

Fripp States During Ad-hoc Loop Overdubbing

During Ad-hoc Loop Overdubbing, your ad-hoc sequence can extend through multiple iterations of the loop. For example, you may set an 8-bar phrase to playback in a loop, and improvise 32 bars over it. The ad-hoc sequence would include all 32 bars.

Then again, you might start and stop numerous different ad-hoc sequences while the playback sequence continues to loop. During Ad-hoc Loop Overdubbing, Fripp ends an ad-hoc sequence (and starts the next one) when you're silent for an entire iteration of the loop. It also ends an Ad-hoc Sequence when you stop playback.

The following diagram illustrates the various states that can occur during Ad-hoc Loop Overdubbing.

You begin playback with the Loop button toggled *on*, and Fripp enters the *StartingAdhocLoopedOverdubbedRecording* state. It creates a new sequence, and transitions immediately to the *PendingFirstLoopedOverdubbedInput* state. The start time of the sequence is the time that playback began. Fripp sets a timer to go off at the beginning of the next loop. If you don't play anything before the timer goes off, Fripp transitions to the *RestartingAdhocLoopedOverdubbedSequence* state, which resets the start timestamp of the sequence to the start time of the current loop iteration and transitions immediately back into the *PendingFirstLoopedOverdubbedInputState*. Another timer is set to go off at the beginning of the next loop.

Fripp remains in the *PendingFirstLoopedOverdubbedInput* state, and continues to reset the start time of the sequence, until you provide some MIDI Input. When you do that, Fripp cancels the timer, records that input, and enters either the *PendingLoopedOverdubbedInputCompletion* or *PendingMoreLoopedOverdubbedInput* state, depending on the input. Fripp records your subsequent inputs and bounces between the *PendingLoopedOverdubbedInputCompletion* and *PendingMoreLoopedOverdubbedInput* states, until you do something to signal the end of the current sequence.

The *PendingMoreLoopedOverdubbedInput* state is similar to the *PendingMoreInput* state during Ad-hoc Recording mode, with a twist. Both states try to infer the end of a sequence by measuring silence, but they calculate the length of the silence (the *MaxSilenceInterval*) differently. For the *PendingMoreLoopedOverdubbedInput* state:

```
MaxSilenceInterval = <timespan from now to the end of the next loop iteration>
```

So, upon entering the *PendingMoreLoopedOverdubbedInput* state, Fripp sets a silence timer to the calculated *MaxSilenceInterval*. If you play some more, Fripp cancels the timer, and enters either the *PendingOverdubbedInputCompletion* state or the *PendingMoreOverdubbedInput* state (depending on your input). However, if you allow the playback to continue through the end of the current loop and another entire iteration of it, the silence timer goes off, Fripp ends the current sequence with your last input, and transitions back to the *StartingAdhocLoopedOverdubbedRecording* state, to begin a new sequence.

Finally, if while in the *PendingMoreLoopedOverdubbedInput* state, you stop playback before the silence timer goes off, Fripp cancels the timer, ends the current sequence with your last MIDI input, and transitions back to the *Coordinating Recording* state.

[Next: Fripp States during Explicit Recording](#)