

# Arpeg 01

## *MIDI Arpeggiator*

### Introduction

### Overview

Basics

MIDI Message Handlir

Timing and Synchron

### Functions

#### Big Buttons Group

Capture Mode

Scan Mode

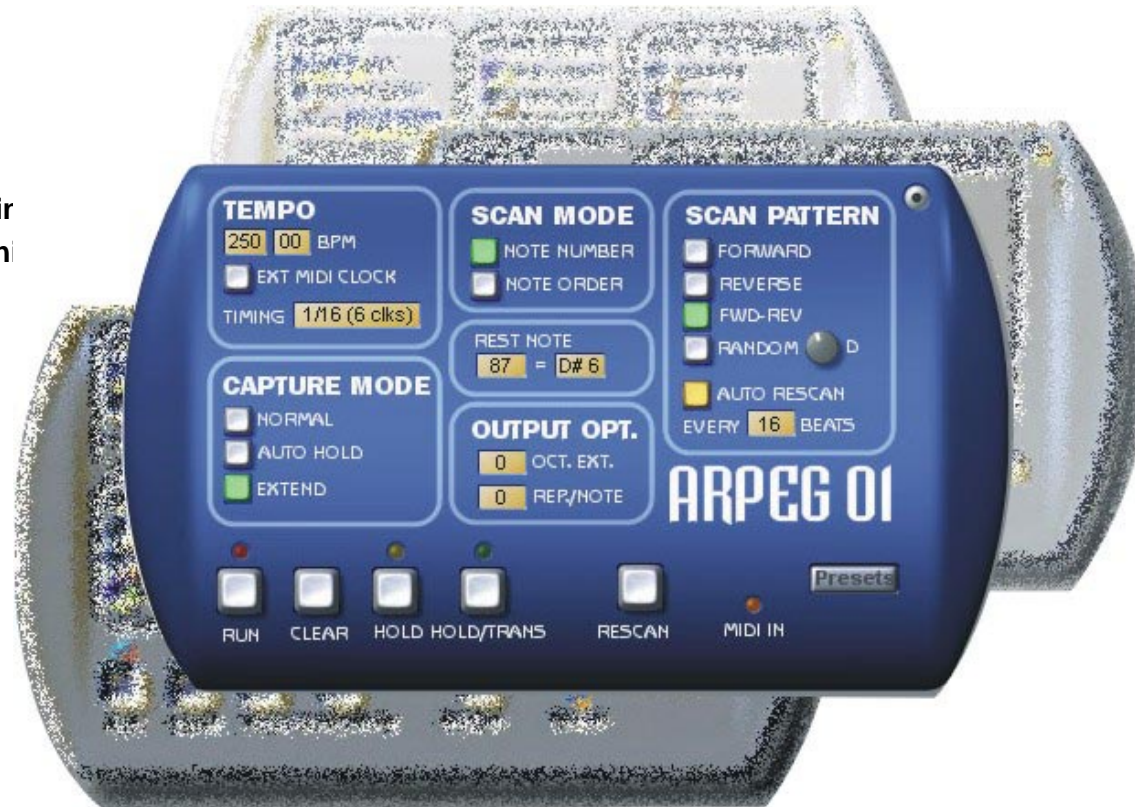
Rest Note

Scan Pattern

Output

Tempo/Timing

### Known Issues



# Introduction

## Welcome to the CreamWare Arpeg 01!

You now have a versatile but very intuitive and approachable arpeggiator at your disposal, which can be used not only with Pulsar synthesizers and samplers, but also with any external MIDI device.

The Arpeg 01 presents you with tremendous possibilities:

- Use it to effortlessly (and interactively) generate melodic sequences in real time – with easy-to-use control options, all of which can be adjusted at any time to yield practically endless variations – for direct use, to stimulate your own musical imagination, or to advance the frontiers of musical-acoustic research!
- Even for accomplished keyboardists, for whom keyboard technique is not an issue, the Arpeg 01 is a *fabulous* labor-saving device!
- The Arpeg 01 is a highly dependable tool for bringing *any* relationship (musical *or* non-musical) to a speedy end. It's therefore also a sure-fire vehicle for

getting your solo career (musical *or* non-musical) off to an early start!

But seriously ...

**No idea what an arpeggiator is?** In simplest terms, it's a device which *captures* chords (or short sequences of notes) that you play into it, and then *scans* the captured notes, periodically sending them out one at a time, thus generating *arpeggios* from your input.

The results of this simple process can be annoying and banal, or truly esoteric. Quite a lot depends upon the sounds you apply it to and the contexts you use it in. We won't presume to prescribe these things for you. All options are open. The Arpeg 01 puts enough capability within your immediate grasp to permit you to go off spontaneously in any direction you choose. (And if you nevertheless find yourself wishing for more, you'll surely want to check out its big brother, the Arpeg 02, which includes enough features and flexibility to enable you reach *any* extreme!)

Most features of the Arpeg 01 are easier to understand by using them than by reading about them. Even if you're just getting acquainted with the Arpeg 01, but especially if you already have a basic idea of what an arpeggiator is, there's nothing at all wrong with going straight into hands-on experimentation – you can refer back to the manual to go more in-depth or for clarification of details as needed. But *do* read the manual sometime – there are features here you wouldn't want to miss out on!

# Overview

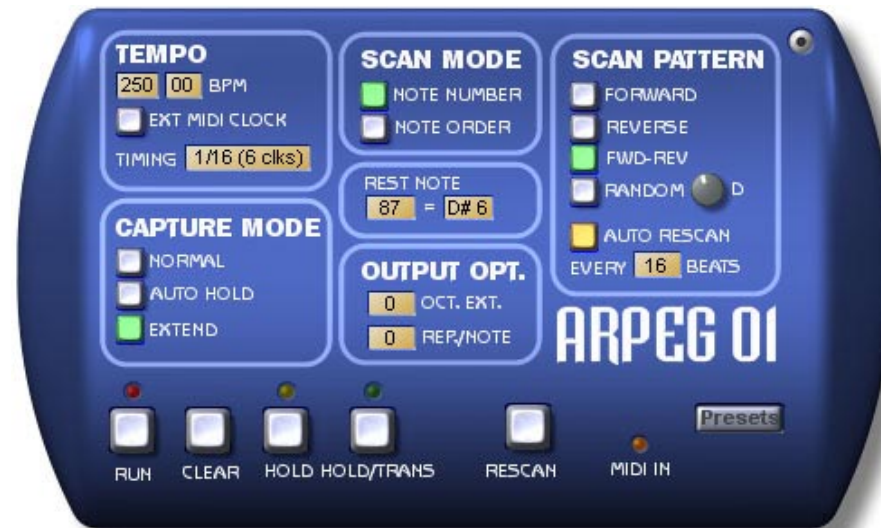
## Basics

This is a MIDI arpeggiator device:

- It is driven by MIDI note events on its **MIDI** input, which it *captures* in its internal chord buffer.
- It continually *scans* this chord buffer according to the current settings and produces MIDI note events as output.

(Note: in this text, MIDI note event and other input is referred to as coming from a MIDI keyboard, although the arpeggiator can of course be driven by *any* live or sequenced source of note events.)

The arpeggiator is monophonic. It puts out one note at a time, no overlapping notes and no chords.



## MIDI Message Handling

The handling of received MIDI messages depends on their type. In general:

- Received **note-on messages** are captured in the chord buffer (up to sixteen at any one time).
- Received **note-off messages** may or may not cause the matching note-on messages to be removed from the chord buffer, depending upon the current settings.

- *Received* note messages are *not* echoed to the output while the arpeggiator is running. The note events which the arpeggiator sends out are primarily those which the arpeggiator itself *generates* via scanning of the captured chord.

- All other received *channel messages* (e.g., modwheel, pitch bend, etc.) are echoed directly to the output at all times and have no particular pre-defined effect upon the arpeggiator. (Of course, as with all Pulsar devices, you can assign MIDI controllers to various controls on the arpeggiator.)

- *Non-channel messages* – i.e., **System Exclusive, System Common and System Realtime** (MIDI Clock) messages – are always ignored by the arpeggiator at its **MIDI** input and are not echoed to the output under any circumstances.

- However, MIDI Clock messages can be sent in on the **MCik** input – these are likewise not echoed.

There are some exceptions to the above rules regarding echoing of channel messages. These are discussed at the appropriate points in the manual (see **Big Buttons Group – HOLD**).

The channel number of incoming MIDI messages is ignored. There is no MIDI channel control. Output events generated by the arpeggiator have the same channel number as the input events from

which they are generated. It is assumed that the arpeggiator will be driven by MIDI events on a single channel. However, this isn't a critical point – if the input includes note events from more than one channel, the output will correspondingly include events on all of these channels, mixed up in possibly complex and interesting ways – which can be considered a feature for "special effects" purposes. Go ahead and try it!

## Timing and Synchronization

The arpeggiator's timing resolution is 24 clocks or pulses per quarter-note (PPQN). The note-on and note-off events it produces are aligned to these time increments. It can run standalone or synchronized to a MIDI clock presented at its **MCik** input. The **MIDI** and **MCik** inputs can be connected to a common source when the desired note and clock events both come from the same source – e.g., the **Sequencer MIDI Source module**, delivering both sequenced MIDI (including MIDI clocks) and live (keyboard) MIDI passed through the sequencer.

The arpeggiator has no clock output. It can be synchronized to sequencers and other devices by switching it to external clocking (see **Tempo/Timing – EXT MIDI CLOCK**) and feeding in a MIDI clock via the **MCik** input. Two or more arpeggiators can also be run in lockstep by connecting them to a common MIDI clock source. Or, if you can tolerate slightly less precision, you can simply run them unconnected and set them to the same tempo, since the arpeggiators resynchronize themselves to their internal clock sources whenever new chords are played into them.

# Functions

This section presents a point-by-point description of the buttons and other controls on the arpeggiator device surface. The discussion here is broken down into sections which correspond to the organization of the surface itself:

## Big Buttons Group

Capture Mode

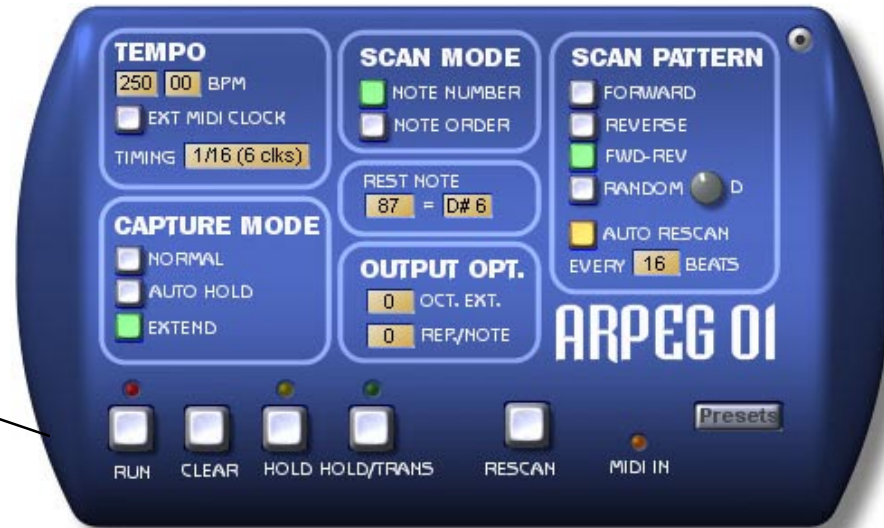
Scan Mode

Rest Note

Scan Pattern

Output

Tempo/Timing





## Big Buttons Group

This is the set of large buttons located at the bottom of the arpeggiator. These are essentially the "running" controls. Viewed as a group, they produce immediate, "big" changes. This distinguishes them from *settings* – i.e, all *other* controls on the surface – which can likewise be adjusted at any time, but whose effects are less "major" and not always immediately audible.

**RUN/STOP:** This is the On/Off button – it alternately activates and deactivates the arpeggiator. Stopping the arpeggiator clears the chord buffer and terminates the current output note, if any. Note that when the arpeggiator is not running, it passes *all* received MIDI events directly through to its output, and the other buttons in this group are disabled.

**CLEAR:** Pressing this button "empties" the arpeggiator (i.e., clears the chord buffer), permitting a completely new chord to be captured. The arpeggiator continues to run as before.

**CLEAR** also deactivates **HOLD** and **HOLD/TRANS** if they are active (see below).



**HOLD:** This function freezes the chord buffer, locking the current captured chord into the arpeggiator. Incoming MIDI note events are no longer captured, nor can they cause the notes which have already been captured to be removed. Instead, received MIDI note events are passed directly through to the output. This lets you "accompany" the arpeggiator live.

Once activated, **HOLD** can be deactivated only via **CLEAR** or **RUN/STOP**.

Note: **HOLD** is also activated automatically whenever **HOLD/TRANS** (see below) is activated.

**HOLD/TRANS:** Activating **HOLD/TRANS** will instantly activate **HOLD** (see above) if it is not already active, thus freezing the chord buffer. While **HOLD/TRANS** is active, the arpeggiator output can be "live-transposed" up or down (simple semitone transpose) from the MIDI keyboard.

The transpose produced by playing any note on the keyboard is equal to the offset of this note relative to middle C (MIDI 60). While **HOLD/TRANS** is active, the keyboard has no effect upon the captured chord, other than transposing it as described.

However, **HOLD/TRANS** (unlike **HOLD**) can be activated and deactivated freely. When **HOLD/TRANS** is deactivated, **HOLD** remains active, as does the last transpose value applied under **HOLD/TRANS**. This allows switching between transposition and accompaniment of the frozen (and still-transposed) chord.

The transpose produced under **HOLD/TRANS** is cleared via **CLEAR** or **RUN/STOP**, and is thus always zero whenever **HOLD** or **HOLD/TRANS** is first activated.

**RESCAN:** This function "restarts" the arpeggiator scan each time it is actuated. The next arpeggiator output note is the one which should appear "first", based on the current **Scan Mode** and other settings (for example, the lowest note in the current captured chord).

## Capture Mode

These settings control the way in which received MIDI notes are *captured* in (and removed from) the chord buffer. One of the four possible modes is always active. In all capture modes, the arpeggiator keeps track of the order in which captured notes arrived, in addition to recording note number (of course) and velocity for each note. Thus, the use of both **NOTE NUMBER** and **NOTE ORDER** scan modes is always possible, regardless of the **Capture Mode** setting.

The capture mode setting can be changed at any time. In some cases, after doing this, you may need to use **CLEAR** to empty the chord buffer completely (e.g., if you change from **AUTO HOLD** to **NORMAL** after releasing all keys).

**NORMAL:** In this mode, captured notes remain in the chord buffer only for as long as the corresponding key is held down. The arpeggiator output pattern varies dynamically as keys are released. When no keys are being held down, the arpeggiator produces no output.



**AUTO HOLD:** Under **AUTO HOLD**, captured notes remain in the chord buffer indefinitely, even after all held keys have been released. The arpeggio continues to play as if all keys which have been played were still being held. New notes continue to be captured as long as at least one key is still being held. The first note or chord which is played following the release of all keys begins a new capture "session", at the same time clearing all previously captured notes.

**EXTEND:** Notes are added to the chord buffer as they are played and remain in the chord buffer indefinitely. This capture mode permits notes to be added one at a time. **EXTEND** mode thus makes it easy to create melodic arpeggios – a particular note can appear multiple times at different points within the arpeggio.

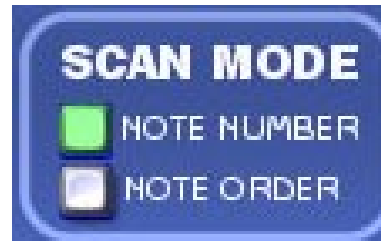
Note capture continues until the chord buffer is full (sixteen captured notes). The **CLEAR** button must be used to clear the chord buffer and silence the arpeggiator, or to capture a new chord.

Note that **Scan Mode** (see following section) must be set to **NOTE ORDER** in order to have the arpeggiator play the notes back in the order in which they were captured.



## Scan Mode

This control selects the basic method used to scan the captured chord and determine the next note to be played. The currently selected scan pattern (see below) produces a specific variation on the selected scan mode. The scan mode setting can be changed at any time.



**NOTE NUMBER:** Scanning of the captured chord is done on the basis of note numbers – e.g., from lowest note to highest note.

**NOTE ORDER:** Scanning of the captured chord is done on the basis of the time sequence in which the notes were captured.

## Rest Note

This control permits a specific MIDI input note to be designated as a "rest" note. Whenever the arpeggiator lands on this note during a scan of the captured chord, it inserts a rest – i.e., no output note is generated for this scan step. The inserted rest extends to include repeat notes, if any (see **Output – REPEATS / NOTE**).



The **REST NOTE** feature can be used to produce syncopated arpeggios in **NOTE ORDER** scan mode (hint: this is easiest with the **EXTEND** capture mode).

The **REST NOTE** setting can be changed freely while the arpeggiator is running, with the result that rests appear at different points in the output. Setting **REST NOTE** to 128 disables the rest note function.

The scan mode can be switched to **NOTE NUMBER** while the rest note function is active without any ill effects. This merely results in all rests (if there is more than one) occurring consecutively, since they are (of course) all generated from notes having the same note number.

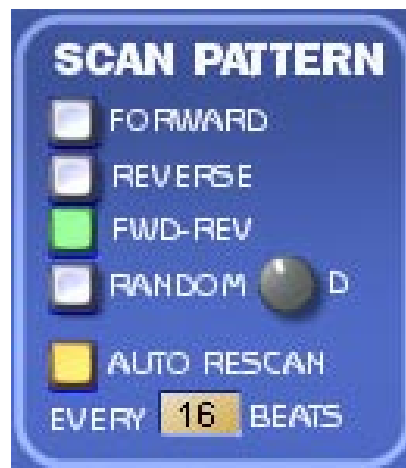
## Scan Pattern

These controls determine the specific way in which the captured chord is scanned. These settings work in tandem with the **Scan Mode** setting. The **Scan Pattern** setting can be changed at any time.

**FORWARD:** The captured chord is scanned in order of increasing note number (**Scan Mode** set to **NOTE NUMBER**) or in forward time sequence (**Scan Mode** set to **NOTE ORDER**).

**REVERSE:** The captured chord is scanned in order of decreasing note number (**Scan Mode** set to **NOTE NUMBER**) or in reverse time sequence (**Scan Mode** set to **NOTE ORDER**).

**FWD-REV:** Scanning of the captured chord scan alternates between **FORWARD** and **REVERSE** as described above, reversing itself each time it reaches the "end" of a scan (highest/lowest or first/last note). The notes at either "end" of a scan are not repeated when the direction reverses (i.e., these notes get played only once, not twice).



**RANDOM:** Chord scan follows a random pattern. The **D** (Random Depth) control sets the degree or range of randomness.

**RANDOM** scan behaves differently under different **Scan Mode** settings:

- If **Scan Mode** is set to **NOTE NUMBER**, then **RANDOM** scan is basically a variation on normal **FWD-REV**. The scan proceeds in single steps from one note number to the next without skipping over any notes, but reverses its direction at random. With **D** set to minimum, this mode is in fact equivalent to **FWD-REV**. With **D** set to maximum, the scan reverses itself after almost every note (and thus tends to "stick in place", continually alternating between two notes).

- If **Scan Mode** is set to **NOTE ORDER**, then **RANDOM** scan selects output notes randomly from among all possible output notes, taking the captured chord and all other settings (including **OCTAVE EXTEND** – see **Output**) into consideration. The **D** control confines this selection to a specific range of "scan steps" away from the previous note in either direction. With **D** set to minimum, the arpeggiator "sticks" on a single note. With **D** set to maximum, the arpeggiator may select any note within 16 scan steps of the previous note – in effect, almost totally random.

**AUTO RESCAN EVERY  $n$  NOTES:** When activated, this function causes the arpeggio scan to be restarted automatically after the specified number of arpeggiator output beats. This is useful for imposing a specific rhythmic count or "loop length" on the output which does not depend upon the number of captured notes or other scan settings.

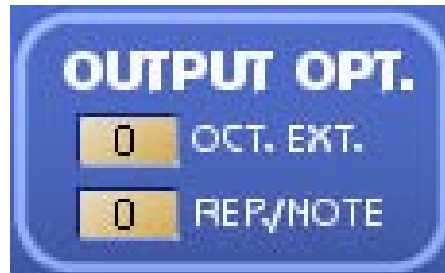
The auto rescan counter is reset whenever new chords are played on the MIDI keyboard – however, *not* while **HOLD** or **HOLD/TRANS** is active. It is also reset whenever the manual **RESCAN** function is actuated.

## Output

These settings provide control over additional functions for modification of the arpeggiator output. They can be changed at any time.

**OCTAVE EXTEND:** Produces cyclical upward transposition of the arpeggiator output by one or more octaves. The transpose amount is automatically "stepped" by one octave each time the arpeggiator completes a scan of the captured chord in the current scan direction. The captured chord is thus effectively extended into additional higher octaves as if the actual notes in the captured chord had been duplicated in those octaves. Setting **OCTAVE EXTEND** to zero disables it.

"Stepping" of the **OCTAVE EXTEND** transpose amount is always done in a manner which is consistent with the selected scan pattern. Assuming that **OCTAVE EXTEND** is enabled (i.e, it is set to 1 or higher):



- With **Scan Pattern** set to **FORWARD**, output transpose is stepped upward by one octave following each pass through the captured chord until the scan in the highest octave (as specified by the **OCTAVE EXTEND** setting) is complete. Transpose is then reset to 0 and the cycle repeats.

- With **Scan Pattern** set to **REVERSE**, output transpose is stepped downward by one octave following each pass until a scan with a transpose of 0 is complete. Transpose is then reset to the highest octave (as specified by the **OCTAVE EXTEND** setting) and the cycle repeats.

- With **Scan Pattern** set to **FWD-REV**, the scan direction is not reversed upon completion of a single forward scan of the captured chord, as would normally occur. Instead, output transpose is stepped upward by one octave and another forward

scan is done. This repeats until the forward scan in the highest octave is complete – at this point, the scan direction reverses and the reverse scan is done, still in the highest octave. Subsequently, output transpose is stepped downward by one octave each time a reverse scan is complete (likewise, with no scan direction reverse) until a scan with a transpose of 0 is complete. The scan direction then switches again to forward and the entire cycle repeats.

- With **RANDOM** scan patterns, the **OCTAVE EXTEND** setting correspondingly extends (by the specified number of octaves) the set of possible output notes which the random scan can produce – again, as if the actual notes in the captured chord have been replicated in higher octaves.

**REPEATS/NOTE:** When set to values other than zero, this setting causes the arpeggiator to repeat each output note for the specified number of additional beats before proceeding to scan the captured chord for a new note. **REPEATS/NOTE** works with all scan modes and scan patterns.

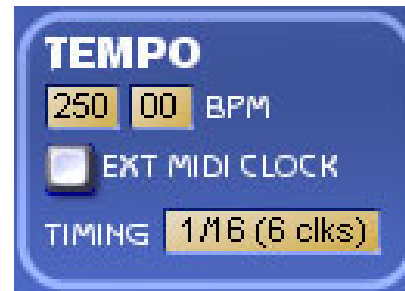
## Tempo/Timing

This group includes controls for setting the basic tempo of the arpeggiator and the arpeggiator beat length, as well as for selecting the arpeggiator clock source.

### TEMPO

Normally, the arpeggiator runs on its own internal clock source (**EXT MIDI CLOCK** off – see below). With this clock source, the running tempo is set directly via the **TEMPO** control. It can be adjusted over a range of 30 .. 250 BPM (Beats Per Minute), including fractional BPM values.

When **EXT MIDI CLOCK** is activated, the **BPM** display indicates the tempo of the incoming MIDI clock stream on the **MCik** input. It defaults to 60.00 BPM if no MIDI clock is being received on this input.



### EXT MIDI CLOCK

When this option is activated, the arpeggiator runs synchronized to an external MIDI clock stream connected to its **MCik** input. Arpeggiator timing is then determined directly by the MIDI clock stream, whose tempo is detected and displayed in the **TEMPO** field (see above).

### TIMING

This sets the number of clocks which constitute one arpeggiator output *beat*. The arpeggiator normally outputs a new (or repeated) note on every beat. The corresponding musical note value is also indicated (e.g., 6 clocks = 1/16th note).



# Known Issues

We're at work on these issues. But we wanted to get the arpeggiator into your hands as soon as possible. The following list is included just so that you also know about a few things we *already* know about:

- The MIDI-based timing resolution restricts the range of available note width settings at lower tempos.
- There is no **MIDI** or other clock output from the device.

# Index

## A

AUTO HOLD 8  
AUTO RESCAN EVERY n NOTES 11

## B

basic method 9  
buttons 6

## C

CAPTURE MODE 8  
captured notes 8  
channel messages 4  
channel number 4  
chord buffer 3, 8  
CLEAR 6, 8

## E

EXT MIDI CLK 13  
EXT MIDI CLOCK 4  
EXTEND 10  
EXTEND 1 8

## F

first note 11  
FORWARD 11, 12  
fractional BPM values 13  
freezing the chord buffer 7  
FWD-REV 11, 12

## H

highest note 9  
HOLD 4, 6  
HOLD/TRANS 6

## I

internal chord buffer 3  
internal timebase 13  
Introduction 2

## L

last note 11  
loop length 11  
lowest note 9

## M

MClk input 4  
MIDI 60 7  
MIDI Clock messages 4  
MIDI input 3  
MIDI note events 3  
modwheel 4  
monophonic 3

## N

NORMAL 8  
NOTE NUMBER 8, 9, 10, 11  
NOTE ORDER 8, 9, 10, 11  
note-off events 4  
note-off messages 3  
note-on events 4  
note-on messages 3

## O

On/Off button 6  
output beats 11  
Output events 4  
output note 11

## P

pitch bend 4  
PPQN 4  
Problems 14  
Pulsar devices 4  
pulses per quarter-note (PPQN) 4

## R

RANDOM 12  
Random Depth 11  
REPEATS / NOTE 12  
RESCAN 11  
REVERSE 11, 12  
RUN/STOP 6

## S

SCAN MODE 11  
SCAN PATTERN 12  
semitone transpose 7  
Sequencer MIDI Source module 4  
settings 6  
source 3  
SWEEP TRANSPOSE 12  
System Common 4  
System Exclusive 4  
System Realtime 4

## T

timebase 13  
timing resolution 4  
transpose 7  
transposition 12