiate a sample dump itself (with the Dump button in parameter mode P3) or respond to a sample dump request.

Responding to a request is only possible while in Upload-mode. Enter this state by pressing the Load button in parameter mode P3. During normal operation all Sample Dump messages are ignored.

Loop Numbers and Sample Numbers
Many samplers have sample numbers up to 999. Since the Echoplex has a maximum of 16 loops we have a problem in how to match an incoming sample (e.g. #254) with a loop number.

The Echoplex gives you the possibility to set a current loop number and a "current" sample number as special Sample Dump parameters. While in Upload-mode press Record to edit the loop number and Overdub to edit the sample number.

When you send the current loop it is always sent with the sample number you defined on the Echoplex.

Receiving a sample number in a header or a request is somewhat more complex. To give a maximum of flexibility the following scheme is applied to incoming sample numbers:

Sample Number	Accessed Loop (sent or received)
0	access current loop number.
1-16	access corresponding loop number 1-16. (if you have that many loops)
17-16383	access current loop number. (>999 are clipped to 999)

The current sample number is then set to this value. Sample numbers on the Echoplex range from 0 to 999.

Device ID
The device ID of the Echoplex can be changed while in Upload-mode. Press Multiply to see the device ID. Press it again to start incrementing the value. The longer you press, the faster the value will be incremented.
Broadcasting (device id=127) is implemented.

Sample Dump User Manual

Sending a Dump(Dump-button)

The Dump button in the MIDI (P3) parameters of the Echoplex is used to send a dump to another device.

- You see a 'd' blinking, indicating that data is dumped.
- Pressing the Parameter button while dumping aborts the transmission.
- The device ID is 1.
- The sample number is the same as the loop number.

Sending & Receiving (Upload-button)

Press Load to enter the Upload-Mode.

- You see a '-' blinking, indicating that the Echoplex waits for specific commands, which can come from the buttons or via MIDI.
- The Upload-mode allows much more than just uploading. You can also send a dump from it or change sample dump parameters.

Button commands in Upload-mode

Parameter
Exits the Upload-mode. You will see the P3 display again. If you received a loop it may be stored in a loop other than the one that you are hearing now. Leave the parameter mode and select the loop with the Next button.

Record
Changes current loop number. The default is the loop you where listening to. Maximaum range from 0 to 16 (depends on the number of loops). Ln is shown in the green display.

Multiply
Changes the device ID. The default is 1. Range from 0 to 127. 127 is used to broadcast to all receivers/accept all senders. Id is shown in the green display.

Overdub
Changes current sample number. The default is the current loop number. Range from 0 to 999. Sn is shown in the green display

Insert
Reserved.

Mute
Reserved.

Undo (=Dump)
Send the current loop. The receiving unit will store it as the current sample number.

Next (=Upload)
Request a dump from the other unit. The current sample number will be sent by the other unit and stored in the current loop. This works only in a closed loop (2 MIDI cables).

The three value buttons (Record, Overdub, Multiply) work the same way. The first press shows the current value. Subsequent presses increment the value. If you press the button for a long time it starts repeating. The repeat rate speeds up the longer you press. The maximum speed is reached after approximately 100 increments. When the maximum value is reached, the value is set to minimum (usually 0 or 1) and the repeat rate slows down to give you the possibility to release the button.

Commands received via MIDI
While in Upload-mode the following MIDI Sysex commands are accepted:

SampleHeader(samplenumber)
Samples with the Sample Numbers 1-16 are stored in the corresponding loop. All other Sample Numbers are ignored and the sample is stored in the current loop number.

SampleRequest(samplenumber)
If Sample Numbers 1-16 are requested the corresponding loop is sent. All other Sample Numbers are ignored and the current loop is sent as the requested Sample Number.

Examples

The Echoplex is capable of starting the transmission and of passively waiting until the other unit takes control. The following examples give a few examples for transmitting loops from one Echoplex to another one.

Echoplex A -> Echoplex B

(open loop)

- •AB Connect MIDI-Out of Echoplex A with MIDI-In of Echoplex B.
- •AB Press Parameters on both units, until they are both in the MIDI parameter mode.
  (A shows 'P3')
  (B shows 'P3')
- •B Press Load on Echoplex B.
  (goes into Upload-Mode showing '-')
- •A Press Dump on Echoplex A.
  (A starts dumping showing 'd')
  (B starts uploading showing 'U')
- • Wait until the entire loop is sent
  (A shows 'P3' again)
  (B shows '-' again)
- •B Press Parameters on B to exit Upload-Mode
  (B shows 'P3' again)
- • Echoplex B has now uploaded the loop.

Echoplex A => Echoplex B

(closed loop, A initiates)

- •AB Set MIDIControlSource to “Off” to avoid MIDI loops
  Connect MIDI-Out of Echoplex A with MIDI-In of Echoplex B.
  Connect MIDI-Out of Echoplex B with MIDI-In of Echoplex A.
- •AB Press Parameters on both units, until they are both in the MIDI parameter mode.
  (A shows 'P3')
  (B shows 'P3')
- •B Press Load on Echoplex B.
  (goes into Upload-Mode showing '-')
- •A Press Dump on Echoplex A.
  (A starts dumping showing 'd')
  (B starts uploading showing 'U')
- • Wait until the entire loop is sent
  (A shows 'P3' again)
  (B shows '-' again)
- •B Press Parameters on B to exit Upload-Mode
  (B shows 'P3' again)
- • Echoplex B has now uploaded the loop.

Echoplex A => Echoplex B

(closed loop, B initiates)

- •B Press Load on Echoplex B.
  (goes into Upload-Mode showing '-')
- •A Press Load on Echoplex A.
  (goes into Upload-Mode showing '-')
- •B Press Load again on Echoplex B.
  (B sends a SampleRequest to A)
  (A starts dumping showing 'd')
  (B starts uploading showing 'U')
- • Wait until the entire loop is sent
  (A shows '-' again)
  (B shows '-' again)
- •A Press Parameters on A to exit Upload-Mode
  (A shows 'P3' again)
- •B Press Parameters on B to exit Upload-Mode

(B shows 'P3' again)

- Echoplex B has now uploaded the loop.

Not all samplers or computer programs are capable of all of these possible protocols. Some implement only the open loop protocol, some require a closed loop and don't operate on an open loop. It is even possible that your sampler has no Dump button, but can still handle MIDI Sample Dump when the Echoplex is sending the appropriate commands via MIDI.

How long will it take?

Your SampleDump will probably take a long time. MIDI is slow and samples used in looping tend to be big. The exact time depends on how long the sample is and whether the receiving unit acknowledges the data packages or not (the Echoplex does).

Here are a few estimates for the open loop (no acknowledges):

Sample Length	Transmission Time
0.1 sec	9 sec
1.0 sec	1 min
10.0 sec	10 min
100.0 sec	1 hour 50 min

And here the closed loop (with acknowledges):

Sample Length	Transmission Time
0.1 sec	5 sec
1.0 sec	50 sec
10.0 sec	8 min
100.0 sec	1 hour 50 min

The Other End

Echoplex

Sending MIDI SampleDumps from one Echoplex to another one is simple.

If you are connecting two Echoplexes make sure they don't use the same MIDI channel and/or the MIDI parameter "ControlSource" is switched off on the sending machines (both machines in a closed loop). This is especially necessary now with the MIDI pipe feature when using closed loop, since any MIDI command received is immediately sent out again. If you forget this, the two machines will send every button you press back and forth to each other in an infinite loop. Unhook them and do one of the following:

- Choose another MIDI channel on one of them.
- Switch the MIDI parameter "ControlSource" on the sending machine to off. This is necessary for closed loop, and both must have ControlSource off.
- Select another MIDI parameter "Source #" for one of them, so they send on another octave.

SoundDesigner ®

SoundDesigner subtracts 2 from the device ID number.

Echoplex -> SoundDesigner
Sending samples from Echoplex to SoundDesigner.

SoundDesigner -> Echoplex
SoundDesigner and the Echoplex do not agree on how to calculate the checksum. The Echoplex sends a NAK (=Not Acknowledge) after every data-package. Luckily SoundDesigner ignores all handshake messages and the Echoplex stores it anyway so you will end up with the correct sample in your Echoplex.

Alchemy ™

Alchemy™ needs to initiate sample dumps in both directions.

Alchemy™ actually sends the sample number off by one. When you ask for sample #5 it actually requests #4 from the Echoplex.

Echoplex -> Alchemy™
Go into Upload-mode on the Echoplex.
Send a request from Alchemy (menu: Network: Get Sound).

Alchemy™ -> Echoplex
Go into Upload-mode on the Echoplex.
Start the dump from Alchemy™ (menu: Network: Put Sound)
Make sure the sample length actually has the number of samples that you want to transmit. Sometimes Alchemy stores short sounds in huge files filling the end with zeroes.

K2000 ™

The K2000™ reserves the samples 1-199 for ROM samples which can not be dumped or overwritten.

It automatically adds 200 to incoming samples with sample numbers <200.

Sample number 0 writes an incoming sample to the first free place.

The K2000™ always adds 1 to the sample number.

You have to set the device ID (called Sysex ID) to the same as on the Echoplex. Setting it to 127 doesn't do the standard behavior (accepting all IDs), so make sure they are equal on both units.

Echoplex -> K2000™
Start sending from the Echoplex at any time. Make sure the sample you write to is free.
You need to go into the Edit Keymap to see your new sample. It is probably best if you add a new Keymap with your new sample.

K2000 ™-> Echoplex
Go into Upload-mode on the Echoplex.
Select the right sample number and send a Request (Undo-button).
You can also start the dump from the K2000™. The function is well hidden in the sample editor. Press <Dump>. The Echoplex will ignore the sample number you send and store it in the loop number you defined in the Echoplex Upload-mode (Record-button).

E-MU e64™

The device ID of the E-MU™ is set to 127 and cannot be changed. It therefore will always accept all samples, independent of your Device ID.

Echoplex -> E-MU e64™
The E-MU™ accepts dumps at any time. Just start sending from the Echoplex.

E-MU e64™ -> Echoplex
Send a Request from the Echoplex at any time. (faster)
or
Start the dump from the E-MU. This is very slow!

Troubleshooting

MIDI Sample Dump is a standard. But not all samplers implement this standard the same way. This can lead to problems.

Display

Display for received messages

- **H** header received
- **I** header received but ignored (e.g. too long)
- **L** data received (load data packet)
- **?** wrong packet received (e.g. Sysex for other machines)
- **A** ACK received
- **N** NAK received
- **W** WAIT received
- **C** CANCEL received
- **J** junk received (e.g. notes, ignored)
- **-** nothing received

Display for sent messages

- **d** data sent (dump data packet)
- **r** dump request sent
- **c** Cancel sent
- **H** Header sent

Error values
In case of an error (display 'E') the red display shows an error number.

- **1** overrun (data was sent too fast)
- **2** buffer overflow (too much data too fast)
- **3** timeout (aborted transmission in the midst of a packet)
- **4** received value out of range
- **5** unexpected value (expected a specific value but received something else)
- **6** checksum error

Chapter 9: SYSEX Control Detailed Documentation

```
PROTOCOL FOR ECHOPLEX PARAMETER DUMP
====================================

1) GENERIC ECHOPLEX SYSEX FORMAT
--------------------------------

SYSEX AURISIS_ID ECHOPLEX_ID device_id version command {data} [checksum]
SYSEXEND

SYSEX       = 240  = $F0
AURISIS_ID  = 00 01 48   = $00 $01 $30
ECHOPLEX_ID = 11   = $0B
device_id   = value [0..127] set in the MIDI Upload mode. 0 is off (no
              communication)
version     = format of this message (for backward compatibility)
command     = see below
{data}      = depends on version and command
SYSEXEND    = 247  = $F7


2) VERSION
----------

All Echoplex units are backwards compatible.
The version numbers start with 1 (=LD4.0 revision 95/96)

If data is requested in a known version, the answer will be in this format.

Version 0 (=zero) is interpreted as the current version of this unit.

Unknown version numbers are ignored (error message ???)

The format of the data bytes of the following commands depends on this
version number.


3) COMMANDS
-----------

3.1) COMMANDS VERSION 1
-----------------------

Overview:

F0  00 01 30 0B dev vers cmd {data} F7

INFO_REQUEST
F0  00 01 30 0B dev vers 0  F7
240 00 01 48 11 dev vers 0 247

INFO_DATA
F0  00 01 30 0B dev vers 1 vers (mem_1 mem_2 mem_3)  F7
240 00 01 48 11 dev vers 1 vers (mem_1 mem_2 mem_3) 247
```

```
GLOBAL_PARAM_REQUEST
F0  00 01 30 0B dev vers 10 from length pset  F7
240 00 01 48 11 dev vers 16 from length pset 247

GLOBAL_PARAM_DATA
F0  00 01 30 0B dev vers 11 from length pset val_1 .. val_n  F7
240 00 01 48 11 dev vers 17 from length pset val_1 .. val_n 247

LOCAL_PARAM_REQUEST
F0  00 01 30 0B dev vers 12 from length pset  F7
240 00 01 48 11 dev vers 18 from length pset 247

LOCAL_PARAM_DATA
F0  00 01 30 0B dev vers 13 from length pset val_1 .. val_n  F7
240 00 01 48 11 dev vers 19 from length pset val_1 .. val_n 247

ALL_PARAM_REQUEST
F0  00 01 30 0B dev vers 14  F7
240 00 01 48 11 dev vers 20 247

GLOBAL_PARAM_RESET
F0  00 01 30 0B dev vers 20 pset  F7
240 00 01 48 11 dev vers 32 pset 247

LOCAL_PARAM_RESET
F0  00 01 30 0B dev vers 21 pset  F7
240 00 01 48 11 dev vers 33 pset 247



3.1.1) COMMANDS: INFORMATION GROUP
----------------------------------

3.1.1.1) COMMAND: INFO REQUEST   = $0   = 0
-----------------------------

The info request contains no data bytes.
The header contains already all necessary information.

Returns an INFO command.

F0  00 01 30  0B dev vers 0  F7
240 00 01 48 11 dev vers 0 247


3.1.1.1.1) COMMAND: INFO   = $1  = 1
----------------------
Essentially ignored.

F0  00 01 30 0B dev vers 10 vers (mem_1 mem_2 mem_3)  F7
240 00 01 48 11 dev vers 16 vers (mem_1 mem_2 mem_3) 247

_____________________________________________________
|byte#  | bits | description
|_______|______|_____________________________________
|0      |      | vers  | version number of this unit
|1-3    | 21   | mem   | soundmemory size
|_______|______|_____________________________________

The version number of the sending unit may be used in further
communications.
```

3.1.2) COMMANDS: PARAMETER GROUP

The global parameters are accessed by indexes.

The indexes in version 1 are:

```
 0  VGPrmPrevParamSet
 1  VGPrmParamSet
 2  VGPrmMIDIChannel
 3  VGPrmMIDIReceiveCommand
 4  VGPrmMIDIFirstKey
 5  VGPrmMIDIFBCtrlr
 6  VGPrmMIDIVolCtrlr
 7  VGPrmMIDIFirstLoop
 8  VGPrmMIDIDevID
 9  VGPrmMIDISampleNumHi
10  VGPrmMIDISampleNumLo
```

The local parameters are accessed by indexes.

The indexes in version 1 are:

```
 1 Loop/Delay          (4 bits)
 2 Timing Quantize     (2 bits)
 3       8th/Cycle             (8 bit)
 4       SyncMode              (3 bit)
 5       TrigThreshold         (4 bits)
 6       RecordMode            (1 bit)
 7       OverdubMode           (1 bit)
 8       RoundMode             (1 bit)
 9       InsertMode            (4 bits)
10       MuteMode              (1 bit)
11       Overflow              (1 bit)
12 MoreLoops           (4 bits)  (real NLoops-1)
13 AutoRecord          (1 bit)
14 Next LoopCopy       (2 bits)
15 SwitchQuant         (3 bits)
16 Velocity            (1 bit)
17 SamplerStyle        (2 bits)
18 Tempo               (7 bits)
```

3.1.2.1) COMMAND: GLOBAL_PARAM_REQUEST = $10 = 16

Reads a range of global param values and returns them
as a GLOBAL_PARAM_DATA command.

```
F0  00 01 30 0B dev vers 10 from length pset  F7
240 00 01 48 11 dev vers 16 from length pset 247
```

byte#	bits	description
0	7	from	param index range start
1	7	length	param index range end
2	7	pset	param set (for future use, so far 127)

3.1.2.2) COMMAND: GLOBAL_PARAM_DATA = $11 = 17

Sets a range of global param values to new values.

```
F0  00 01 30 0B dev vers 11 from length pset val_1 .. val_n  F7
240 00 01 48 11 dev vers 17 from length pset val_1 .. val_n 247
```

byte#	bits	description
0	7	from	param index range start
1	7	length	param index range end
2	7	pset	param set (for future use, so far 127)
3..n	7	value	param value(s)

If there are less values than defined by 'from' and 'length',
then these parameters will be reset to their default value.

3.1.2.3) COMMAND: LOCAL_PARAM_REQUEST = $12 = 18

Reads a range of local param values and returns them
as a LOCAL_PARAM_DATA command.

```
F0  00 01 30 0B dev vers 12 from length pset F7
240 00 01 48 11 dev vers 18 from length pset 247
```

byte#	bits	description
0	7	from	param index range start
1	7	length	param index range end
2	7	pset	param set number. 127 means the current preset selected in the Echoplex

3.1.2.4) COMMAND: LOCAL_PARAM_DATA = $13 = 19

Sets a range of local param values to new values.

```
F0  00 01 30 0B dev vers 13 from length pset val_1 .. val_n  F7
240 00 01 48 11 dev vers 19 from length pset val_1 .. val_n 247
```

byte#	bits	description
0	7	from	param index range start
1	7	length	param index range end
2	7	pset	param set number. 127 means the pset actually selected in the Echoplex
3..n	7	value	param value(s)

If there are less values than defined by 'from' and 'length',
then these parameters will be ignored.

3.1.2.5) COMMAND: ALL_PARAM_REQUEST = $14 = 20

Reads a range of local param values and returns them
as a sequence of one GLOBAL_PARAM_DATA command and

as many LOCAL_PARAM_DATA commands as there are ParamSets.

```
F0  00 01 30 0B dev vers 14  F7
240 00 01 48 11 dev vers 20 247
```

3.1.3) COMMANDS: RESET GROUP (not implemented in version 1)

Resets the global or the current local set.

3.1.3.1) COMMAND: GLOBAL_PARAM_RESET = $20 = 32

Resets the global parameters to the factory defaults.

```
F0  00 01 30 0B dev vers 20 pset F7
240 00 01 48 11 dev vers 32 pset 247
```

3.1.3.2) COMMAND: LOCAL_PARAM_RESET = $21 = 33

Resets the current local parameter set to the factory defaults.

```
F0  00 01 30 0B dev vers 21 pset F7
240 00 01 48 11 dev vers 33 pset 247
```

4) CHECKSUM CALCULATION

THE CHECKSUM IS REMOVED!
IT MAKES IT TOO HARD TO USE MIDI SOFTWARE LIKE 'MAX'.
MAYBE AN OPTIONAL SYSTEM IN THE FUTURE.

The checksum calculation is the same as in the MIDI Sample Dump.
All bytes AFTER the SYSEX (=$F0) are XORed. This checksum is then transmitted as the
last data byte before SYSEXEND (=$F7).

5) COMMUNICATION EXAMPLES

```
UNIT A (version 1)                              UNIT B (version 2)

InfoRequest( version 1)          -------->
                                 <--------      Info( version 2) // in format
                                                vers 1

LocalData( version 1, 4, 4)      -------->      // stores l_param[4]

LocalData( version 1, 4, 127)    -------->      // stores l_params[4..last]

LocalRequest( version 1, 0, 127) -------->
/* stores l_params[4..last] */   <--------      LocalParam( version 1, 0, last)

LocalRequest( version 0, 0, 127) -------->
/* error: unknown version */     <--------      LocalData( version 2, 0, last)
```

```
                                  <--------     LocalRequest( version 1, 0, 10)
LocalParam( version 1, 0, 10)     -------->     // stores l_params[0..10]

                                  <--------     LocalRequest( version 2, 0, 10)
LocalData( version 1, 0, 10) /*or error ???*/    -------->     // stores
params[0..10]


Similar behavior for global parameters.

Please note the special global variable (VGParamSet at index 0)
which switches between the local sets.

Note: This example fully uses the info commands to find out the
version of the different units.
In most cases this will not be necessary and the two units will
simply assume to use the same protocol version.
```