

Anwendungsentwicklung: Benutzeroberfläche zur Ansteuerung der Effektblöcke des Soundmoduls SD1000 der Firma Ketron

Inhaltsverzeichnis

Inhaltsverzeichnis.....	I
Abbildungsverzeichnis.....	III
1 Aufgabenstellung.....	1
2 Quellprojekt.....	2
3 Herausforderung	3
4 Hilfsmittel	4
5 Ctrlr Konzept.....	5
5.1 Ctrlr arbeits Modi	6
5.2 Panels	7
5.3 Modulatoren	8
5.4 Modulator Attribute.....	9
5.5 Combo Modulatoren	11
6 Anwendungsorientierte Beispiele	12
6.1 Panel Eigenschaft über Slider regulieren	13
6.2 Control Change Slider.....	14
6.3 SysEx Slider	15
6.4 SysEx Slider mit Panel Output Channel	16
6.5 Combo-Modulator für Presets	17
6.6 NRPN Messages Slider.....	18
6.7 NRPN Kurzzusammenfassung.....	20
6.8 NRPN Message Combo (Preset)	21
6.8.1 lua-Script: setDelayPresetEfx	23
6.8.2 lua-Script: setDelayPresetEfx_	23
6.8.3 lua-Script: setDelayPresetEfx_Parameters.....	24
7 Fehlende Informationen.....	26
7.1 Insert FX On	27
7.2 Insert MFX Chorus/Flanger/Phaser/Tremolo/Rotary Controls	28
7.3 High-Quality Distortion Preset Parameter	29
7.4 Aktivierung nicht vollständig dokumentierter Controls.....	30
8 Speicher und Lade Realisierung.....	31
8.1 Vorkonfiguration	32
8.2 Speichervorgang.....	33
8.3 Ladevorgang	34
8.4 Realisierungsergebnis.....	35
9 LUA-Scripts	36

9.1	EFX Preset Scripts	37
9.2	Programm Selektion	38
9.3	Mega Panic Reset (Global Reset).....	39
9.4	Channel Reset.....	40
9.5	Speichern und Laden	41
9.5.1	Vorkonfiguration	41
9.5.2	Speichern.....	42
9.5.3	Laden	42
10	SD 1000 Panel.....	43
10.1	Panel Controls	44
10.2	Master Controls.....	45
10.3	EFX Controls.....	46
10.4	HQ-Distortion Controls.....	47
10.5	Development und Debug Controls.....	48
10.5.1	Enable MIDI Message On Loading Controls	48
10.5.2	Preset Values zum Laden und Speichern	48
10.5.3	Development Controls.....	49

Abbildungsverzeichnis

Abbildung 1: Soundmodul SD1000 der Firma Ketron	1
Abbildung 2: Ausgangspanel	2
Abbildung 3: leeres Panel im Normal Modus.....	6
Abbildung 4: leeres Panel im Editor Modus	6
Abbildung 5: Modulator - Slider - Auswahlliste.....	8
Abbildung 6: Slider-Modulator mit MIDI Eigenschaftsfeld	9
Abbildung 7: Slider Modulator Eigenschaften: modulatorValue und modulatorValueExpression.....	10
Abbildung 8: simple Auswahlliste uiComboContent	10
Abbildung 9: Auswahlliste mit modulatorWert Definition.....	10
Abbildung 10: MIDI Kanalwahl über das Dateimenü	13
Abbildung 11: Slider Modulator MIDI Channel	13
Abbildung 12: Slider Modulator „Volume“ mit CC MIDI Nachricht	14
Abbildung 13: Slider Modulator „Master Volume“ mit SysEx MIDI Nachricht	15
Abbildung 14: Slider Modulator „Load EFX1 (3a) Preset“ mit SysEx MIDI Nachricht	16
Abbildung 15: Combo Modulator „Reverb type“ mit SysEx MIDI Nachricht	17
Abbildung 16: NRPN MIDI Implementierung des SD1000.....	18
Abbildung 17: Slider Modulator „Input Gain“ mit NRPN MIDI Nachricht	18
Abbildung 18: Modulatoren Gruppe „Delay Controls“ des EFX1	21
Abbildung 19: Eigenschaften und Attribute des Loading-Modulators am Beispiel Delay Preset EFX1	22
Abbildung 20: Storing-Modulator „Efx1_Delay_Preset_Value“ – Speichern & Laden des Presets	23
Abbildung 21: EnableSingleEfxMidiMessageForLoading - Anpassung.....	30
Abbildung 22: MIDI Monitor – programList-Modulator.....	38
Abbildung 23: MIDI Monitor - Global Reset.....	39
Abbildung 24: MIDI Monitor - Channel Reset	40

1 Aufgabenstellung

Ziel dieser Arbeit ist eine grafische Oberfläche zur Ansteuerung des Soundmoduls SD1000 der Firma Ketron (Abbildung 1) zum Anlegen und Laden eigener Voreinstellungen.

Zum Erstellen der Oberfläche wurde die Software Ctrlr MIDI verwendet, welche es ermöglicht grafische Elemente auf einem Panel zu erzeugen und diesen dann bestimmte MIDI-Befehle zuzuordnen.



Abbildung 1: Soundmodul SD1000 der Firma Ketron

2 Quellprojekt

Basis dieser Arbeit ist das im Sommersemester 2016 entwickelte Ctrlr Panel (Abbildung 2) des Wahlpflichtfach MIDI-Musikinstrumentenschnittstelle von Lömker Mario und Layes Aryan. Dieses wird mit der Funktion erweitert um die Möglichkeit zu besitzen vorgenommene Effekt-Einstellungen speichern und laden zu können. Somit kann der Nutzer eigene Voreinstellungen, auch „Presets“ genannt, definieren und bei Bedarf laden.



Abbildung 2: Ausgangspanel

3 Herausforderung

Die Herausforderung bei der Realisierung des Projekts ist die Tatsache, dass einerseits die Dokumentation des Soundmoduls SD1000 nicht alle erforderlichen Befehlssätze enthält, der SD1000 keine Status Rückmeldungen liefert und die verwendete opensource Software Ctrlr keine offizielle Dokumentation bietet. Die Einarbeitung und Umsetzung des Projekts muss selbständig angegangen werden.

4 Hilfsmittel

Da das Benutzerhandbuch des Ketron SD1000 nicht alle benötigten Informationen zum Ansteuern des Soundmoduls enthält werden weitere Datenblätter zur MIDI-Implementierung berücksichtigt.

Diese Dokumente liegen dem Projekt in digitaler Form bei:

- SD1000_web.pdf: Ketron SD1000 Benutzerhandbuch
- Firm5716-EK_decrypted.pdf: Dream S.A.S. France Handbuch der Firmware 5716 (V1.3)

5 Ctrlr Konzept

Im folgenden Abschnitt wird auf die Terminologie und das Konzept hinter der opensource Software Ctrlr eingegangen.

Diese Dokumentation bezieht sich auf Ctrlr mit den folgenden Versionsdaten.

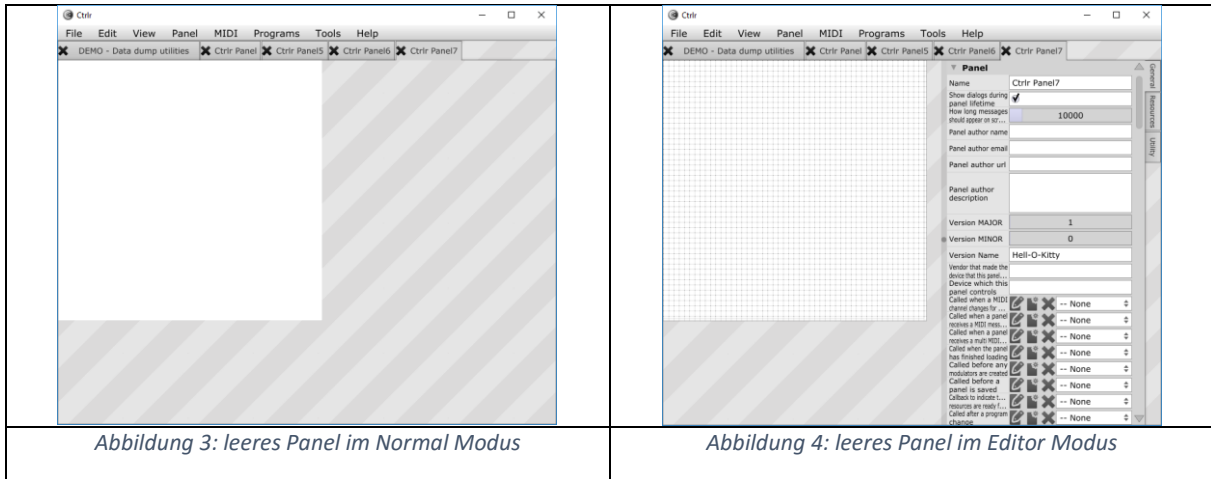
Version = 5.3.201, Build date = Wed Mar 30 00:52:44 CEDT 2016, Branch = Nightly, Juce = 4.0.2, libusb = 1.0.19, liblo = 0.28,
--

5.1 Ctrlr arbeits Modi

Ctrlr erlaubt einerseits fertige Panels zum Ansteuern von MIDI Hardware zu nutzen. Bietet aber auch die Möglichkeit Panels zu entwickeln (erstellen, bearbeiten und erweitern). Ctrlr arbeitet daher in zwei unterschiedlichen Panel Modi, dem Editor Modus und dem Normal Modus.

Im Normal Modus (Abbildung 3) ist nur das Panel sichtbar und kann zum Ansteuern der MIDI Hardware verwendet werden. Im Editor Modus (Abbildung 4) wird neben dem Panel, am rechten Fensterrand die Eigenschaften Listen (engl. property lists) des aktuell markierten Modulators aufgelistet.

Zwischen den beiden Modi kann mithilfe der Tastenkombination STRG+E gewechselt werden.



5.2 Panels

Jedes Projekt in Ctrlr verwendet ein Panel und wird daher in einer „*.panel“ Datei gespeichert. Panels sind Container für Modulatoren. Zusätzlich sind einem Panel MIDI Verbindungseinstellungen, Informationen zur Versionsverwaltung und viele weitere Eigenschaften zugeordnet. Wird ein Lua-Script erzeugt ist dies ebenfalls im Panel gespeichert und über das gesamte Panel aufrufbar.

5.3 Modulatoren

Unter Modulatoren werden in Ctrlr alle Elemente bezeichnet die auf einem Panel platziert und angezeigt werden können. Modulatoren sind Container für Attribute (engl. properties) und sind mit einer Komponente ausgestattet.

In diesem Projekt wurden folgende Modulatoren Typen verwendet:

- uiSlider
- uiToggleButton
- uiButton
- uiLabel
- uiGroup
- uiTabs
- uiCombo

Modulatoren können im Editor Modus mit einem Rechtsklick im Panelbereich (schwarz schraffiert) hinzugefügt werden (Abbildung 5).

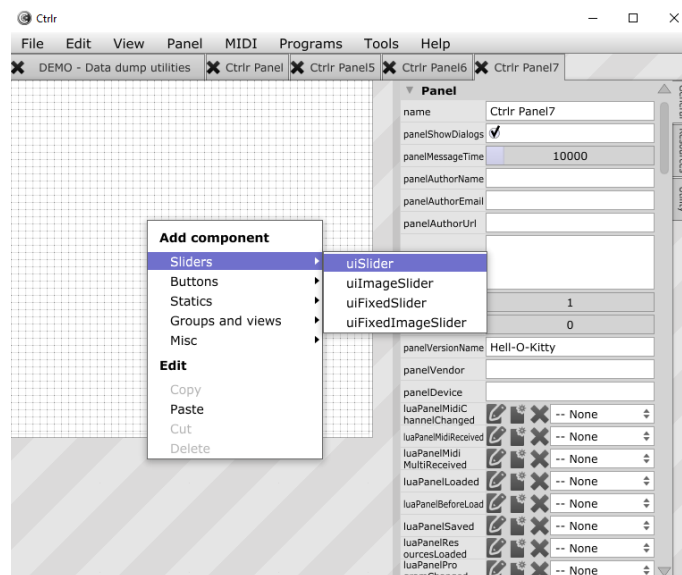


Abbildung 5: Modulator - Slider - Auswahlliste

5.4 Modulator Attribute

Modulatoren besitzen eine Vielzahl von Attributen. Diese können im Editor Modus eingesehen werden. Dieser Abschnitt behandelt die zentralen und wichtigsten Attribute von Modulatoren.

Modulatoren mit dem Eigenschaftsfeld MIDI (Abbildung 6) bieten die Möglichkeit MIDI Nachrichten zu generieren und senden. Modulatoren die diese Fähigkeit besitzen sind zum Beispiel:

- uiSlider
- uiToggleButton
- uiButton
- uiCombo

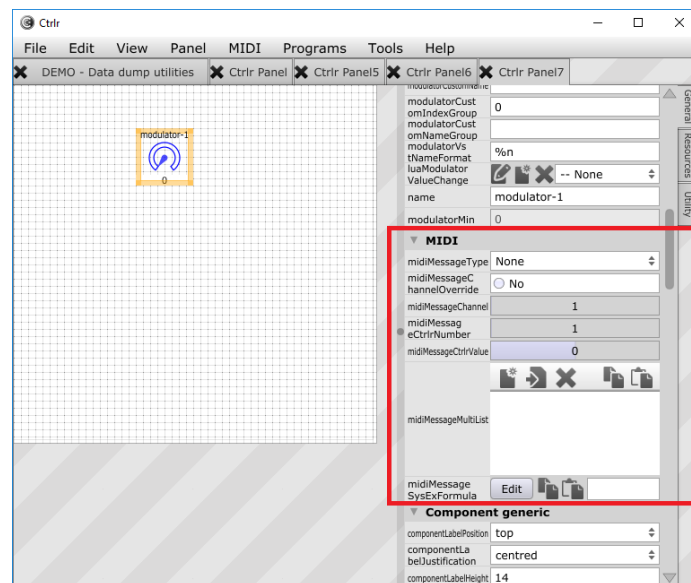


Abbildung 6: Slider-Modulator mit MIDI Eigenschaftsfeld

MIDI Modulatoren können die MIDI Nachricht dynamisch generieren. Das heißt es ist zum Beispiel möglich den Wert eines Slider-Modulators in der MIDI Nachricht zu berücksichtigen und übermitteln.

MIDI Modulatoren besitzen daher die Attribute „modulatorValue“ und „modulatorValueExpression“ (Abbildung 7). Das Attribut modulatorValue entspricht bei einem Slider dem aktuellen Zahlenwert. Das Attribut modulatorValueExpression ermöglicht nach mathematischer Definition wie der Zahlenwert übertragen werden soll. Diese Eigenschaft wird nicht weiter behandelt, Informationen zu Expressions sind auf der Website: <http://ctrlr.org/expressions-in-ctrlr-2/> verfügbar.

Achtung: Slider-Wertänderungen funktionieren nur im Normal Modus.

Ctrlr Konzept

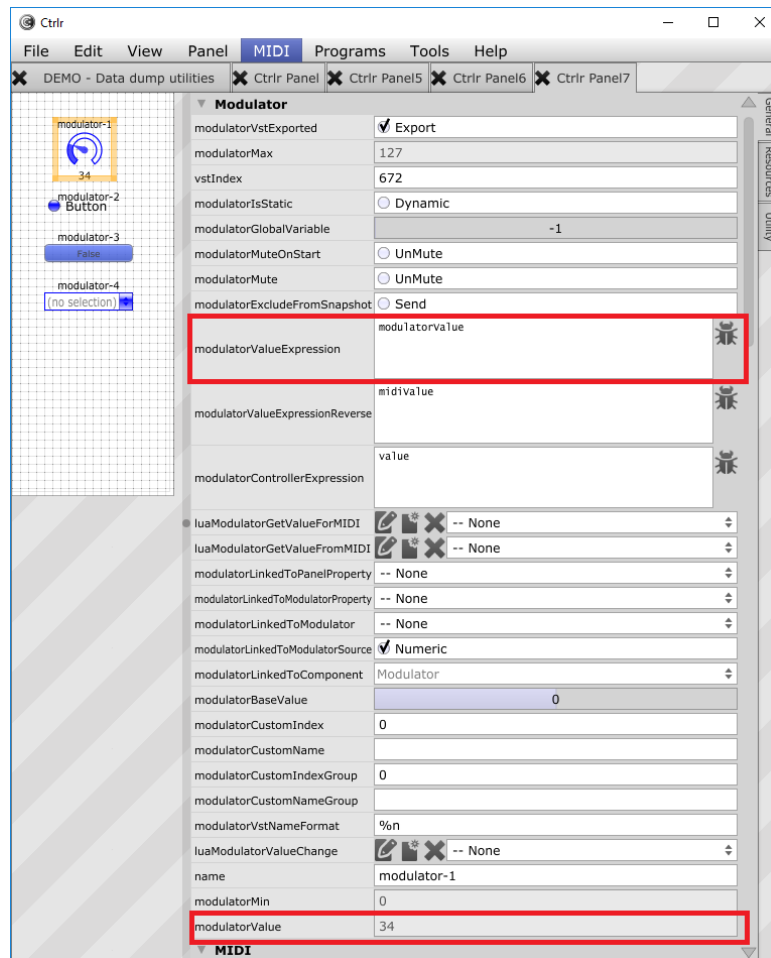


Abbildung 7: Slider Modulator Eigenschaften: modulatorValue und modulatorValueExpression

Bei uiCombo-Modulatoren bietet das Komponenten Attribut „uiComboContent“ die Möglichkeit eine Auswahlliste zu erstellen. Wird die Auswahlliste wie in Abbildung 8 gesetzt entspricht das Attribut modulatorValue dem Arrayindex beginnend bei 0.

Mit der in Abbildung 9 dargestellten Auswahlliste entspricht das Attribut modulatorValue dem Integer-Wert X der Zeilendefinition $abc = X$.

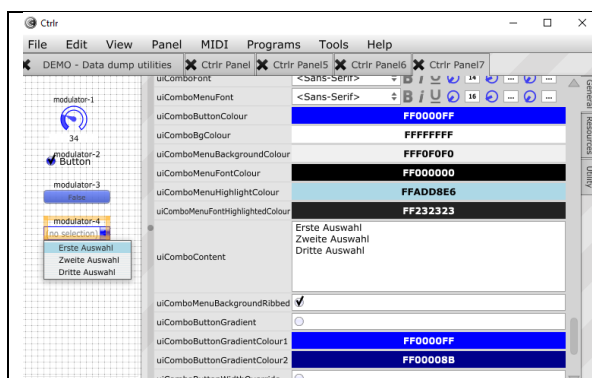


Abbildung 8: simple Auswahlliste uiComboContent

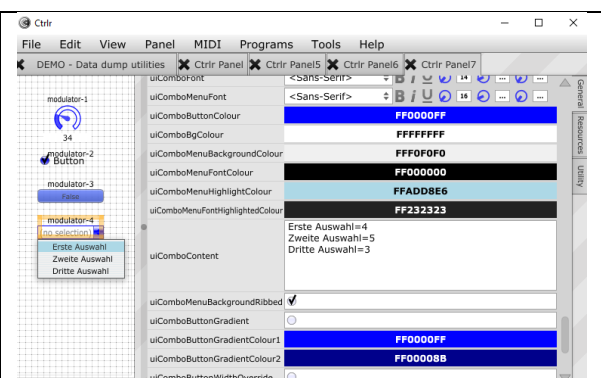


Abbildung 9: Auswahlliste mit modulatorWert Definition

Erste Auswahl -> modulatorValue = 0
Zweite Auswahl -> modulatorValue = 1
Dritte Auswahl -> modulatorValue = 2

Erste Auswahl -> modulatorValue = 4
Zweite Auswahl -> modulatorValue = 5
Dritte Auswahl -> modulatorValue = 3

5.5 Combo Modulatoren

Combo Modulatoren verwenden sogenannte „mappedValues“. Wenn in der „Combo contents“ Auflistung keine definitionen „abc = x“ steht entspricht der Modulator Wert dem Arrayindex (Zeilennummer beginnend bei 0).

6 Anwendungsorientierte Beispiele

Im folgenden Abschnitt werden einige Anwendungsorientierte Beispiele beschrieben um das Konzept hinter der opensource Software Ctrlr zu verstehen und selbst anwenden zu können.

Die Beispiele sollen dabei helfen zu verstehen wie MIDI Attribute von Modulatoren für die unterschiedlichen MIDI Nachrichten Typen CC, SysEx und NRPN eingestellt werden müssen. Die Beispiele zur Übertragung von MIDI Nachrichten basieren auf dokumentierten MIDI Implementierungen des SD1000.

6.1 Panel Eigenschaft über Slider regulieren

Da das SD1000 Sound Modul auf 16 MIDI Kanälen arbeitet benötigt das Panel eine Schaltfläche zum Auswählen des gewünschten MIDI Kanals. Ctrlr bietet die Möglichkeit den Kanal über das Dateimenü (Abbildung 10) auszuwählen. Um im Panel einen direkten Zugriff, ohne Umweg übers Dateimenü, auf die MIDI Output Channel Eigenschaft zu erhalten wird ein Modulator (Abbildung 11) in diesem Abschnitt beschrieben der diese Aufgabe erfüllt.

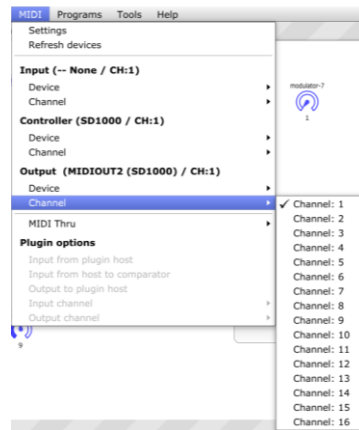

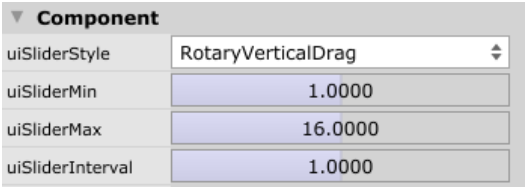


Abbildung 10: MIDI Kanalwahl über das Dateimenü

Beispiel



Abbildung 11: Slider Modulator MIDI Channel

<p>um den Slider-Modulator mit der MIDI Output Channel Eigenschaft des Panels zu verbinden, muss im Modulator-Abschnitt die Eigenschaft „modulatorLinkedToPanelProperty“ entsprechend gewählt werden:</p> <p>modulatorLinkedToPanelProperty: panelMidiOutputChannelDevice</p>	
<p>Da ein Slider mit Standardeinstellungen 128 Werte (0 bis 127) Anzeigt es aber nur 16 MIDI Kanäle gibt (1 bis 16) müssen folgende Eigenschaften des Sliders überarbeitet werden.</p> <p>uiSliderMin: 1 uiSliderMax: 16</p>	

Mit diesem Modulator lässt sich der MIDI Kanal ohne Umwege innerhalb des Panels wählen. Wird der aktuelle Kanal bei einer MIDI Nachricht benötigt existieren die Platzhalter **yy** und **y**. Auf die Platzhalter wird in den weiteren Beispielen eingegangen.

6.2 Control Change Slider

In diesem Abschnitt wird auf die Einstellungen eines Modulators eingegangen deren Resultat Control Change (CC) MIDI Nachrichten ergeben. Im SD1000 Handbuch ab Seite 58 sind diese unter der Bezeichnung „CTRL“ gelistet.

Beispiel

Um die Kanal Lautstärke (engl. Volume) des SD1000 einzustellen wird ein Slider-Modulator mit den folgenden Einstellungen erzeugt.



Abbildung 12: Slider Modulator „Volume“ mit CC MIDI Nachricht

	CTRL 07	BnH 07H cc	Volume (default=100)	
Type:	CC			<div> MIDI <div> midiMessageType <div>CC</div> </div> midiMessageChannelOverride <div>No</div> </div> <div> midiMessageChannel <div>1</div> </div> <div> midiMessageCtrlNumber <div>7</div> </div> <div> midiMessageCtrlValue <div>0</div> </div> <div> <div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div> </div> <div> midiMessageMultiList <div></div> </div> <div> midiMessageSysExFormula <div>Edit</div> <div></div> </div>

Midi Ausgabe

Ch:[1] No:[7] Val:[72] RAW:[b0 07 48]
Ch:[4] No:[7] Val:[72] RAW:[b3 07 48]

Mit diesem Modulator lässt sich für jeden Kanal separat die Lautstärke regeln.

6.3 SysEx Slider

In diesem Abschnitt wird ein Ctrlr Modulator erzeugt der bei Wertänderung eine MIDI Nachricht im Sysex Format sendet. Im SD1000 Handbuch ab Seite 59 sind diese unter der Bezeichnung „Standard Sysex“ oder „Sysex“ gelistet.

Beispiel

Um die Master Lautstärke des SD1000 einzustellen wird ein Slider-Modulator mit den folgenden Einstellungen erzeugt.

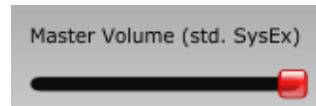


Abbildung 13: Slider Modulator „Master Volume“ mit SysEx MIDI Nachricht

	Standard Sysex	F0H 7FH 7FH 04H 01H 00H II F7H	Master volume (II=0 to 127, default 127) (note 4). Not reset by GS reset	
Type: SysEx SysEx Formular: F0 7F 7F 04 01 00 xx F7 xx entspricht dem aktuellen Modulator Wert in Hexadezimal				

Midi Ausgabe

Ch:[--] No:[----] Val:[----] RAW:[f0 7f 7f 04 01 00 7f f7]
--

Mit diesem Modulator lässt sich die Kanalübergreifende Lautstärke (Master Volume) regeln.

6.4 SysEx Slider mit Panel Output Channel

In diesem Abschnitt wird ein Ctrlr Modulator erzeugt der bei Wertänderung eine MIDI Nachricht im SysEx Format sendet und dabei die Kanalwahl berücksichtigt.

Beispiel

Um auf einem Effektblock des SD1000 ein Preset zu laden fordert die Sysex Nachricht laut Dokumentation insg. drei variable Teile:

- p= Channel (0x00 bis 0x15) – MIDI Kanal
- vv= Preset (0x00 bis 0x13) – Preset
- efx= 0x3a oder 0x3b – Effektblock

Um das Beispiel einfach zu halten wird der Effektblock mit 0x3a festgelegt. Die folgenden Modulator Einstellungen führen zu variablen Channel- und Presetwahl.



Abbildung 14: Slider Modulator „Load EFX1 (3a) Preset“ mit SysEx MIDI Nachricht

	SYSEX	F0H 26H 7BH 1AH 1pH vv efx F7H	Load EXF Channel , p= Channel (00H-0FH) , vv=(00H-13H) 00H : Chorus1 01H : Chorus2 02H : Phaser1 03H : Phaser2 04H : Flanger1 05H : Flanger2 06H : Tremolo1 07H : Tremolo2 08H : Delay1 09H : Delay2 0AH : Echo Pan 0BH : Rotor Slow 0CH : Rotor Fast 0DH : Organ Overdrive* 0EH : Valve Distortion* 0FH : Power Distortion* 10H : Metal Distortion* 11H : Tweed Amp. Simulator 12H : British Amp. Simulator 13H : Jazz Amp. Simulator vv= 7FH (off) efx= 3AH (EFX 1) , 3BH (EFX 2) *=available only on EFX 1
Maximum value:	19		Component Slider style: RotaryVerticalDrag Minimum value: 0.0000 Maximum value: 19.0000 Interval: 1.0000 Double clicked setting: <input checked="" type="checkbox"/> Enabled The value to set when double clicked: 0.0000
Type: SysEx SysEx Formula: F0 26 7B 1A 1y xx 3A F7 y entspricht dem aktuellen „MIDI output channel“ des Panels in Hexadezimal xx entspricht dem aktuellen Modulator Wert in Hexadezimal	MIDI MIDI message type: SysEx Override panel MIDI channel: <input type="radio"/> No MIDI Channel: 1 MIDI Controller number: 0 MIDI Controller value: 0 Multi Message list: SysEx Formula: <input type="button" value="Edit"/> F0 26 7B 1A 1y xx 3A F7		

Midi Ausgabe

Ch:[--] No:[----] Val:[----] RAW:[f0 26 7b 1a 10 09 3a f7]

Dieser Modulator in Verbindung mit der MIDI Kanalwahl erlaubt das MIDI-Kanal spezifische Laden eines Presets in Effektblock „EFX 1“ 0x3a. Nachteil: Presetnamen sind nicht sichtbar. Workaround: 17

6.5 Combo-Modulator für Presets

Da ein Slider-Modulator nur eine Zahl anzeigt, Presets aber Namen besitzen kann mit Hilfe eines Combo-Modulators eine Auswahlliste für Presets erzeugt werden die mehr Informationen anzeigt.

Wichtig dabei ist das in dem Eigenschaftsfeld „uiComboContent“ die Visuellen-Werte einem Integerwert „zugeordnet“ werden. Der Visuelle-Wert wird im Auswahlmeneu angezeigt, der Integerwert steht der MIDI-MESSAGE zur Verfügung. Wird keine Zuordnung angegeben, entspricht der Integerwert dem Array-Index beginnend bei 0.

Beispiel



Abbildung 15: Combo Modulator „Reverb type“ mit SysEx MIDI Nachricht

	SYSEX	F0H 26H 7BH 00H 00H vv F7H	Reverb type (vv=0 to 7), default = 03H 10H : Room1 11H : Room2 12H : Room3 13H : Hall1 14H : Hall2 15H : Plate 16H : Delay 17H : Pan delay (note 5)	
uiComboContent Room1=16 Room2=17 Room3=18 Hall1 (std)=19 Hall2=20 Plate=21 Delay=22 Pan delay=23		<div> <div>Component</div> <div> <div>uiComboArrowColour</div><div>FF0000FF</div> <div>uiComboOutlineColour</div><div>FF0000FF</div> <div>uiComboTextColour</div><div>FF000000</div> <div>uiComboTextJustification</div><div>centred</div> <div>uiComboFont</div><div><Sans-Serif> B I U ↺ ↻</div> <div>uiComboMenuFont</div><div><Sans-Serif> B I U ↺ ↻</div> <div>uiComboButtonColour</div><div>FF0000FF</div> <div>uiComboBgColour</div><div>FFFFFFF</div> <div>uiComboMenuBackgroundColour</div><div>FF0F0F0</div> <div>uiComboMenuFontColour</div><div>FF000000</div> <div>uiComboMenuHighlightColour</div><div>FFADD8E6</div> <div>uiComboMenuFontHighlightedColour</div><div>FF232323</div> <div>uiComboContent</div><div>Room1=16 Room2=17 Room3=18 Hall1 (std)=19 Hall2=20 Plate=21</div> </div> </div>		
Type: SysEx Formular: xx entspricht dem aktuellen Modulator Wert in Hexadezimal	SysEx F0 26 7B 00 00 xx F7	<div> <div>MIDI</div> <div> <div>midiMessageType</div><div>SysEx</div> <div>midiMessageChannelOverride</div><div>No</div> <div>midiMessageChannel</div><div>1</div> <div>midiMessageCtrlrNumber</div><div>1</div> <div>midiMessageCtrlrValue</div><div>0</div> <div> <div>⚙️ ➡️ ✖️</div> <div>📄 📄</div> </div> <div>midiMessageMultiList</div> <div> <div>midiMessageSysExFormula</div> <div>Edit</div> <div>📄 📄</div> <div>F0 26 7B 00 00 xx F7</div> </div> </div> </div>		

6.6 NRPN Messages Slider

In diesem Abschnitt wird gezeigt welche Modulator Einstellungen vorgenommen werden müssen um NRPN MIDI Nachrichten zu übertragen. NRPN Messages bestehen im Grunde aus drei CC Messages (Abbildung 16).

CTRL 99	BnH 63H vv	NRPN high
CTRL 98	BnH 62H vv	NRPN low
CTRL 06	BnH 06H cc	Data entry : provides data to RPN and NRPN

Abbildung 16: NRPN MIDI Implementierung des SD1000

Beispiel: Input Gain des MFX Mix Effektkanals EFX 1 (0x3a)

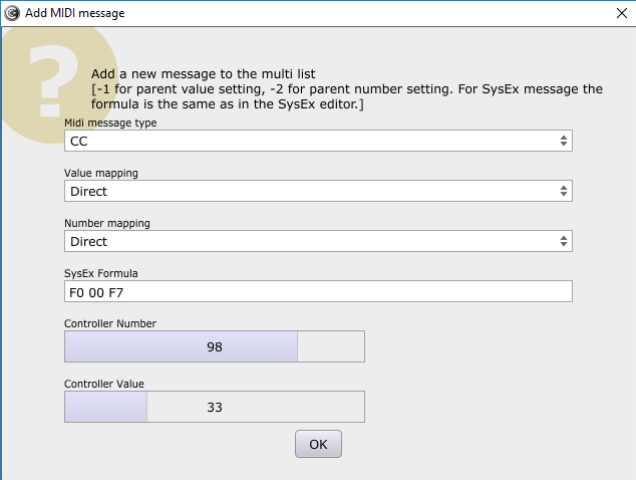
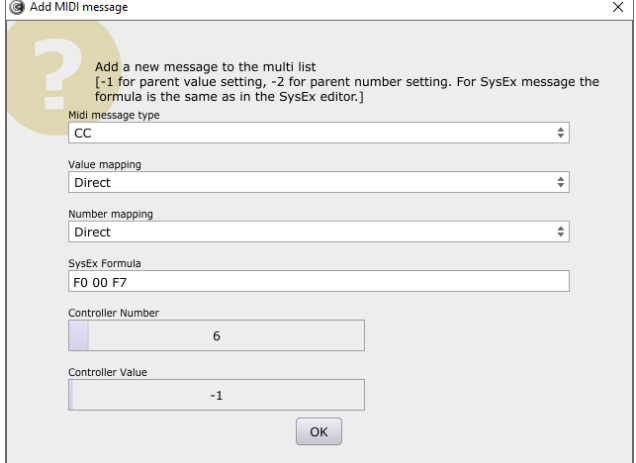


Abbildung 17: Slider Modulator „Input Gain“ mit NRPN MIDI Nachricht

NRPN High	NRPN Low	Description
3xh	21h	Input gain, 0 to 7Fh

<p>Type: Multi</p> <p>Multi Message List CC,Direct,Direct,99,58,F0 00 F7 CC,Direct,Direct,98,33,F0 00 F7 CC,Direct,Direct,6,-1,F0 00 F7</p> <p>99 entspricht NRPN high 58 = 0x3a entspricht EFX1</p> <p>98 entspricht NRPN low 33 = 0x21 entspricht Input gain</p> <p>6 entspricht Data entry -1 Platzhalter für Modulator Wert</p>	<p>MIDI</p> <p>MIDI message type: Multi</p> <p>Override panel MIDI channel: <input type="radio"/> No</p> <p>MIDI Channel: 1</p> <p>MIDI Controller number: 0</p> <p>MIDI Controller value: 0</p> <p>Multi Message list CC,Direct,Direct,99,58,F0 00 F7 CC,Direct,Direct,98,33,F0 00 F7 CC,Direct,Direct,6,-1,F0 00 F7</p> <p>SysEx Formula: <input type="text"/> Edit</p>
--	---

<p>Controler Number 99 entspricht NRPN high</p> <p>Controler Value 58 = 0x3a entspricht EFX1</p>	<p>Add MIDI message</p> <p>Add a new message to the multi list [-1 for parent value setting, -2 for parent number setting. For SysEx message the formula is the same as in the SysEx editor.]</p> <p>Midi message type: CC</p> <p>Value mapping: Direct</p> <p>Number mapping: Direct</p> <p>SysEx Formula: F0 00 F7</p> <p>Controller Number: 99</p> <p>Controller Value: 58</p> <p>OK</p>
--	---

<p>Controler Number 98 entspricht NRPN low</p> <p>Controler Value 33 = 0x21 entspricht Input gain</p>	
<p>Controler Number 6 entspricht Data entry</p> <p>Controler Value -1 Platzhalter für Modulator Wert</p>	

Midi Ausgabe:

<p>Ch:[1] No:[99] Val:[58] RAW: [b0 63 3a] – High (0x3a = EFX1) Ch:[1] No:[98] Val:[33] RAW: [b0 62 21] – Low (0x21 = Input gain) Ch:[1] No:[6] Val:[127] RAW: [b0 06 09] – Data (0x09)</p>
<p>Ch:[4] No:[99] Val:[58] RAW:[b3 63 3a] Ch:[4] No:[98] Val:[33] RAW:[b3 62 21] Ch:[4] No:[6] Val:[127] RAW:[b3 06 7f]</p>

6.7 NRPN Kurzzusammenfassung

Die in diesem Abschnitt aufgelisteten Werte dienen einer Übersicht der hauptsächlich vorkommenden und eingesetzten Parameter bei NRPN MIDI Nachrichten.

NRPN high (CTRL 99) definiert zu welchem Effektkanal die NRPN Nachricht gerichtet ist

NRPN low (CTRL 98) definiert welches Register des vorher definierten Effektkanals angesprochen wird

NRPN Data entry (CTRL 06) enthält den Wert der in das Register geschrieben wird

Controller Number:

99 = NRPN high

98 = NRPN low

06 = Data entry

Controller Value:

58 = 0x3a = EFX1

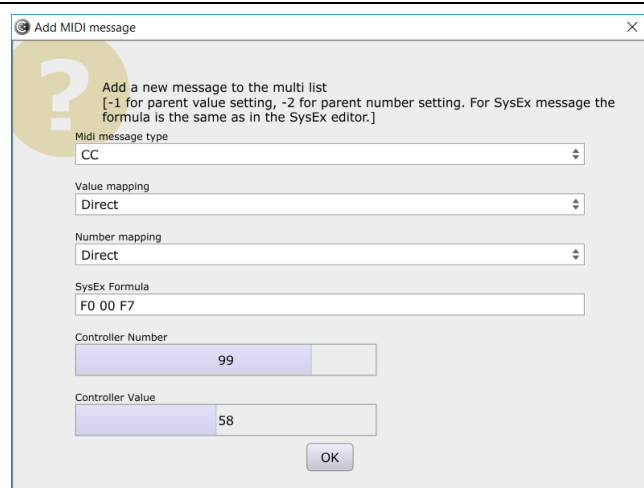
59 = 0x3b = EFX2

60 = 0x3c = EFX1&2

61 = 0x3d = HQ-Distortion

-1 = -> Modulator Value

für weitere siehe SD1000 Handbuch -
MIDI Implementation



6.8 NRPN Message Combo (Preset)

Im folgenden Abschnitt wird das eigen-entworfene Konzept beschrieben, dass für alle EFX Preset-Modulatoren im Panel gilt. Im Kern besteht das Model aus drei Modulatoren:

- Loading-Modulator: Combo Modulator aus Abbildung 18
- Storing-Modulator: Abbildung 20
- Showing-Modulator: Label mit grüner Schriftfarbe aus Abbildung 18

Der Loading-Modulator erlaubt das Laden von Presets in der Benutzeroberfläche ohne/und mit MIDI-Nachrichten. Zudem ist es möglich das Preset wiederholt auszuwählen.

Durch den Storing-Modulator ist es möglich beim Laden eines gespeicherten Panelzustands nur den Preset MIDI Befehl zu senden, ohne die ebenfalls gespeicherten Parameter des Presets zu überschreiben und ignorieren.

Mit dem Showing-Modulator wird dem Nutzer signalisiert welches Preset aktuell geladen und als Basis der Konfiguration dient.

Beispiel

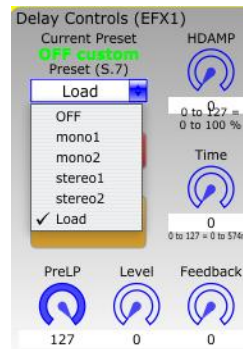


Abbildung 18: Modulatoren Gruppe „Delay Controls“ des EFX1

Insert MFX Delay Controls		
3xh	58h	Delay Preset, 0..4 (off, mono1, mono2, stereo1, stereo2)

Insert MFX Delay Presets

Preset is selected by using MIDI NRPN 3x58h, with value = preset number, from 0 to 4

Nb	Name	Preset Default values			
		Delay Mode	Level	Delay Time	Feedback
0	Off	0	0	0	0
1	Delay 1	0	64	20	8
2	Delay 2	0	64	35	20
3	Pan Delay 1	1	64	50	16
4	Pan Delay 2	1	64	70	32

Note:
Preset value for HDamp is always 0, Pre-Low-Pass-Filter is always 127.

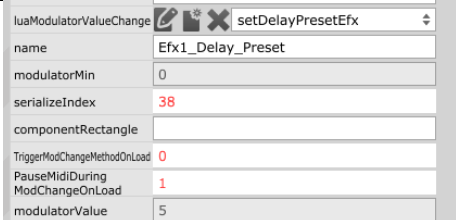
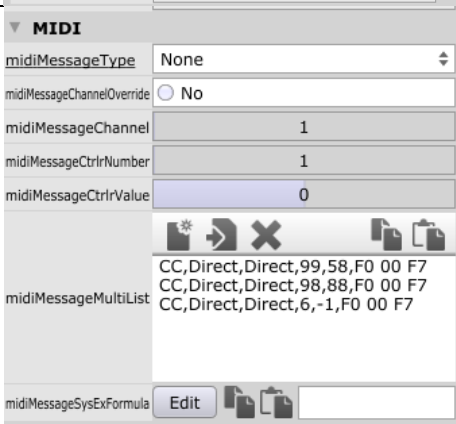
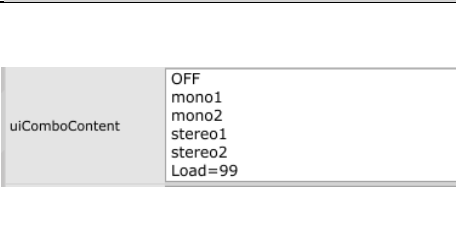
<p>luaModulatorValueChange: Methode die ausgeführt wird sobald der Modulator Wert sich verändert</p>	
<p>midiMessageType: None Die midiMessageMultiList kann ignoriert werden da der midiMessageType auf None steht. Der Inhalt ist überbleisel früherer Panelsversionen als der Modulator die MIDI Nachricht nicht über Scripts versendet hat.</p>	
<p>uiComboContent OFF mono1 mono2 stereo1 stereo2 Load=99</p>	

Abbildung 19: Eigenschaften und Attribute des Loading-Modulators am Beispiel Delay Preset EFX1

Der Loading-Combo-Modulator selbst hat keine MIDI-Einstellungen. Die MIDI Message wird innerhalb des „luaModulatorValueChange“ Scripts generiert und abgeschickt. Damit ist es möglich den „Load“-Eintrag nach erfolgreicher Preset Auswahl wieder zu setzen – und das selbe Preset erneut zur Auswahl zu stellen. Ein Standard Combo-Modulator erkennt eine erneute Auswahl des bereits ausgewählten Eintrags nicht als Änderung an und sendet keine MIDI Message.

Zusätzlich zu diesem Loading-Combo-Modulator mit Script existiert ein Standard Combo-Modulator, im Model Storing-Modulator genannt und im Normal Modus des Panels unsichtbar, der die MIDI Property Parameter besitzt. Dieser wird im Falle des Speicherns und Ladens verwendet um das Preset auf dem die Werte des Effektblocks basieren zu erhalten. Wichtig ist hierbei, dass das Preset vor dem eigentlichen Parameter geladen und gesendet wird. Dafür werden bei der Initialisierung der Modulator-Property „serializeIndex“ zu Beginn alle EFX-Modulatoren deren Name mit „Preset_Value“ enden initialisiert.

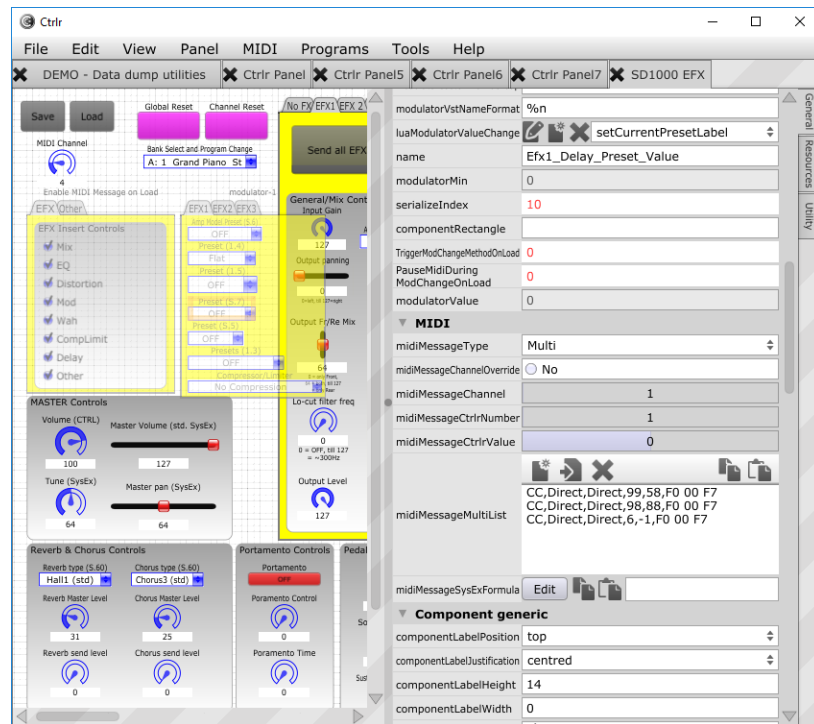


Abbildung 20: Storing-Modulator „Efx1_Delay_Preset_Value“ – Speichern & Laden des Presets

6.8.1 lua-Script: setDelayPresetEfx

```
setDelayPresetEfx = function(--[[ CtrlrModulator --]] mod, --[[ number --]] value, --[[ number --]] source)

    methodName = "setDelayPresetEfx"
    if (loadingStage ~= nil) then
        if (loadingStage == getStateValueIgnoredAutomatedOnModChangeEvents()) then
            console (string.format("----->%s is in state: ignored automated onModChange Event == %s - stop
Method", methodName, tostring(loadingCompleteFlag)))
            return
        end
    end

    modName = mod:getProperty("name")

    efxNum = 0
    if (modName == "Efx1_Delay_Preset") then
        efxNum = 1
    elseif (modName == "Efx2_Delay_Preset") then
        efxNum = 2
    elseif (modName == "Efx3_Delay_Preset") then
        efxNum = 3
    else
        console (string.format("setDelayPresetEfx () executed from mod=%s, should not happen!", modName))
        return
    end

    valueMapped = mod:getValueMapped()
    validPreset = isPresetValid(mod, value)

    if validPreset == true then
        setDelayPresetEfx_ (efxNum, valueMapped, true)
        -- <--- individuell
        panel:setPropertyString("panelMidiPauseOut", "1")
        -- not needed ? Component MIDI Implementation set to NONE
        mod:setModulatorValue(mod:getMaxNonMapped(), false, false, false)
        -- careful: endless loop
        LAST ITEM NEEDS MAPPED VALUE=99
        panel:setPropertyString("panelMidiPauseOut", "0")
    end
end
```

6.8.2 lua-Script: setDelayPresetEfx_

```
setDelayPresetEfx_ = function (efxNum, valueMapped, midiOutputEnabled)

    console (string.format("setDelayPresetEfx_ (efxNum=%d, valueMapped=%d, midiOutputEnabled=%s)", efxNum, valueMapped,
tostring(midiOutputEnabled)))

    if(efxNum < 1 and efxNum > 3) then
        console ("non valid efxNum, break setDelayPresetEfx_")
        return
    end
end
```

```

if (midiOutputEnabled == true) then
    -- Built Message CC/SysEx
    nrpnHigh = 0x3a + (efxNum-1)    -- EFX Modul 0x3a, 0x3b, 0x3c
    nrpnLow = 0x58                  -- Insert MFX Delay Presets
    data = valueMapped              -- int value

    -- Send Message
    sendNRPNMessage( nrpnHigh, nrpnLow, data )
end

-- disable Midi Messages

-- Update UI (Slider = Parameter values)
setDelayPresetEfx_Parameters(efxNum, valueMapped, false)
    -- <--- individuell

-- Set Combobox to Custom

-- enable Midi Messages
end

```

6.8.3 lua-Script: setDelayPresetEfx_Parameters

```

setDelayPresetEfx_Parameters = function (efxNum, valueMapped, midiOutputEnabled)

    console (string.format("setDelayPresetEfx_Parameters (efxNum=%d, valueMapped=%d,
midiOutputEnabled=%s)", tonumber(efxNum,10), tonumber(valueMapped,10), tostring(midiOutputEnabled)))

    panel:setPropertyString("panelMidiPauseOut","1")

    --preset = panel : getModulatorByName("Efx%d_Delay_Preset"):getModulatorValue()
    preset = valueMapped

    PresetValueModName = string.format("Efx%d_Delay_Preset_Value", efxNum)
    PresetValueMod = panel : getModulatorByName(PresetValueModName)
    PresetValueMod : setModulatorValue (preset, false, false, false)

    onOffButtonModName = string.format("Efx%d_Delay_On",efxNum)
    modeModName = string.format("Efx%d_Delay_Mode",efxNum)
    levelModName = string.format("Efx%d_Delay_Level",efxNum)
    delayTimeModName = string.format("Efx%d_Delay_Time",efxNum)
    feedbackModName = string.format("Efx%d_Delay_Feedback",efxNum)
    hdampModName = string.format("Efx%d_Delay_Hdamp",efxNum)
    preLowPassModName = string.format("Efx%d_Delay_PreLp",efxNum)

    mode = panel : getModulatorByName(modeModName)
    level = panel : getModulatorByName(levelModName)
    delayTime = panel : getModulatorByName(delayTimeModName)
    feedback = panel : getModulatorByName(feedbackModName)
    hdamp = panel : getModulatorByName(hdampModName)
    preLowPass = panel : getModulatorByName(preLowPassModName)

    onOffButton = panel : getModulatorByName(onOffButtonModName)

    -- Values for all Presets
    hdamp : setModulatorValue((0),false,false, false)
    preLowPass : setModulatorValue((127),false,false, false)

    -- update Label to current Preset Name
    PresetLabelModName = string.format("Efx%d_Delay_Current_Preset", efxNum)
    PresetListModName = string.format("Efx%d_Delay_Preset", efxNum)
    setEfxPresetLabel(PresetLabelModName, PresetListModName, valueMapped)

    if(preset == 0) then
        onOffButton:setModulatorValue((0), false, false, false)
    else
        onOffButton:setModulatorValue((1), false, false, false)
    end

    if(preset==0)then
        mode : setModulatorValue((0),false,false, false)
        level : setModulatorValue((0),false,false, false)
        delayTime : setModulatorValue((0),false,false, false)
        feedback : setModulatorValue((0),false,false, false)

    elseif(preset==1)then
        mode : setModulatorValue((0),false,false, false)
        level : setModulatorValue((64),false,false, false)

```

```
        delayTime : setModulatorValue((20),false,false, false)
        feedback : setModulatorValue((8),false,false, false)

elseif(preset==2)then
    mode : setModulatorValue((0),false,false, false)
    level : setModulatorValue((64),false,false, false)
    delayTime : setModulatorValue((35),false,false, false)
    feedback : setModulatorValue((20),false,false, false)

elseif(preset==3)then
    mode : setModulatorValue((1),false,false, false)
    level : setModulatorValue((64),false,false, false)
    delayTime : setModulatorValue((50),false,false, false)
    feedback : setModulatorValue((16),false,false, false)

elseif(preset==4)then
    mode : setModulatorValue((1),false,false, false)
    level : setModulatorValue((64),false,false, false)
    delayTime : setModulatorValue((70),false,false, false)
    feedback : setModulatorValue((32),false,false, false)

end

panel:setPropertyString("panelMidiPauseOut","0")

end
```

7 Fehlende Informationen

Wie Anfangs erwähnt werden im Benutzerhandbuch des Ketron SD1000 essentielle Befehle und Informationen nicht aufgeführt. Diese werden im folgenden Abschnitt aufgelistet und erklärt.

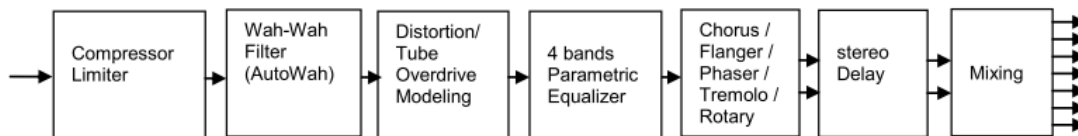
7.1 Insert FX On

Der SD1000 enthält zwei Effektkanäle. EFX1 und EFX2. Diese können jeweils separat oder parallel betrieben werden. Die SD1000 Dokumentation beschreibt wie die Effektkanäle eingestellt werden können, jedoch fehlen jegliche Angaben wie die zwei Effektkanäle aktiviert werden. Da im SD1000 ein SAM DSP der Firma Dream arbeitet konnte in dessen Nachfolger Firmware Handbuch die fehlende Information abgerufen werden.

Insert Effects NRPN Controls

Using an Insert Effect means that a single MIDI part (or also several) can be switched to "Insert FX ON". In this case these MIDI parts are going through the Insert Effect block first (e.g. Guitar Multi-Effects) and after into mixing (Front L/R, Rear L/R, Reverb, Chorus and Delay send).

This is Insert Multi-Effect configuration, all effects can be used at same time:



Be aware that also single effects or combinations of only some of these effects are possible by switching OFF the other effects in the signal processing path.

Sysex message to switch Insert Effect ON: F0H 41H 00H 42H 12H 40H 4pH 22H nn xx F7H (xx = don't care)
with 'p'=MIDI track, 'nn': 0 = track in normal mode, 1 = send to MFX1, 2 = send to MFX2, 3 = send to both MFX (stereo)

NRPN messages for Insert MFX1 must be sent with NRPN High byte = 0x3A.

NRPN messages for Insert MFX2 must be sent with NRPN High byte = 0x3B.

NRPN messages for Insert MFX1&MFX2 (stereo) must be sent with NRPN High byte = 0x3C.

Zum selektieren bzw. aktivieren des gewünschten Effektkanals ist die Sysex Midi Nachricht „Insert Effect ON“ verantwortlich: F0 41 00 42 12 40 4p 22 nn xx F7

7.2 Insert MFX Chorus/Flanger/Phaser/Tremolo/Rotary Controls

Auf Grund von fehlenden Presetwerten deaktivierte Controls:

MFX Chorus/Flanger/Phaser/Tremolo/Rotary Control
Tremolo modulation shape
Rotary fast modulation rate
Rotary acceleration time
Rotary deceleration time

7.3 High-Quality Distortion Preset Parameter

Der SD1000 ermöglicht auf dem ersten Effektkanal einen zusätzlichen Verzerrungseffekt, auch HQ Distortion genannt, zu aktivieren. In der Dokumentation ist der Hinweis auf ein externes Dokument das die Preset Parameter der HQ Distortion beschreibt hinterlegt. Dieses angemerkte Dokument ist jedoch nicht abrufbar und eine Internetrecherche ergab keine verwertbaren Informationen.

Praktikumsbetreuer Prof. Dipl.-Ing. Hans-Werner Neuschwander hat nach einer Anfrage bei der Firma Dream einige dieser Preset Parameter erhalten. Diese sind in die Oberfläche eingebunden um dem Benutzer den Überblick über die getätigten Einstellungen zu gewährleisten. Da allerdings einige Parameter nicht beschrieben wurden ist diese Implementierung nicht 100% konsistent/stabil. Ein speichern und laden kann zu einem anderen Setting führen.

Auf Grund von fehlend dokumentierten Presetwerten deaktivierte Controls:

HQ-Distortion Control
Noise Reduction
Booster
Drive
Tone
Level
Input High Pass Cutoff
Input High Pass Mode
Input Low Pass Cutoff
Input Low Pass Mode

Auf Grund von nicht dokumentierten Preset Parameter entfernte Presets:

Metal

7.4 Aktivierung nicht vollständig dokumentierter Controls

Zum Aktivieren dieser Controls im Script „EnableSingleEfxMidiMessageForLoading“ Zeile 109 (Abbildung 21) enableUndocumentedControl = true setzen, Script speichern und kompilieren.

```
107 handleUndocumentedControls = function (enabledEfxId)
108
109     enableUndocumentedControl = false    --- <--- nicht dokumentierte controls aktivieren/deaktivieren
110
111     setupUndocumentedControls (enabledEfxId, enableUndocumentedControl)
112
113
114 end
```

Abbildung 21: EnableSingleEfxMidiMessageForLoading - Anpassung

8 Speicher und Lade Realisierung

Zur Implementierung der Speicher- und Ladefunktion musste das Ausgangspanel umstrukturiert werden.

Um den Speicher und Ladevorgang erklären zu können muss erst einmal die Frage geklärt werden was gespeichert werden muss. Ziel ist es die mit dem Panel getätigten Einstellungen auf der Festplatte abzulegen und wiederherstellen zu können. Beim Wiederherstellen der Einstellungen muss also jeder Modulator im Panel seinen Wert über MIDI mitteilen. Wie in Kapitel Modulator Attribute beschrieben entspricht dieser Wert dem „modulatorValue“ Attribut. Dieses Attribut muss von allen Modulatoren abgefragt und gespeichert werden.

Beim Laden eines Preset Combo-Modulators kann Ctrlr nicht mitgeteilt werden, dass die Methode „OnModulatorValueChanged“ nicht ausgeführt werden soll. Beim Ausführen dieser Methode, dass Zeitversetzt über Events geschieht, werden alle zugehörigen Modulatoren überschrieben.

Aus diesem Grund wurde das in Kapitel: NRPN Message Combo (Preset) vorgestellte drei Modulatoren Konzept (Loading-, Storing- und Showing) entwickelt.

In Ctrlr ist es nicht ohne erheblichen Aufwand möglich den Wert eines Modulators zu ändern, ohne dass dieser die Funktion OnModulatorValueChanged aufruft. Dies liegt in der nicht vorhandenen Berücksichtigung der Flag ui in der setModulatorValue Methode. Durch diesen Umstand muss ein wenig getrickst werden um beim Laden der gespeicherten Modulatoren die OnModulatorValueChanged verwenden zu können ohne dass diese die Werte anderer Modulatoren ungewollt überschreiben.

8.1 Vorkonfiguration

Der Speicher und Ladevorgang benötigt gewisse Voraussetzungen um zu Funktionieren. Diese Voraussetzungen werden durch die Vorkonfiguration geschaffen und in diesem Abschnitt beschrieben.

Jeder Modulator der MIDI Konfigurationsinformationen enthält erhält ein neues Attribut „serializeIndex“. Dieses Bestimmt einerseits eine Wertzuordnung, welcher Wert welchem Modulator zugewiesen wird und andererseits bestimmt das Attribut die Reihenfolge in der die Modulatoren gesetzt und die MIDI Nachricht an das SD1000 gesendet werden.

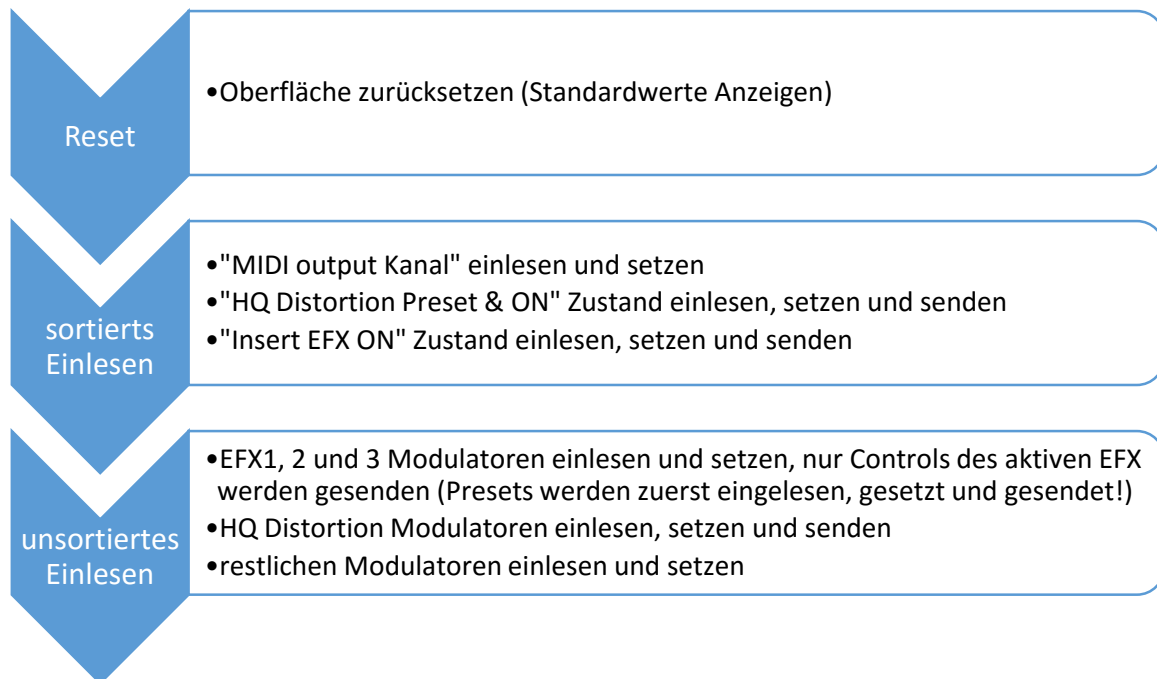
Diese Initialisierung geschieht über die Methode `InitSerializeProperty()` . Zusätzlich zum `serializeIndex` werden die Attribute „vstIndex“ und „pauseMidiMessageDuringModValChangeOnLoad“ gesetzt. Letztere ermöglicht ein differenziertes Ladeverhalten um bei gewissen Modulatoren beim Laden keine Midi Ausgabe zu senden und den Ladevorgang zu debuggen.

8.2 Speichervorgang

Beim Speichern werden alle Modulatoren mit „serializeIndex“ in dessen Nummerierten Reihenfolge abgefragt, in einem Array abgelegt und anschließend auf der Festplatte gespeichert. Pro Modulator-Wert werden 2 Bytes gespeichert, da der „programList“ Modulator mehr als 128 Möglichkeiten besitzt.

8.3 Ladevorgang

Die Reihenfolge des Ablaufs ist entscheidend. Da einige MIDI Befehle vom MIDI Kanal abhängen muss dieser zuerst geladen und gesetzt werden. Zusätzlich hat sich in einem Test gezeigt, dass ein senden der MIDI Befehle „HQ-Distortion Preset“ und „HQ-Distortion ON“ vor dem eigentlich aktivieren des Effektkanals (Insert EFX ON) zum konsistenteren Ladeergebnis führt. Werden die MIDI Befehle des HQ-Distortion nicht vorher gesendet, kann es vorkommen, dass dieser entgegen den geladenen Einstellungen nicht aktiv ist und erst nach einem EFX-Kanalwechsel (EFX1 -> EFX X -> EFX1) aktiv wird.



8.4 Realisierungsergebnis

Der Ladevorgang ist nicht vollständig getestet, da einerseits zu viele unterschiedliche Möglichkeiten bestehen und das SD1000 Soundmodul nicht vollständig deterministisch Dokumentiert ist.

9 LUA-Scripts

Das in Ctrlr entwickelte Panel verfügt über eine Vielzahl von programmierten und automatisierten Abläufen. Diese sind mittels der von Ctrlr unterstützten Script Sprache „Lua“ realisiert. Die Scripts ermöglichen den Zugriff auf Modulatoren, deren Komponente und weiteren Eigenschaften.

9.1 EFX Preset Scripts

Da der SD1000 keinerlei Rückmeldung über die aktuellen Einstellungen liefert muss die Benutzeroberfläche alle Einstellungen selbst pflegen.

Mit Hilfe der folgenden Scripts erkennt Ctrlr in welchem Effekt Kanal (1,2 oder 1&2) Presets geladen werden sollen, und verändert (aktualisiert) die dazugehörigen Parameter auf der Benutzeroberfläche um wieder konsistent zu sein.

Script	Preset	Handbuch
setAmpPresetEfx	Insert MFX AMP-Model Presets	Firm5716-EK Seite 6
setEQPresetEfx	Insert MFX Parametric Equalizer Presets	SD1000_web Seite 55
setModPresetEfx	Insert MFX Chorus/Flanger/Phaser/Tremolo/Rotary Presets	SD1000_web Seite 56
setCompPresetEfx	Insert MFX Compressor/Limiter Presets	SD1000_web Seite 54
setWahPresetEfx	Insert MFX Wah-Wah Presets	Firm5716-EK Seite 5
setDistPresetEfx	Insert MFX Distortion Presets	SD1000_web Seite 55

Script	Beschreibung
isPresetValid	kontrolliert ob Auswahl des Preset Combo Modulators valide ist Preset Combo Modulator muss mindestens zwei Element beinhalten, letztes Element heißt „Load“ und wird nach der Auswahl eines validen Preset automatisch wieder angewählt
setEfxPresetLabel	setzt Preset Label entsprechend der mappedValue des Combo Modulators

9.2 Programm Selektion

In der Vorgänger Version des Panels war die Wahl des Soundprogramms unzureichend gelöst. Um zu erkennen welches Programm ausgewählt ist musste die SD1000 Soundliste aus dem Handbuch herangezogen werden.

In der aktuellen Panel Version sind alle Verfügbaren Sounds mit Namen in einem Combo-Modulator aufgelistet (Abbildung 22). Sortiert ist die Liste nach den vier Voice Banks aus dem SD1000 Handbuch. Nach Selektion eines Programms im Combo-Modulator „programList“ berechnet das Script calculateProgramChange aus der Programmauswahl die MIDI Befehle: „Switch Voice Bank“ und „Program Change“ und sendet sie an den SD1000. Dies erleichtert die Programmauswahl erheblich.

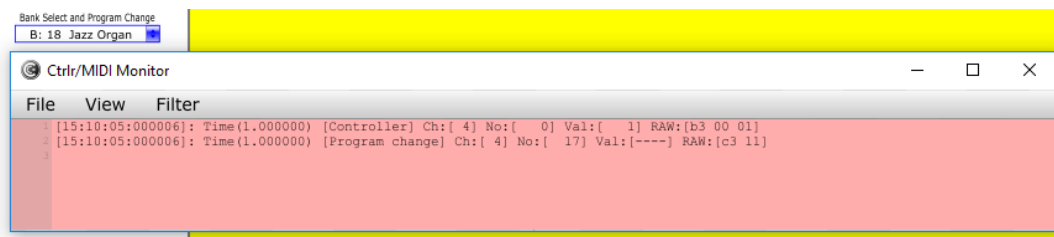


Abbildung 22: MIDI Monitor – programList-Modulator

9.3 Mega Panic Reset (Global Reset)

Der Mega Panic Reset setzt alle Einstellungen des SD1000 komplett auf allen 16 MIDI Kanälen zurück und stoppt jede Wiedergabe (Abbildung 23: MIDI Monitor - Global Reset).

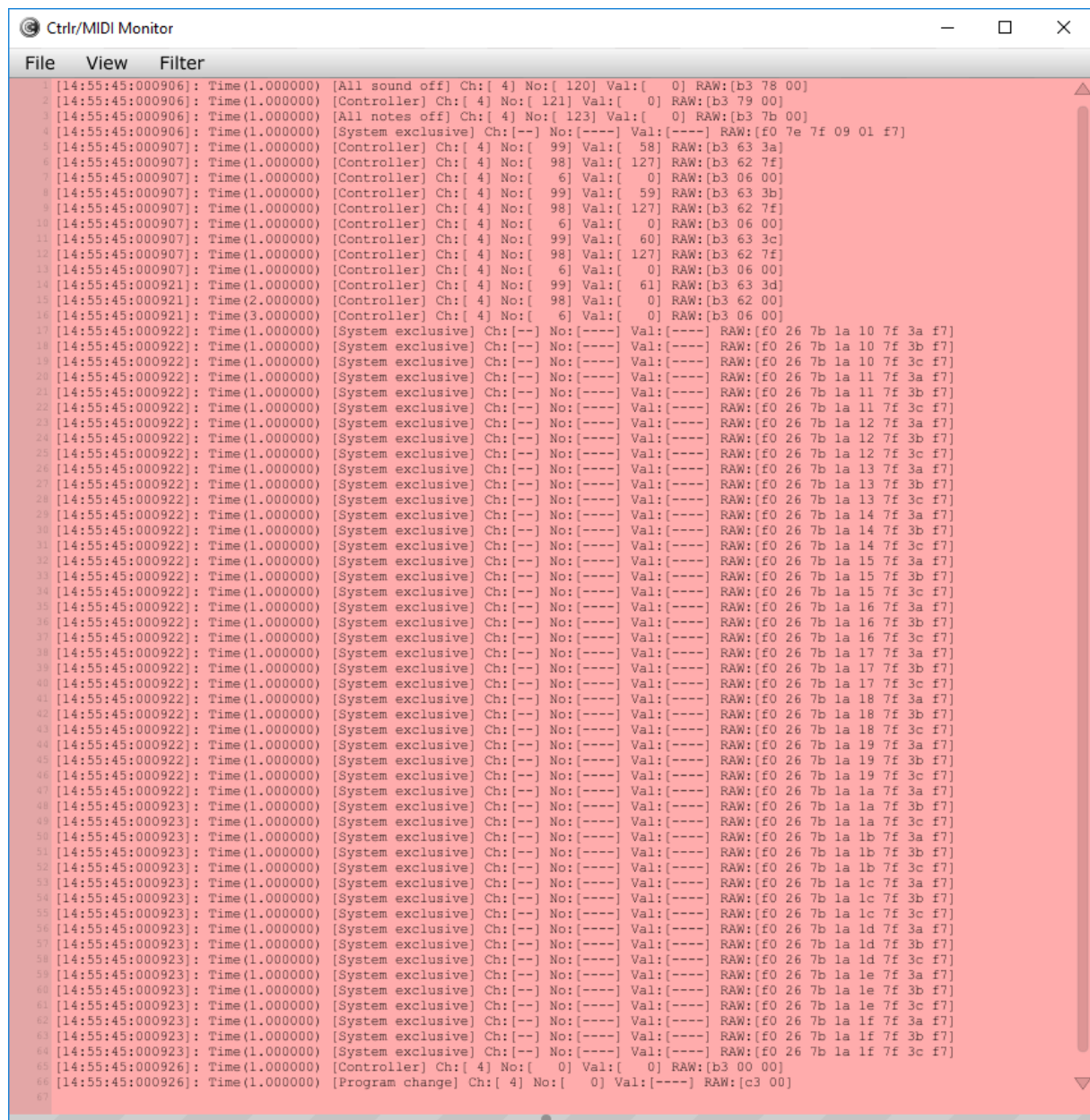


Abbildung 23: MIDI Monitor - Global Reset

Folgende MIDI Befehle werden dabei gesendet:

Quelle	MIDI Implementierung	Channel	EFX
SD1000 S. 59	all sound off	panelMidiOutputChannel	-
SD1000 S. 59	reset all controllers	panelMidiOutputChannel	-
SD1000 S. 59	all notes off	panelMidiOutputChannel	-
SD1000 S. 59	general midi reset	no Channel	-
Firm5716-EK S. 4	Insert MFX Reset All	panelMidiOutputChannel	3a, 3b, 3c
SD1000 S. 60	efx preset = OFF	1 bis 16	3a, 3b, 3c

Zusätzlich wird um in der Oberfläche konsistent zu bleiben alle Default-Werte angezeigt.

9.4 Channel Reset

Der Channel Reset setzt stoppt jede Wiedergabe und setzt den SD1000 auf dem selektierten MIDI Kanal zurück (Abbildung 24). Die Oberfläche wird ebenfalls auf die Default-Werte zurückgesetzt.

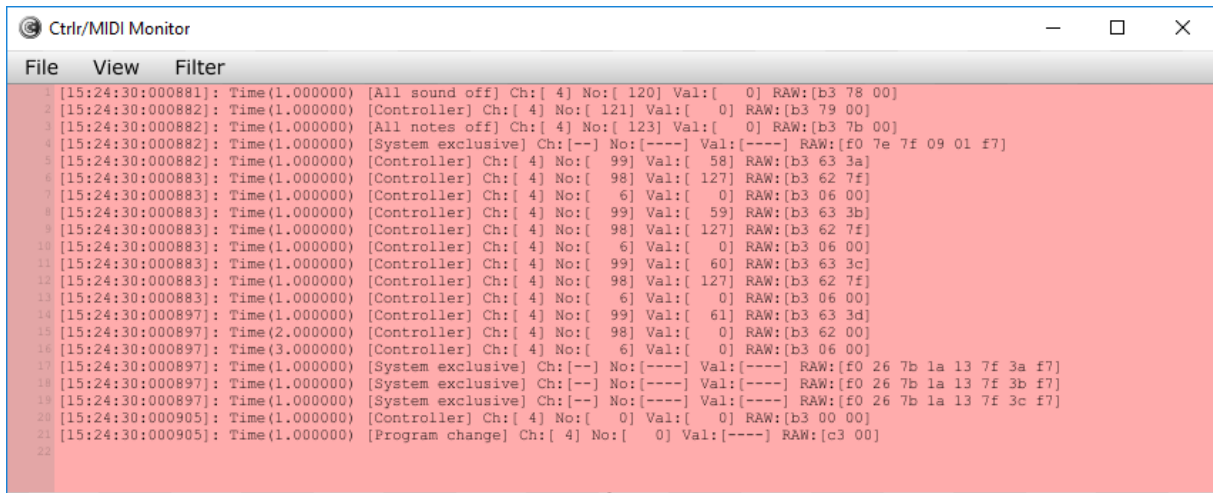


Abbildung 24: MIDI Monitor - Channel Reset

Folgende MIDI Befehle werden dabei gesendet:

Quelle	MIDI Implementierung	Channel	EFX
SD1000 S. 59	all sound off	panelMidiOutputChannel	-
SD1000 S. 59	reset all controllers	panelMidiOutputChannel	-
SD1000 S. 59	all notes off	panelMidiOutputChannel	-
SD1000 S. 59	general midi reset	no Channel	-
Firm5716-EK S. 4	Insert MFX Reset All	panelMidiOutputChannel	3a, 3b, 3c
SD1000 S. 60	efx preset = OFF	panelMidiOutputChannel	3a, 3b, 3c

9.5 Speichern und Laden

Da der Speicher und Ladevorgang eine vor Initialisierung benötigt sind die Scripts in drei Abschnitte unterteilt.

9.5.1 Vorkonfiguration

Mit der Methode „listAllModulatorsSorted ()“ werden alle Modulatoren in Arrays gespeichert. In welchem Array der Modulator landet hängt von seinem Namen ab.

Modulator	Array; mit i = 1,2,3
wenn uiType == uiLabel und Label zu einer Efx Gruppe gehört	string.format("allEfx%dDesignElements", i)
wenn Efx Modulator - name:startsWith(string.format("Efx%d", i))	string.format("allEfx%dDesignElements", i)
wenn Efx Modulator – und Efx Preset Store Modulator name:endsWith ("Preset_Value")	string.format("efx%dPresetValueModulators", i)
wenn Efx Modulator – und Efx Preset Loader Modulator name:endsWith ("Preset_Value")	-
wenn Efx Modulator – und kein Efx Preset Loader/Store Modulator	efxModulators string.format("efx%dModulatorsWithoutPresetValues", i) saveInFxGroupRegister (mod, modName): string.format ("efx%d%sControls", i, groupName)
wenn HQ-Distortion Modulator und keine Preset Loader Modulator name:startsWith ("HQ_")	HQDistModulators
wenn Efx Group-Modulator name: startsWith(string.format("grp_%d_", i))	string.format("allEfx%dDesignElements", i)
wenn Development-Modulator name:startsWith ("dev_")	
Rest	ungroupedModulators

Dadurch ist es nun möglich auf die einzelnen Modulatoren Typen zuzugreifen und gezielt und sortiert zu Initialisieren.

Die sortierte Initialisierung des „serializeIndex“ Attributs erfolgt im Script „InitSerializeProperty“. Dabei werden alle Modulatoren im Panel in der Reihenfolge der folgenden Tabelle durchlaufen und mit einem serializeIndex ausgestattet.

Nach der Index Initialisierung wird dem Modulator „loadingCompleMod“ der höchste serializeIndex zugewiesen.

Modulator		serializeIndex (1-n)
midiChannel		1
HQ_Dist_Preset_Value		2
HQ_Dist_ON		3
efxModus		4
fxTabs		5
programList		6
Efx[1,2,3]_*_Preset_Value		ja, Priorität: Preset Store
Efx[1,2,3]_*_Preset		nein, (Performance)
Efx[1,2,3]_*_Current_Preset		nein, (Label)
Efx[1,2,3]_*		ja, keine Priorität
HQ_*_Preset		nein, (Performance)
HQ_*_Current_Preset		nein, (Label)

MA_Reverb_*		ja, keine Priorität
MA_Chrous_*		ja, keine Priorität
MA_Portamento_*		ja, keine Priorität
MA_Pedal_*		ja, keine Priorität
MA_*		ja, keine Priorität
loadingCompleteMod		n

9.5.2 Speichern

Zum Speichern wird die Methode „saveAllModVal“ ausgeführt. Sie fragt zu Beginn nach einem Speicherort und Dateinamen, erzeugt die Datei und sammelt anschließend, über die Ctrlr Panel Methode „getModulatorValuesAsData“, das Modulator Attribut modulatorValue aller Modulator in einem Array – sortiert nach dem Attribut „serializeIndex“. Das Array mit den modulatorValue-Attributen wird in die Datei geschrieben und ist damit abgespeichert.

```
bytesPerValue = 2 -- 2 Bytes pro modulatorValue
dataToWrite = panel:getModulatorValuesAsData("serializeIndex", CtrlrPanel.EncodeNormal,
bytesPerValue, false)
if fileToWrite:replaceWithData (dataToWrite) == false then
    utils.warnWindow ("File write", "Failed to write data to file: "..fileToWrite:getFullPathName())
end
```

9.5.3 Laden

Der Ladevorgang (Methode „loadAllModVal“) fordert den Nutzer über das Dateiöffnen-Dialog Fenster auf eine Datei zum Laden auszuwählen. Anschließend wird der Inhalt der Datei in 2-Byte Schritten eingelesen und auf die Modulatoren beginnend bei serializeIndex = 1 angewendet.

Ist das MIDI Attribut „midiMessageType“ des Modulators auf None gesetzt wird für den Modulator keine MIDI Nachricht gesendet. Ist das Attribut auf etwas Anderes als None gesetzt sendet der Modulator die MIDI Nachricht und stellt die Einstellung des Speicherpunkts wieder her. Um mehr Kontrolle über das senden der MIDI Nachrichten zu erhalten wird vor dem setzen des modulatorValue die Enable MIDI Message On Loading Controls abgefragt. Diese pausieren gegeben falls die MIDI Ausgabe des Panels und verhindern damit das Senden der aktualisierten Modulator MIDI Nachricht.

Da der EFX-Tabulator kein Modulator Attribut modulatorValue besitzt und keine lua-Methode bei Änderung des Werts ausführt wird auf einen zweiten Modulator zurückgegriffen der diese Fähigkeiten besitzt. Dieser führt das lua-Script „setEfxModus“ aus und aktiviert bzw. deaktiviert die Controls der nicht angewählten EFX-Kanäle.

aktiver EFX-Tab	EFX1 Controls	EFX2 Controls	EFX3 Controls
EFX1	An („Multi“)	Aus („None“)	Aus („None“)
EFX2	Aus („None“)	An („Multi“)	Aus („None“)
EFX1&2	Aus („None“)	Aus („None“)	An („Multi“)

Beispiel: EFX1 ist gewählter EFX-Kanal

- alle EFX1 Controls aktiviert (MIDI Message Type = „Multi“)
- alle EFX2 Controls deaktiviert (MIDI Message Type = „None“)
- alle EFX1&2 Controls deaktiviert (MIDI Message Type = „None“)

Dies verhindert den in 10.3 angesprochenen Fall des Überschreibens von Einstellungen beim Laden.

10 SD 1000 Panel

In diesem Abschnitt geht es hauptsächlich um eine Benutzeranleitung zum in den vorherigen Abschnitten entwickelten Ctrlr Panel SD1000.



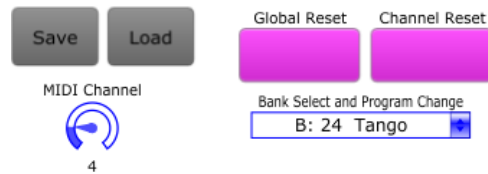
Ctrlr Panel SD100

Das Panel kann zu vier Hauptgruppen zusammengefasst werden. Diese werden im Folgenden kurz erläutert.

10.1 Panel Controls

Die Panel Controls setzen sich aus den folgenden Elementen zusammen:

- Speicher und Lade Funktion
- Global und Channel Reset
- MIDI Channel Wahl

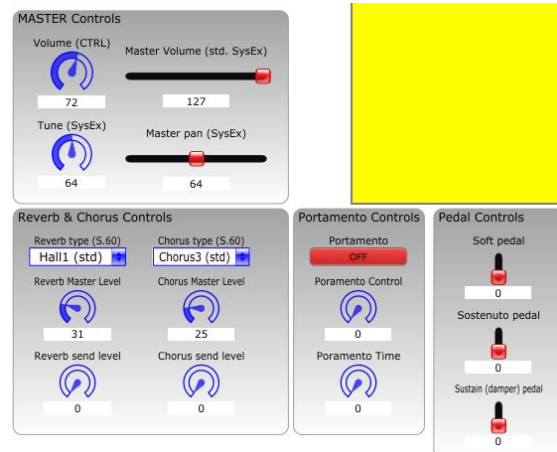


Channel Wahl, Program Selektion, Reset, Speichern und Laden Schalter

10.2 Master Controls

Der Master Bereich setzt sich aus den folgenden Gruppen zusammen:

- MASTER Controls
- Reverb & Chorus Controls
- Portamento Controls
- Pedal Controls



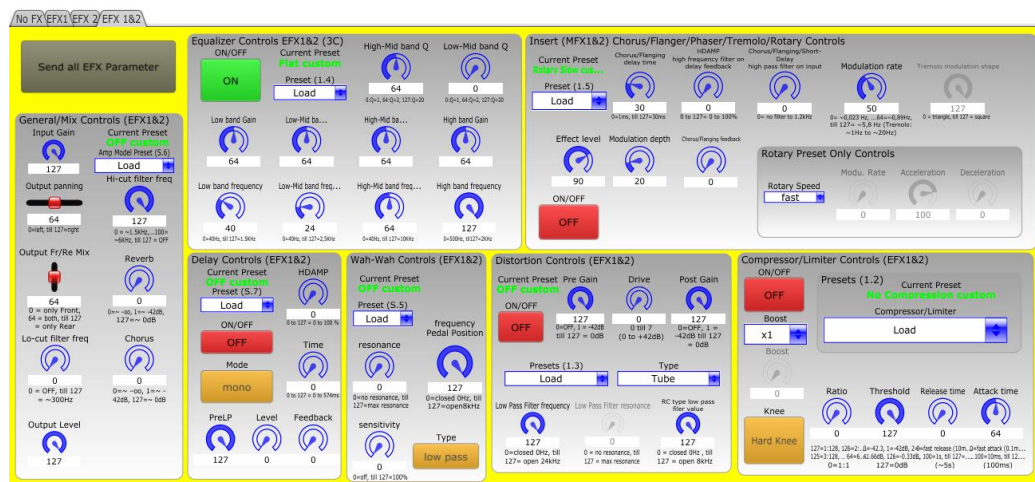
Master: Volume, Pan, Reverb, Chorus, Portamento und Pedal Controls

10.3 EFX Controls

Der EFX Bereich setzt sich aus folgenden Effektbausteinen zusammen:

- General/Mix Controls
- Compressor/Limiter Controls
- Wah-Wah Controls
- Distortion Controls
- Parametric Equalizer Controls
- Mod-FX Controls (Chorus/Flanger/Phaser/Tremolo/Rotary Controls)
- Delay Controls

Jeder Effektbaustein verfügt über eine Presetwahl, die die Voreinstellungen entsprechend dem Handbuch läd. Zusätzlich sind alle dokumentierten Parameter regelbar.



EFX mit allen Verfügbaren Effektblöcken „FX Inserts“

Über die oberen Tab-Reiter kann der Effekt-Kanal (No FX, EFX1, EFX2 oder EFX 1&2) gewählt werden. Beim Wechsel des Effekt Kanal sendet das Panel (über den nicht sichtbaren efxModus-Modulator) den, im SD1000 Handbuch nicht dokumentierten, MIDI Sysex Befehl „Insert FX On“ an den MIDI Controller.

Die zwei Effektkanäle EFX1 und EFX2 arbeiten unabhängig voneinander. Wie der Zusammenschluss EFX1&2 arbeitet konnte in dieser Arbeit nicht vollständig rekonstruiert werden. Fest steht, dass Einstellungen die in Kanal EFX1 oder EFX2 getätigt werden in EFX1&2 erhalten bleiben. Beim Speichern und Laden muss daher darauf geachtet werden, dass keine Werte des EFX1 oder EFX2 Kanals Werte des EFX1&2 Verbunds überschrieben und umgekehrt. Wechselt der Nutzer vom EFX1 in EFX1&2 ist die Oberfläche nicht mehr synchron mit den Einstellungen des SD1000. Der Nutzer erhält kein Feedback über diese Tatsache. Die Schaltfläche „Send all EFX Parameter“ gibt dem Nutzer jedoch die Chance alle Einstellungen des aktuellen EFX-Kanals erneut zu übertragen und die Oberfläche mit dem SD1000 zu synchronisieren und konsistent herzustellen.

Mögliche Verbesserung: beim Wechsel des EFX-Kanals wird automatisch die „Send all EFX Parameter“ Methode ausgeführt. Damit verliert man die Möglichkeit auf einem Kanal EFX1 und EFX2 getrennt einzustellen und über EFX1&2 im Verbund aktiv einzusetzen.

10.4 HQ-Distortion Controls

Auf EFX-Kanal EFX1 und EFX1&2 ist neben den bereits erwähnten Effektblöcken ein zusätzlicher Verzerrungseffekt konfigurierbar. Dieser Effektblock besteht wie die anderen Blöcke aus einer Presetwahl und deren Parameter. Da die Presets Parameter unzureichende Dokumentiert sind mussten die meisten Modulatoren deaktiviert werden.

Aktiviert man die nicht dokumentierten Modulatoren führt der Speicher – Ladevorgang zu unterschiedlichen Ausgangssituationen!



HQ-Distortion Controls: Kann im EFX1 und EFX 1&2 Modus bedient werden

10.5 Development und Debug Controls

In diesem Abschnitt wird auf die zur Entwicklung verwendeten und im Fehlerfall verwendbaren Elemente des Panels eingegangen. Diese sind, um den Nutzer nicht zu irritieren, im normalen Panel Modus nicht sichtbar. Mit der Tastenkombination STRG+E kann in den Panel Edit Modus gewechselt werden, in dem diese Bedienelemente sichtbar sind.

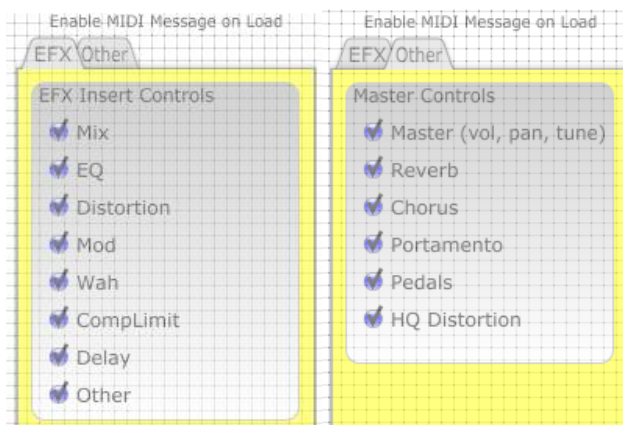
Bug: wenn Ctrlr im Edit Modus startet werden diese Elemente nicht angezeigt.
Lösung: nochmal in den Panel Edit Modus wechseln 2x STRG+E

10.5.1 Enable MIDI Message On Loading Controls

Mit diesen Checkboxes kann gewählt werden ob, eine bestimmte Modulatoren Gruppe beim Ladevorgang an den SD1000 gesendet werden soll. Bei nicht Markierung werden die Modulator Werte nach wie vor aus der Datei ausgelesen und in der Oberfläche angezeigt, jedoch nicht über MIDI gesendet.

Entstehen beim Speichern und Laden zwei unterschiedliche Ausgangssituationen ist es mit diesen Einstellungen möglich die Fehlerhaften Modulatoren zu finden.

Während der Entwicklung konnte so fehlerhafte Modulatoren entdeckt werden.



10.5.2 Preset Values zum Laden und Speichern

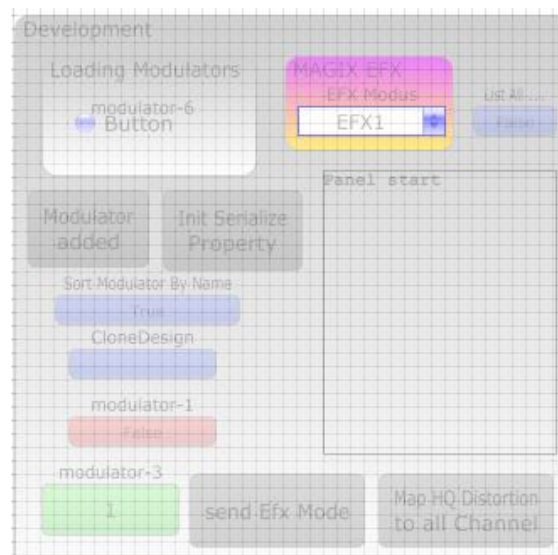
Wie in Abschnitt NRPN Message Combo (Preset) erläutert befinden sich in diesem Tab die Modulatoren die beim Speichern und Laden als Speicherplatz verwendet werden. Jede Combo führt nach Wertänderung (OnValueChangedMethod) das Script „setCurrentPresetLabel“ aus und gibt damit über ein Label den aktuellen Presetnamen aus. Die Namensfestlegung ist entscheidend, dass das Script für diese Combos alle funktioniert. Außerdem ist diese Namensgebung nötig um die Reihenfolge des serializeIndex Einzuhalten (Preset Values müssen vor den Parameter Values gesendet werden, dass keine Überschreibung stattfindet).

```
efxId = 1,2,3
group = Mix, EQ, Mod, Delay, Wah, Dist, Comp
string.format („Efx%d %s_Preset_Value“, efxId, group)
```



10.5.3 Development Controls

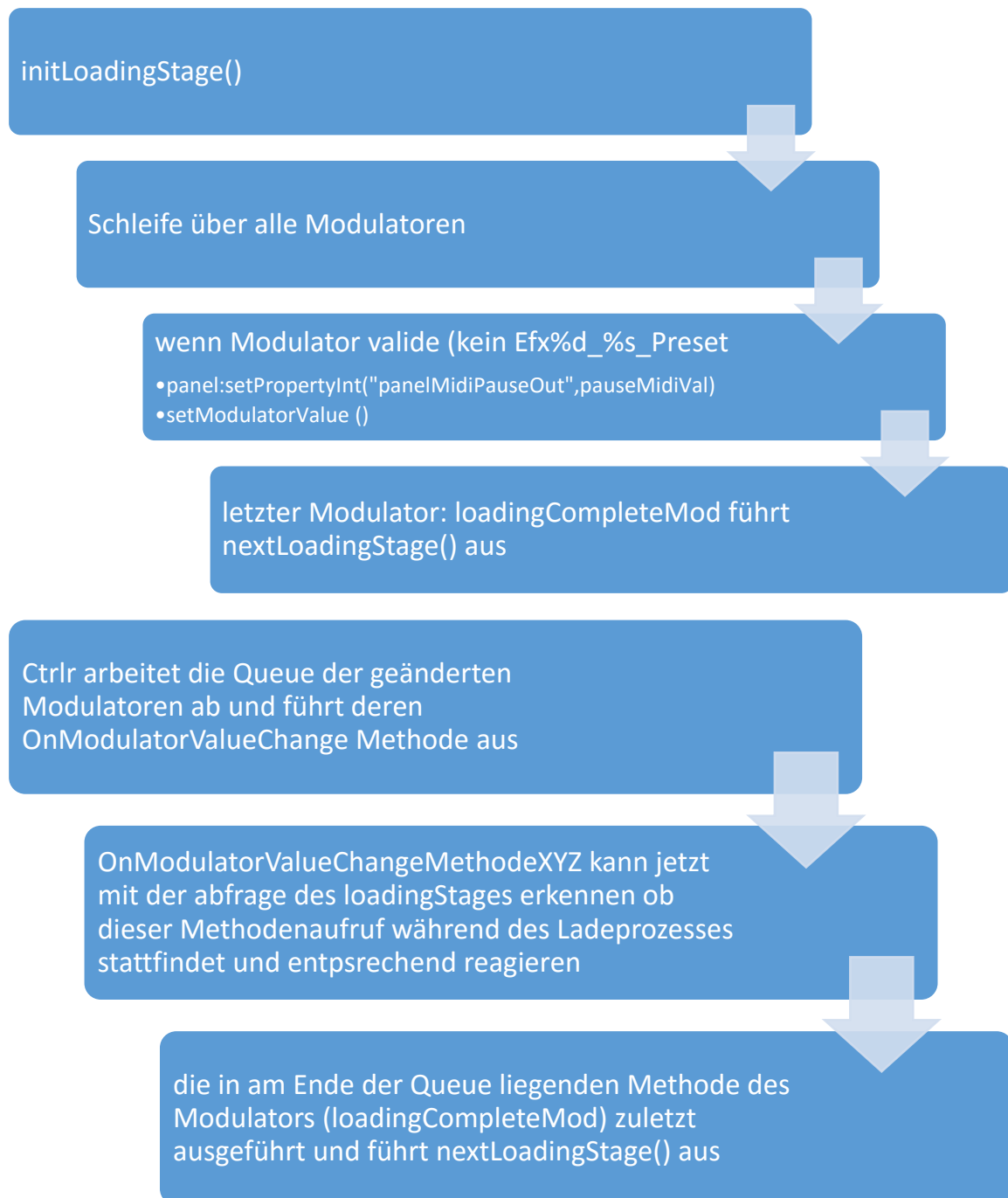
Die in dieser Gruppe zusammengefassten Elemente sollten bei der normalen Nutzung des Panels nicht bedient werden.



10.5.3.1 Loading Stages

Die Checkbox (modulator-6) in der Gruppe Loading Modulators ist nötig um die asynchrone Abwicklung von Modulator Wertänderungen und dem Ausführen der OnModulatorValueChange Methode regulieren zu können. Für den Fall das eine Wertänderung eine Methode aufruft die über lua-Code MIDI Nachrichten sendet muss der Fall abgefragt werden können ob diese Nachricht überhaupt gesendet werden darf.

Daher wird der Ladeprozess in mehrere Stadien eingeteilt.



Vorraussetzung für das Funktionieren dieser Lösung ist, dass der Modulator „loadingCompleteMod“ den höchsten serializeIndex besitzt.

Die `loadingStage` Abfrage war in früheren Panelversionen, als die Preset-Combos zum Laden und Speichern direkt verwendet wurden, zwingend notwendig. Aus Performance Gründen wurde jedoch zum NRPN Message Combo (Preset) Konzept gewechselt. Die `LoadingStage`-Implementierung ist weiterhin vorhanden und wird von einigen `OnValueChange`-Methoden abgefragt.

10.5.3.2 EFX Modus

Der EFX-Modus Modulator intern mit dem Namen `efxModus` hinterlegt speichert den zuletzt gewählten „Insert FX On“ und kümmert sich auch um die Übertragung der MIDI Nachricht. Dieser Workaround liegt an der fehlenden `Ctrlr` Implementierung einer `OnValueChanged`-Methode eines `uiTab`-Modulators.

10.5.3.3 InitSerializeProperty

Bei einem Doppelklick werden alle Modulatoren entsprechend des lua-Scripts „`InitSerializeProperty`“ im Panel durchlaufen und mit einem `serializeIndex` ausgestattet.