

DMXControl

Sprachbeschreibung und Tutorial
für
Device- und Forms-Konfiguration

Version 2.8.1
Status: beta



PopSoft

<http://www.dmxcontrol.de/>

Autoren:

Frank Burghardt, Berlin
Stefan Krupop, Braunschweig

e-mail: info@dmxcontrol.de

Inhalt

1	EINLEITUNG	3
1.1	Nutzungsbedingungen	3
1.2	Systemanforderungen	3
2	Sprachbeschreibung	3
2.1	Allgemeines	3
2.2	Beispiel – Gedimmter Scheinwerfer	4
2.3	Aufgaben	5
2.4	Übersicht	6
2.4.1	Grafische Elemente	6
2.4.2	Funktionale Elemente	8
2.5	Syntax	9
2.5.1	Generische Attribute	9
2.5.2	Gerätebeschreibung	10
2.5.3	Kanalbeschreibung	10
2.5.4	Menübeschreibung	11
2.5.5	Steuerungselemente	12
2.5.6	Passive Gestaltungselemente	14
2.5.7	Hilfe	15
2.5.8	Sequences	15
3	Erweiterte Programmiermöglichkeiten	16
3.1	Generelle Prinzipien	16
3.2	Elementare Sprachelemente und Konventionen	16
3.3	XML Syntax	18
3.4	Eine Beispiel-Prozedur	18
4	Tutorial	20
4.1	Schritt 1: Studieren der Bedienungsanleitung	20
4.2	Schritt 2: Funktionalität der Form festlegen	20
4.3	Schritt 3: Erstellen des XML files	21
4.3.1	Device-Beschreibung	21
4.3.2	Beschreibung der Kanäle	21
4.3.3	Beschreibung der Forms	22
4.3.4	Weitere Bedienelemente	24
4.4	Schritt 4: Die höhere Schule	26
5	Fehlerhinweise	28
5.1	Bekannte Probleme	28
6	Appendix	29
6.1	Beispiel für die mögliche Komplexität von Kontextmenüs	29
6.2	XML Schema für DMXcontrol	30
6.3	Device-Beschreibung aus dem Tutorial	33

Für spezielle Hilfe bei der Erstellung von Gerätebeschreibungsfiles wenden Sie sich bitte an:

Daniel Miertz (DDF support)

e-mail: ddf-support@dmxcontrol.de

Version info	neu	
Rel. 2.8	Help tag Color picker	- z.B. für DMX-Tabelle des Gerätes - RGB control

1 EINLEITUNG

DMXControl ist ein Werkzeug zur Gestaltung Ihrer Lichtshow. Mit diesem Programm können Sie Ihre DMX-fähigen Geräte vom Computer steuern und benötigen dabei nur minimale Kenntnisse über das DMX-Konzept.

Dieses Tutorial beschäftigt sich mit der Erstellung von Gerätebeschreibungen für DMX-Geräte zur Integration in DMXControl. Es wird beschrieben, wie diesen Geräten auch spezifische Menüs zugeordnet werden können.

1.1 Nutzungsbedingungen

DMXControl ist Freeware. Die Weitergabe des Programms und der Dokumentationen ohne Änderungen ist erlaubt. Wir bitten alle Nutzer, sich für Feedback-Zwecke im DMXControl-Forum kostenlos zu registrieren.

Die Autoren übernehmen keine Verantwortung für eventuelle Schäden, die sich aus der Nutzung der Software ergeben.

1.2 Systemanforderungen

Für die Erstellung von Konfigurations- und Forms-Beschreibungen benötigen Sie im Prinzip nur einen Text-Editor. Eine Ansicht der Dateien im Internet Explorer kann vorteilhaft sein.

Konfigurations- und Forms-Beschreibungen sind Textdateien mit der Erweiterung „.xml“. Diese können natürlich auch mit bekannten XML-Tools erstellt werden. Das XML Schema für DMXControl befindet sich zur Information im Anhang. Falls Sie es für Ihren XML-Editor einsetzen wollen, finden Sie es auch als xds-file in unserem Downloadbereich.

2 Sprachbeschreibung

2.1 Allgemeines

Die Beschreibung von DMX-Geräten erfolgt bei DMXControl in einem XML-Format, das durch einen Parser in die interne Gerätebeschreibung transformiert wird.

Zur Zeit unterstützt DMXControl noch keinen grafischen Geräte-Editor. Daher gibt dieses Dokument Anleitung zur manuellen Erstellung von Gerätebeschreibungen mittels eines normalen Editors oder XML-Tools.

XML ist eine im Internet sehr gebräuchliche Sprachsyntax, die z.B. in anderen Dialekten bei Webseiten oder WAP-Seiten verwendet wird. Aber sie müssen kein Experte sein, um eine neue Gerätebeschreibung zu erstellen. Folgende einfache Hinweise genügen:

- Bitte achten Sie darauf, dass Ihr Editor keine (unsichtbaren) Steuerzeichen abspeichert. Beim Windows-Editor oder emacs wird es keine Probleme geben. Falls Sie Word o.ä. verwenden wollen, bitte die Datei im Textformat speichern.
- Der DMXControl-Dialekt von XML verlangt zu jedem öffnenden Tag („<tag>“) immer ein schliessendes Tag („</tag>“). Nur die Tags der letzten Ebene werden gleich implizit abgeschlossen (z.B. <item caption= "Weiss" value="0" />).

- Die Endzeichen „/>“ und </tag> sind semantisch äquivalent, also auch <item caption= "Weiss" value="0" > </item> ist gültig.
- Jedes Tag kann Attribute besitzen, die Sie unten in der Tabelle erklärt finden. Jeder Attributwert wird mit einem Gleichheitszeichen eingeleitet und der Wert muss immer in Doppelapostrophe eingeschlossen sein.
- Alle Tag- und Attribut-Bezeichner werden bei DMXControl in kleinen Buchstaben notiert.
- Die Tags sind hierarchisch zu definieren. Das sollten sie über entsprechende Einrückungen auch kenntlich machen.
- Die Reihenfolge der Attribute eines Tags ist in der Regel egal. Wo es ausnahmsweise doch auf die Reihenfolge ankommt, wird unten explizit beschrieben.
- Kommentarzeilen werden in der Form notiert <! –Das ist ein Kommentar -->
- Sie können sich xml-files (somit auch die DMX devices files) in übersichtlicher Form z.B. mit dem Internet Explorer anschauen.

So, das war es schon - schauen wir uns ein einfaches Beispiel an.

2.2 Beispiel – Gedimmter Scheinwerfer

Jedes Gerät sollte ein treffendes Icon erhalten, um in der Bühnendarstellung eine angepasste Darstellung zu erhalten. Dieses Icon wird im Unterverzeichnis Images abgelegt.

Hinweis: Die Icons können in DMXC hinterher noch geändert werden (Kontextmenü Bühne).



Abbildung 1: Icon für Scheinwerfer

Die vorangestellten Zeilennummern des folgenden XML-Beispielfiles sind nicht einzufügen; sie dienen nur der Beschreibung des Beispiels:

```

1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <device image="light.gif" initsequence="set 0 128" >
3      <channels>
4          <function channel="0" minvalue="0" maxvalue="255"
              name="Helligkeit" fade="yes" />
5      </channels>
6      <form width="177" height="85">
7          <deviceimage top="0" left="0" />
8          <devicename top="0" left="40" />
9          <deviceadress top="16" left="40" />
10         <slider channel="0" startvalue="0" endvalue="255"
              top="40" left="0" height="41" width="176" default="0" />
11     </form>
12 </device>

```

Zeile 1 teilt dem Parser die verwendete XML-Version mit.

Zeile 2 definiert die globale Beschreibung des neuen Gerätetyps inklusive Verweis auf das zu verwendende Icon und einer Initialisierung des Gerätes (die optionale initsequence wurde hier demonstriert, ist aber bei einem Scheinwerfer nur bedingt sinnvoll) .

Zeile 4 als Teil der Kanal-Beschreibung weist dem Geräte-DMX-Kanal mit der unter „channel“ angegebenen Nummer einen Namen zu und definiert den zulässigen DMX-

Wertebereich. Die channel-Nummern müssen daher immer von 0 aufsteigend ohne Unterbrechung für die einzelnen Kanäle angegeben werden. Die Zeilen 6-9 beschreiben das Form, also die grafischen Koordinaten im Kontextmenü für das Gerätebild, Namen und Startadresse in der Einheit „Pixel“.



Abbildung 2: Kontextmenü des gedimmten Scheinwerfers

Danach folgen die Bedienelemente mit ihren Eigenschaften, in diesem Beispiel ein slider mit grafischen Koordinaten. Für andere Gerätetypen könnten hier auch Bedienelemente wie Radiobuttons, Klapplisten oder Buttons deklariert werden. Die beiden Buttons rechts oben (Pin-Button und Move-Button) werden standardmäßig generiert.

2.3 Aufgaben

Bevor Sie Ihren Gerätetyp definieren, schauen Sie bitte auf unserer Webseite nach, ob es schon eine passende oder ähnliche Definitionen für Ihr Gerät gibt. Um ein neues Gerät zu definieren, modifizieren Sie am besten ein existierendes File eines ähnlichen Gerätes.

Natürlich haben die Autoren nicht Zugang zu allen DMX-Geräten der diversen Hersteller. Falls Sie ein neues Gerät definiert haben sollten, würden wir uns freuen, wenn Sie das Ergebnis der Allgemeinheit zur Verfügung stellen würden, indem Sie uns die entsprechende Dateien senden (dmxcontrol@gmx.de oder später das upload-Menü der Webseite benutzen).

Wenn Sie Ihre Arbeit abgeschlossen haben, legen Sie einfach das xml-file im „devices“-Unterverzeichnis in Ihrer DMX-Installation ab und speichern das entsprechende gif-Bild (im Format 32x32 Pixel, dabei hat sich der Transparenzmode bewährt) im Unterverzeichnis „images“ ab – und das Gerät sollte beim nächsten Start von DMXControl sichtbar sein.

2.4 Übersicht

Jede Beschreibung enthält vier Hauptteile (siehe Abbildung 3):

- Beschreibung globaler Eigenschaften (im obigen Beispiel Zeile 1-2)
- Beschreibung der einzelnen DMX-Kanäle (im Beispiel Zeile 3-5)
- Beschreibung des grafischen Kontextmenüs (im Beispiel Zeile 6-11)
- Prozedurcode (optional, im Beispiel nicht enthalten)

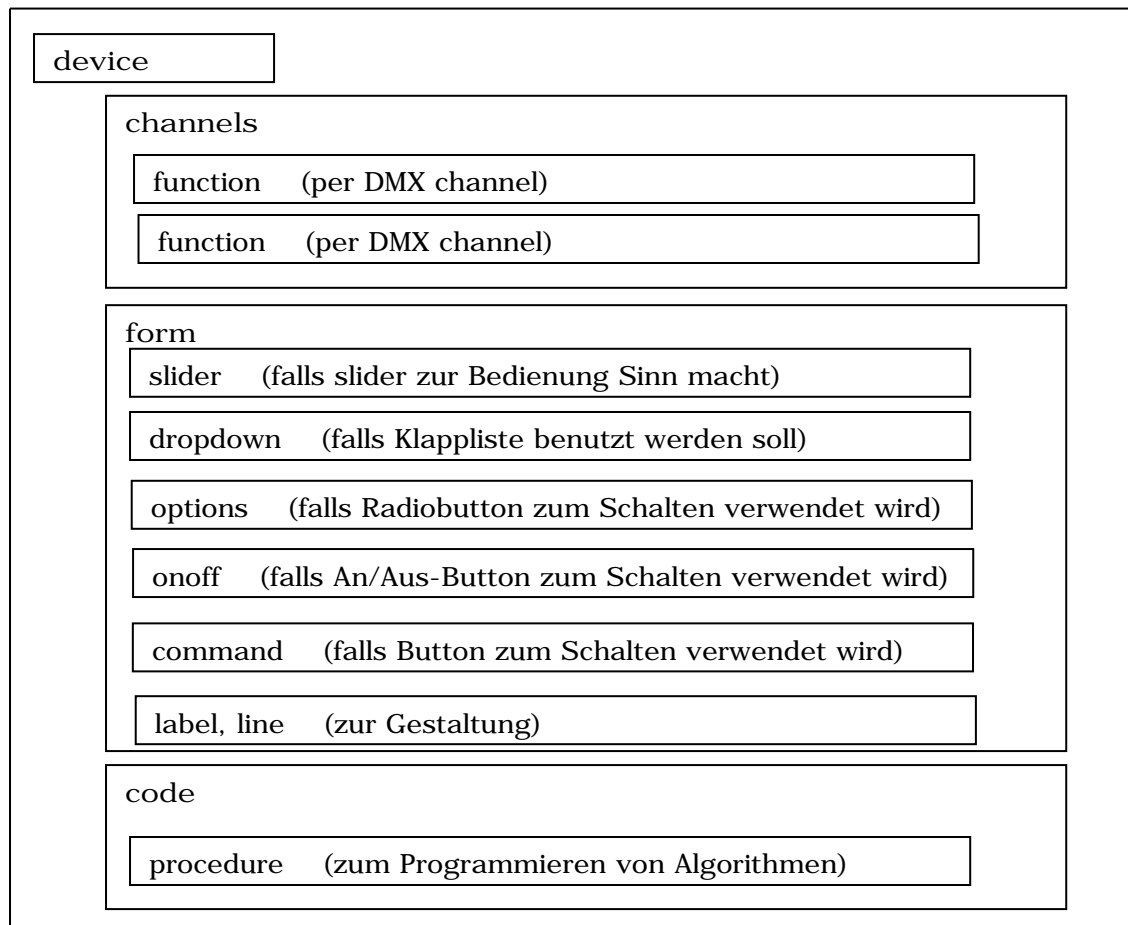


Abbildung 3: Bestandteile des Device- und Forms-Konfigurationsfiles

Alle Forms-Elemente müssen mit grafischen Koordinaten zur Positionierung im Kontextmenü versehen werden. Die Elemente „Label“ und „Line“ sind passiv und dienen nur der Gestaltung der Oberfläche. Die restlichen aktiven Elemente der Form (auch Steuerelemente genannt, z.B. onoff) können zur aktiven Steuerung des DMX-Gerätes eingesetzt werden.

2.4.1 Grafische Elemente

Abbildung 4 zeigt eine Übersicht über alle unterstützten grafischen Elemente. In der Erklärungstexten ist eine Zuordnung zu den erforderlichen Syntaxelementen vorgenommen.

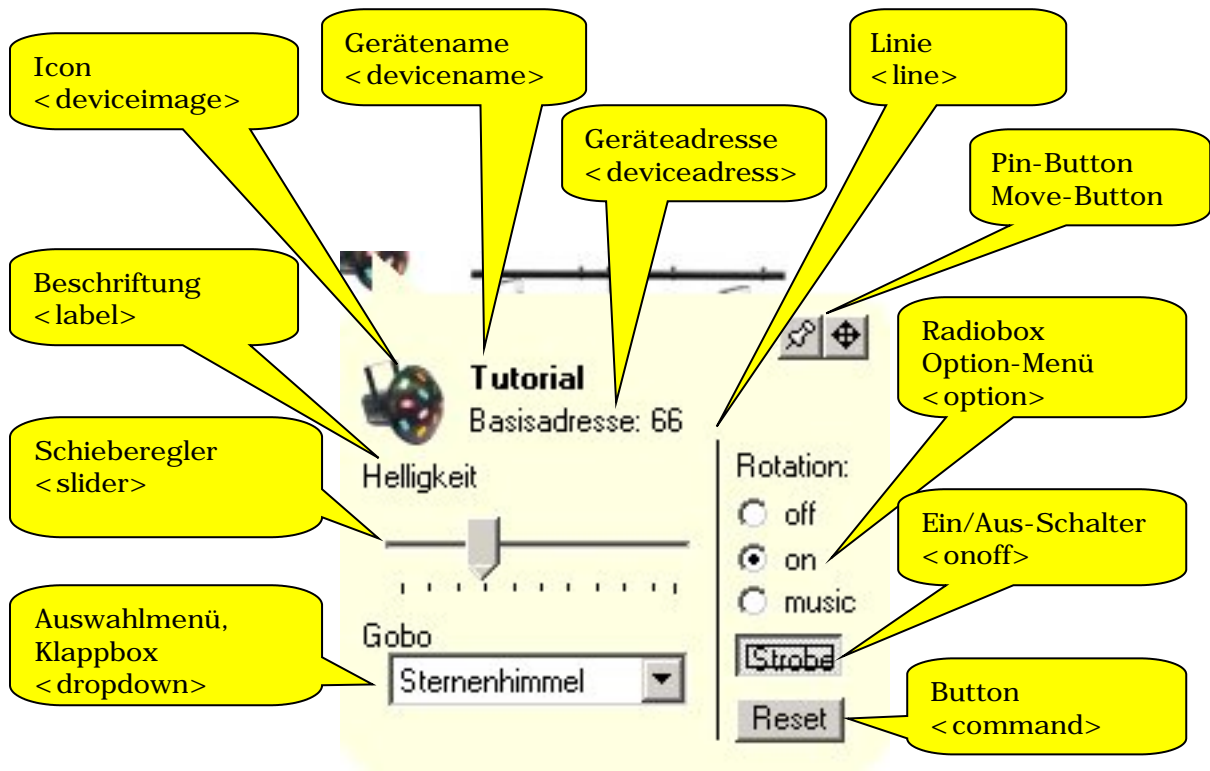
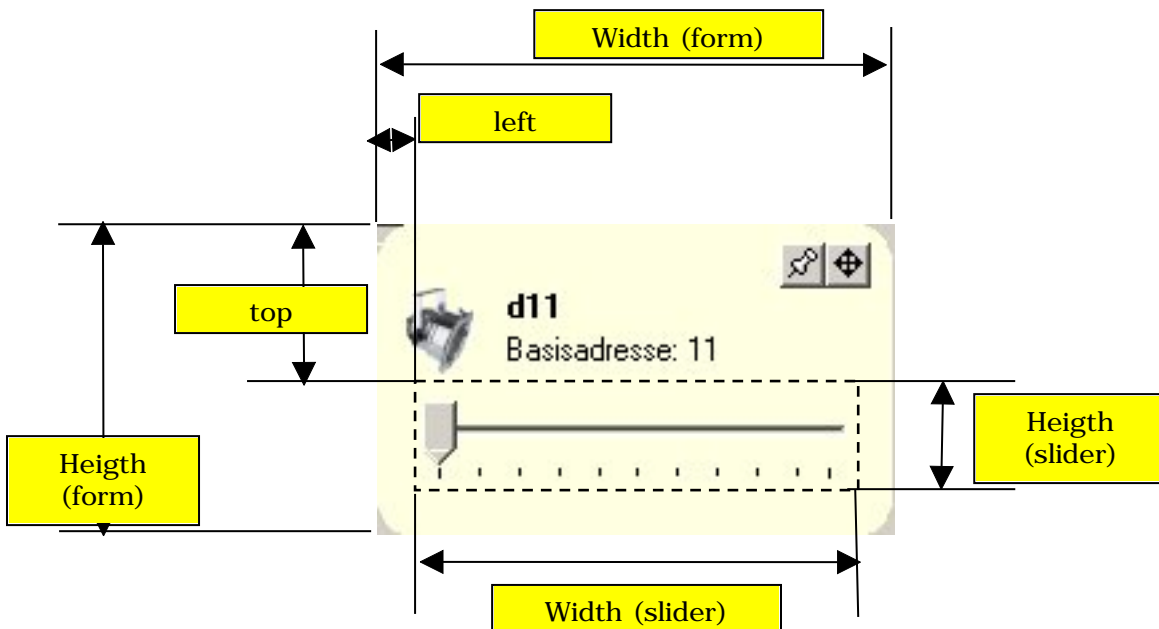


Abbildung 4: Alle unterstützten grafischen Elemente

Die folgende Abbildung illustriert die Bedeutung der grafischen Koordinaten:



x und y einordnen

Abbildung 5: grafische Koordinaten

2.4.2 Funktionale Elemente

Die aktiven Steuerungselemente interagieren in 3 möglichen Konzepten mit DMXControl bzw. dem Gerät, wobei bei der Definition eines konkreten Steuerelementes genau ein Konzept ausgewählt wird:

	Konzept	Beschreibung	Beispiel
1	Channel/value concept	Über das channel-Attribut wird das Steuerelement einem konkreten Kanal zugeordnet. Im Steuerelement werden konkrete Werte oder Wertebereiche definiert, die bei der Betätigung des Steuerelementes gesetzt werden.	- option Liste caption= "Sternenhimmel" value= "165" - Wertebereich des Schiebereglers startvalue= "0" endvalue= "255"
2	Sequences	Eine Anweisungsfolge wird im Steuerelement als string definiert, die beim Betätigen ausgeführt wird.	clicksequence= "set 1 75; set 2 100"
3	Action/procedures	Eine separat definierte Prozedur enthält komplexere Programmanweisungen. Die Prozedur wird über das action-Attribut zugeordnet und bei jedem Betätigen des Steuerelementes ausgeführt.	action= "SetGobo"

Insbesondere bedeutet das, dass das channel-Attribut und das action-Attribut alternativ in den Steuerelementen zu verwenden sind.

Bitte beachten Sie: Für viele Anwendungen sind die ersten beiden Konzepte ausreichend. So wie die Komplexität und Mächtigkeit der 3 Konzepte anwächst, steigen auch die Laufzeitanforderungen. Daher sollte immer die einfachste Variante zur Lösung einer Aufgabe gewählt werden.

Auch bei der Kanalbeschreibung kann ein action-Attribut verwendet werden. Das bedeutet, dass die Prozedur bei jeder Änderung des Kanal-Wertes aufgerufen wird.

In Abbildung 6 ist ein Beispiel aufgezeigt, wie verschiedene Steuerelemente mit den Kanälen interagieren können. Die Kanalwerte werden über „channel_n“ adressiert und können über die oben genannten 3 Konzepte gesetzt werden. Der aktuelle Wert, der mit einem Steuerelement verbunden ist, wird über ein dem Steuerelement zugeordnetes frei wählbaren Namen (Referenz) adressiert, hier mit „control_n“ bezeichnet.

Im Beispiel sind drei Steuerelementen „actions“ und zusätzlich „Referenznamen“ zugeordnet. Diese actions können einen oder mehrere channel-Werte setzen und sollten daher in der Regel mit „Setnnnnn“ bezeichnet werden. Weiterhin ist eine „action“ den channel_3 zugeordnet. Das ist dafür vorgesehen, die aktuellen Kanal-Werte zu lesen und den Steuerelementen mitzuteilen. Daher wird eine solche Prozedur in der Regel den Namen „Getnnnnn“ bekommen, da sie aktuelle Kanalwerte lesen muss.

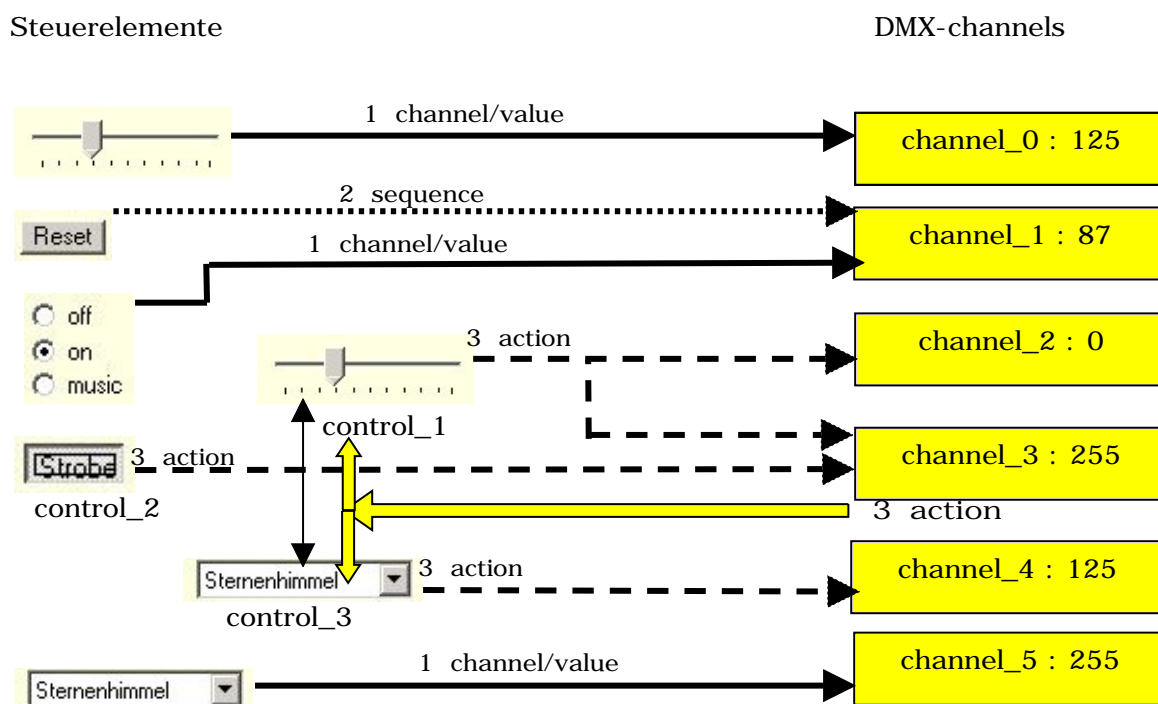


Abbildung 6: Interaction der Steuerelemente

Hinweis: Die Beschriftungen der Buttons haben hier keine Bedeutung.

2.5 Syntax

Dieses Kapitel beschreibt die Syntax der Device- und Forms-Konfigurationsfiles von DMXControl. Ein XML-Schema befindet sich im Anhang.

2.5.1 Generische Attribute

Zur Verkürzung der folgenden Tabellen werden sich oft wiederholende Attribute mit gleicher Bedeutung hier einmal global beschrieben.

Referenz	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
GP1	value	Zugeordneter DMX-Wert	integer	0, 128, 255
GP2	caption	Beschreibender Aufzählungswert Erklärender Name für eine Einstellung	string	Grün Stern
GP3	top	Relative y-Koordinate des Elementes von der linken oberen Ecke	integer	
GP4	left	Relative x-Koordinate des Elementes von der linken oberen Ecke	integer	
GP5	width	Breite des Elements	integer	
GP6	hight	Höhe des Elements	integer	

Alle Koordinatenangaben werden in Pixel beschrieben.

2.5.2 Gerätebeschreibung

Tag	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
< device >		Level 1		
	image	Dateiname des Icons	String, Filename .gif	Moon.gif
	initsequence	Setzt Initialwerte für die einzelnen DMX-Kanäle des Gerätes	Optional Benutzung z.B. für scanner startposition	set 0 15; set 7 128
< information >		Level 2 Zusätzlicher informaler Kommentar		
< name >		Level 3 Beliebiger Text		Custom Scanner
< vendor >		Level 3 nur informativ	optional	ShowTec
< deviceidentifier >		Level 3 nur informativ	optional	TG-3
< author >		Level 3 nur informativ	optional	T.Tutor
< panarea >		Scannerspezifische Information	optional	
	degree	Winkel, den der Scanner ausleuchtet	optional	85
	offset	Tbd.	optional	0
< tiltarea >		Scannerspezifische Information	optional	
	degree	Winkel, den der Scanner ausleuchtet	optional	176
	offset	Tbd.	optional	0
< help >		Level 2 Hilfetext (ASCII)	optional	

2.5.3 Kanalbeschreibung

Tag	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
< channels >		Pro DMX-Kanal ist ein "function"-Teil zu definieren		
< function >		Level 3 Subtag von „channels“		
	channel	Interne DMX Kanal-Nummern müssen immer von 0 aufsteigend ohne Unterbrechung für die einzelnen Kanäle definiert werden	integer	



	minvalue	Minimaler DMX Wert	integer	
	maxvalue	Maximaler DMX Wert	integer	
	name	Beschreibender Name für Kanal	string	Helligkeit
	fade	Fade-barkeit	string	no, yes
	type	Type des Kanals	string	dimmer, color, gobo, pan, panfine, tilt, tiltfine
	action	Aufruf von code (ein bestimmter Ablauf); Wird immer aufgerufen, wenn sich der Kanalwert ändert.	optional string	

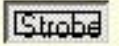
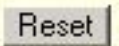
2.5.4 Menübeschreibung

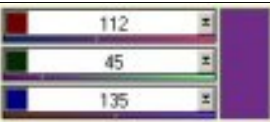
Tag	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
< form >		Level 2		
	width height	Siehe Generische Attribute	integer	
< deviceimage >		Level 3; Subtag von „form“ Relative Position des angezeigten Bildes		
	top left width height	Siehe Generische Attribute	integer	
< devicename >		Level 3 Subtag von „form“; Relative Position des angezeigten Namens		
	top left width height	Siehe Generische Attribute	integer	
< deviceaddress >		Level 3 Subtag von „form“; Relative Position der angezeigten Basisadresse		
	top left width height	Siehe Generische Attribute	integer	
< position >		Koordinatenkreuz für Moving Lights		
	top left height	Siehe Generische Attribute Hinweis: positioniere	integer	

	width	Pointer mit Initsequenz		
--	--------------	-------------------------	--	--

2.5.5 Steuerungselemente

Tag	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
				
< slider >		Level 3 Subtag von „form“; Erzeugt einen Schieberegler		
	top left height width	Siehe GP1 bis GP4	integer	
	channel	Zugeordneter Kanal alternativ zu action	optional	
	startvalue	Wertebereich des Schiebers (unterer Wert)	integer	0
	endvalue	Wertebereich des Schiebers (oberer Wert)	integer	255
	tickfreq	Skalierung, Abstand der Skalenunterteilung	integer	50
	smallchange	Änderungsrate z.B. beim Pfeiltastenscrollen	integer; VB-Eigenschaft	20
	largechange	Änderungsrate z.B. beim Mausklicken (nicht ziehen)	Integer; VB-Eigenschaft	50
	name	Referenzname für procedure code (Variable)	Optional string	
	action	Aufruf von Code bei jeder Änderung an diesem slider	optional	SetColor
				
< drop down >		Level 3 Subtag von „form“; Erzeugt Klappmenü		
	top	siehe GP3	integer	
	left	siehe GP4	integer	
	width	siehe GP5	integer	
	channel	Zugeordneter Kanal; Alternativ zu action	optional integer	
	name	Referenzname für procedure code (Variable)		
	action	Aufruf von Code bei jeder Änderung an diesem Klappmenü		
< item >		Level 4 Subtag von „dropdown“		
	caption	Erklärender Name		
	value	Wert		
< colorlist >		Fügen alle Einträge der dem Gerät zugewiesenen	optional Ersatz/Ergänz	

		Farbliste ein	ung für Item	
< gobolist >		Fügen alle Einträge der dem Gerät zugewiesenen Goboliste ein	optional Ersatz/Ergänzung für Item	
		<input type="radio"/> off <input checked="" type="radio"/> on <input type="radio"/> music		
< options >		Level 3 Erzeugt Radiobox Subtag von „form“;	Max. 5 radiobox groups sind zulässig	
	top left	siehe generische Attribute	integer	
	channel	Zugeordneter Kanal	integer	
	action	Aufruf von Code bei jeder Änderung an dieser Radiobox		
< option >		Level 4 Subtag von „options“		
	caption	siehe GP2	string	
	value top left	siehe generische Attribute	integer	
				
< onoff >		Level 3 Subtag von „form“; Erzeugt Ein/Aus –Button (Status bleibt gehalten)		
	top left width height	siehe generische Attribute		
	caption	Siehe GP2		
	name	Referenzname für procedure code		
	channel	Zugeordneter Kanal alternativ zu action	integer	
	onvalue	Wert bei ON	integer	
	offvalue	Wert bei OFF	integer	
	onsequence	Sequence, die bei ON ausgeführt wird	string	
	offsequence	Sequence, die bei OFF ausgeführt wird	string	
	action	Aufruf von Code bei jeder Änderung an dieser Checkbox		SetFog
				
< command >		Level 3 Subtag von „form“ Definiert neuen Button		
	top left width	Position siehe generische Attribute		
	caption	GP2		"Lampe

		Button-Beschriftung		zünden"
	clicksequence	Beim Buttonclick zu startende Aktivität ist im Attribut spezifiziert		"save 0;set 0 230;hold 5500;restore 0"
	downsequence	Beim Niederdrücken zu startende Aktivität ist im Attribut spezifiziert		Dto.
	upsequence	Beim Loslassen zu startende Aktivität ist im Attribut spezifiziert		Dto.
	action	Aufruf von Code beim Anklicken des Buttons	string	
				
< color picker >		Farbmischer für RGB-Geräte		
	mode	"rgb" oder "cym"	string	
	channel1	Zuordnung zum 1. Kanal	integer	0
	channel2	Zuordnung zum 1. Kanal	integer	1
	channel3	Zuordnung zum 1. Kanal	integer	2
	layout	Wahl zwischen 1 oder 2 (zwei Layoutmöglichkeiten)	integer	1

Beispiel:

```
< colorpicker mode="rgb" channel1="0" channel2="1" channel3="2" layout="2" top="40" left="0" height="75" width="177"/>
```

Bitte halten Sie die Maustaste gedrückt, wenn Sie über das Klappmenü die Einstellung im Farbbalken ändern wollen. Eine Direkteingabe des numerischen Wertes ist ebenfalls möglich.

2.5.6 Passive Gestaltungselemente

Tag	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
< line >		Level 3 Subtag von „form“; Gestaltungselement; erzeugt eine Linie mit angegebenen Endkoordinaten		
	x1, y1 x2, y2	Koordinaten	integer	
< label >		Level 3 Subtag von „form“; Gestaltungselement; Zusätzliche Beschreibung eines Bedienelementes		
	top left	Position, siehe GP3 und GP4	integer	
	caption	Anzuzeigender Text		Helligkeit

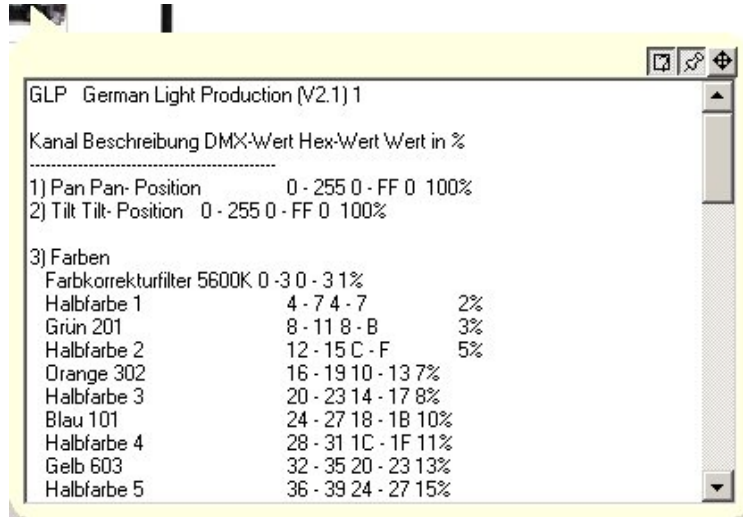
--	--	--	--	--

2.5.7 Hilfe

Folgende Abbildung zeigt ein Beispiel für ein Hilfemenü. Hilfemenüs sind optional. Es wird empfohlen, hier die DMX Belegung des Gerätes anzuzeigen.

Die Hilfebeschreibung erfolgt im ASCII-Format. Bitte erhöhen Sie die Übersichtlichkeit durch Formatierung mit Tabulator-Zeichen oder Unterstreichungen.

Die Hilfe kann durch Klick auf den Fragezeichen-Button (neben dem Pin-Button) ein- und ausgeschaltet werden.



2.5.8 Sequences

Es werden folgende Sequence-Kommandos unterstützt (alle Parameter sind integer):

Operation	Bedeutung
save >channel<	Speichert intern aktuellen Wert des DMX-Kanals >channel<
set >channel< >value<	Setzt den Kanal >channel< auf Wert >value<
hold >time<	Timer (wartet >time< in msec.)
restore >channel<	Setzt den intern gespeicherten Wert des DMX-Kanals >channel< wieder zurück

3 Erweiterte Programmiermöglichkeiten

Die Device- und Form-Konfiguration von DMXControl erlaubt auch die Programmierung von algorithmischen Veränderungen der DMX-Signale, die bei der Bedienung von Steuerelementen wie onoff-Button, Klappmenü (dropdown) oder Schieberegler (slider) automatisch ausgeführt werden (siehe actions in Kapitel 2.4.2 .)

Sie können also die Werte, die den Steuerelementen zugeordnet sind, über logische Bedingungen und Formeln setzen. Die Prozeduren dienen dazu, z.B. Mehrfachbelegungen von einem Kanal behandeln zu können (z.B. Gobodrehung liegt auf gleichem Kanal wie die Goboauswahl), so dass der Wert je nach Drehgeschwindigkeit anders berechnet werden kann.

Diese Prozeduren erlauben Ihnen auch die Programmierung von Abhängigkeiten der Kanäle oder Bedienelemente eines Gerätes, z.B. verbinden Sie bestimmte Farben mit Gobos in fester oder algorithmischer Zuordnung oder Sie ordnen bestimmte Geschwindigkeiten irgendwelche Farben zu.

3.1 Generelle Prinzipien

Die Prozeduren werden als action-Attribut den <function>-Tags (Kanälen) oder den Steuerungselementen <onoff>, <slider> oder <dropdown> zugeordnet. Bei Bedienung entsprechender Steuerungselemente wird der Code der Prozedur automatisch interpretiert und ausgeführt. Ist die action dagegen einem Kanal (function) zugeordnet, so wird sie bei jeder Änderung des zugehörenden Kanalwertes ausgeführt.

Die Prozeduren verwenden „Referenzen“ auf die Werte der Steuerelemente, die innerhalb der form-Definition der Steuerungselemente durch das „name“-Attribut deklariert sind, z.B.

```
<dropdown top="16" left="207" width="113" name="color_color"
action="SetColor">
```

Dies bedeutet: Die Prozedur SetColor wird bei Bedienung des Klappmenüs ausgeführt und der zu ändernde Wert des Klappmenüs wird als "color_color" adressiert.

Weiterhin existieren implizite Referenzen als standardmäßige Zuordnung zu den Kanälen. So ist z.B. „channel_2“ implizit mit dem Kanal (function) mit channel=2 verbunden, dem auch die action „GetColor“ zugeordnet ist.

```
<function channel="2" minvalue="0" maxvalue="255" name="Farbe"
fade="no" action="GetColor" colorchannel="yes"/>
```

Variablen im eigentlichen Sinne als frei belegbare Speicherwerte sind bisher nicht definiert.

3.2 Elementare Sprachelemente und Konventionen

Während die Referenzen mit ihrem Namen bezeichnet werden ("color_color"), wird der aktuelle Wert vom zugeordneten Steuerelement durch geschweifte Klammern adressiert (" { color_color} ").

Um die Interpretation des Prozedurcodes durch einen Parser zu vereinfachen, sind in der jetzigen Programmversion einige Konventionen vereinbart, die Sie leider einhalten müssen, obwohl sie nicht sehr nutzerfreundlich erscheinen. Zum Vergleich und zur

leichteren Erlernbarkeit liefern die nachfolgenden Tabellen auf der rechten Seiten immer eine generische Sprachvariante.

Folgende Zeichen dienen als Separatoren:

Zeichen	Bedeutung
!	beginnt einen Befehl
	separiert die einzelnen Bestandteile
\$	beendet einen Befehl

Es werden folgende arithmetische Operationen unterstützt:

Operation	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
mod	Modulo-Operation (Division mit Rest)

Jeder Ausdruck muss von einer Klammer umgeben sein, um berechnet zu werden. Folgende Beispiele illustrieren die Verwendung arithmetischer Operationen:

DMXControl code	Generische Sprachvariante
!set_channel 2 (227+{color_speed})\$	channel_2 := 227 + color_speed
!set_channel 3 (((gobos_gobo)-1)*51)+25)\$	channel_3 := (gobos_gobo-1)*51+25
!set_control gobos_speed (((channel_3)-1) mod 51)-26)\$	gobos_speed := ((channel_3 - 1) mod 51)-26

Die verwendeten Standardoperatoren haben folgende Semantik:

Operation	Bedeutung
set_control >cntr< >val<	setzt das control mit dem angegebenen Namen >cntr< auf den Wert des Ausdruckes >val<
set_channel >ch< >val<	setzt den angegebenen Kanal >ch< auf den Wert des Ausdruckes >val<

Folgende Vergleichsoperatoren können verwendet werden:

Operation	Alternative Schreibweise	Bedeutung
>	> gt	größer
<	< lt	kleiner
=	&eq; eq	gleich

Mit Hilfe der Vergleichsoperationen können Bedingungen formuliert werden, da DMXControl die if-Anweisung unterstützt.

DMXControl code	Generische Sprachvariante
!if {channel_3} < 1	if channel_3 < 1 then

Die Prozedur "INITCONTROLS" wird beim Öffnen der Form ausgeführt, um die Controls auf die Werte zu setzen, die durch die aktuellen DMX-Werte vorgegeben werden. Darin sollten also nur "Get"-Funktionen aufgerufen werden, nur lesen! Initcontrols muss nicht definiert sein, ist aber empfohlen, damit die Controls immer den aktuellen Zustand des Gerätes abbilden.

Noch zu vervollständigen: In den Farb- und Gobolisten lassen sich Farben mit Checkbox markieren. Diese werden dann statt {favcolor??_x} eingefügt (Name bzw. Wert)

3.3 XML Syntax

Tag	Attribut	Hierarchie und Eigenschaft	Bemerkung	Beispiel
<code>		Level 2 Subtag von „device“; Enthält alle Prozeduren		
<procedure>		Definiert code für eine action		
	name	Name der Prozedur	string	

3.4 Eine Beispiel-Prozedur

Dieses Beispiel zeigt eine Prozedur, die die Werte der Steuerungselemente „color_color“ und „color_speed“ eines Moving Heads in Abhängigkeit vom aktuellen DMX-Wert des Kanals 2 (Farbwechsler) setzt. Der Mac 250+ hat 4 Möglichkeiten für das Farbrad:

- Feste Farbe (erstes If im Beispiel unten),
- Drehung mit Uhrzeigersinn (2. If im Beispiel unten),
- Drehung gegen Uhrzeigersinn (3. If im Beispiel unten) sowie
- zufällige Farben mit unterschiedlicher Geschwindigkeit (4. If).

Hier die Deklarationen im Konfigurationsfile, die relevanten Variablen sind fett hervorgehoben:

```
<function channel="2" minvalue="0" maxvalue="255" name="Farbe" fade="no"
  action="GetColor" colorchannel="yes"/>
```

„color_color“ ist die Referenz auf den Wert der Klappbox zur Farbauswahl.

```
<dropdown top="16" left="207" width="113" name="color_color"
  action="SetColor">
```

„color_speed“ ist die Referenz auf den Wert des Schiebereglers zur Geschwindigkeit:

```
<slider top="16" left="320" height="25" width="65" startvalue="0"
  endvalue="18" tickfreq="9" smallchange="1" largechange="5"
  name="color_speed" action="SetColor"/>
```

Die Prozedur unterteilt die DMX_Werte des Kanal 2 in 4 Intervalle und führt unterschiedliche Zuweisungen aus. Der aktuelle Wert wird also geprüft und die Controls werden entsprechend eingestellt.

DMXControl code	Generische Sprachvariante
<pre> <procedure name='GetColor'> !if { channel_2} &lt; 208 !set_control color_color { channel_2}\$!set_control color_speed 0\$!if ({ channel_2} &gt; 207) and ({ channel_2} &lt; 227) !set_control color_color -1\$!set_control color_speed (226- { channel_2})\$ \$!if ({ channel_2} &gt; 226) and ({ channel_2} &lt; 246) !set_control color_color -2\$!set_control color_speed ({ channel_2}-227)\$ \$!if ({ channel_2} &gt; 245) and ({ channel_2} &lt; 256) !set_control color_color -3\$!set_control color_speed ((255- { channel_2}) * 2)\$ \$ \$ </procedure> </pre>	<pre> Procedure GetColor() begin if channel_2 < 208 then color_color := channel_2; color_speed := 0; (else) if (channel_2 > 207) and (channel_2 < 227) then color_color := -1; color_speed := 226-channel_2; (else) if (channel_2 > 226) and (channel_2 < 246) then color_color := -2; color_speed := channel_2-227 (else) if (channel_2 > 245) and (channel_2 < 256) then color_color := -3; color_speed:= (255-channel_2) * 2; ; end </pre>

4 Tutorial

Dieses Tutorial zeigt Ihnen die Erstellung und praktische Anwendung von DMX-Konfigurationsfiles am Beispiel des Gobowechslers TG-3.

4.1 Schritt 1: Studieren der Bedienungsanleitung

Der TG-3 belegt 2 DMX-Kanäle. In der Beschreibung finden Sie folgende DMX-Informationen:

1. DMX-Kanal	Feature	2. DMX-Kanal	Feature
0-15	white	0-5	Lamp off
16-31	red	6-63	open
32-47	dark blue	64-127	Strobe-effect (5 flashes/second)
48-63	green	128-132	closed
64-79	yellow	133-135	reset
80-95	orange	136-143	open
96-111	pink	144-151	closed
112-127	light blue	152-159	Gobo 1: Schnipsel
128-255	white	160-167	Gobo 2: Sternenhimmel
		168-175	Gobo 3: Sonne
		176-183	Gobo 4: Torten-Dreiecke
		184-191	Gobo 5: Dreieck
		192-199	Gobo 6: 8-flügliges Windrad
		200-207	Gobo 7: 5-eckiger Stern
		208-215	Gobo 8: dreigeteilter Kreisring
		216-231	Gobo 9: 4-flügliges Schaufelrad
		232-255	Gobo 10: 4 Quadrate multicolor

4.2 Schritt 2: Funktionalität der Form festlegen

Bei diesem Gerät bietet es sich an, jeweils ein Klappenmenü für die Farbe und für die Gobos anzulegen. Weiterhin soll es einen Aus-Schalter geben und einen Button, um den Stroboskop-Effekt zu starten. Für alle Fälle legen wir auch einen reset-Button an. Mit folgender Skizze wird das entsprechende Form symbolisiert:

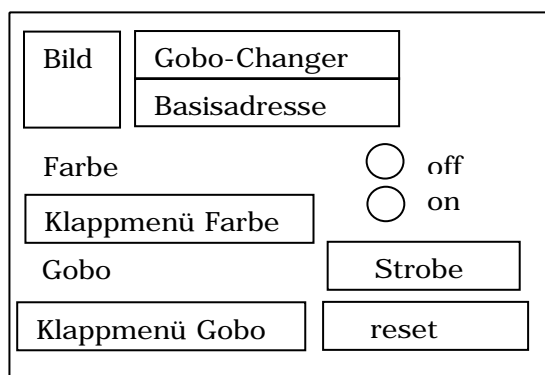


Abbildung 7: Skizze für das TG-3 Kontextmenü

4.3 Schritt 3: Erstellen des XML files

Vorteilhafterweise wird ein existierendes XML-file wiederverwendet. Wir beginnen mit der device-Beschreibung. Die Zeilenummerierung dient nur der Erklärung des Beispiels.

4.3.1 Device-Beschreibung

Wir benötigen standardmäßig die Zeile 1 mit der XML-Version und das <device>-Tag. Entweder wird ein neues Icon im Pixel Format 32x32im image-Unterverzeichnis des device-Verzeichnisses angelegt oder man prüft die Wiederverwendbarkeit der vorhandenen Bilder. In unserem Fall ähnelt das Bild moon.gif stark unserem TG-3, so dass wir uns dafür entscheiden.

Vorsichtshalber wird eine Initialisierung des Gerätes vorgesehen, aber noch nicht ausgefüllt (Zeile 2).

Im <information>-Tag beschreiben wir das Gerät (Zeile 3-5) und verewigen uns als Autor.

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <device image="moon.gif" initsequence=" " >
3   <information>
4     <name>EUROLITE Gobo Changer TG-3) von Theo Tutorial </name>
5   </information>
...
   <help>
     1. DMX-Kanal Feature
     -----
     0-15 white
     16-31 red
     32-47 dark blue
     2. DMX-Kanal Feature
     -----
     0-5 Lamp off
     6-63 open
     64-127 Strobe-effect (5 flashes/second)
   </help>
x </device>

```

Nun wird diese Datei erst mal unter dem Namen GoboChangerTG3.xml im device-Verzeichnis gespeichert.

4.3.2 Beschreibung der Kanäle

Dieser Teil beginnt mit dem <channel>-Tag. Wir haben 2 DMX-Kanäle und benötigen daher zwei <function>-tags. Von einem existierenden Scanner-Gerät wandeln wir dazu die Kanäle „Farbe“ und „Gobo“ ab.

```

6 <channels>
7   <function channel="0" minvalue="0" maxvalue="255" name="Farbe" fade="no"
8     type="color"/>
9   <function channel="1" minvalue="0" maxvalue="255" name="Gobo" fade="no"
10    type="gobo"/>
11 </channels>

```

Der Farb-Kanal erhält die interne Kanaladresse „0“ und der Gobo-Kanal die „1“. Minimal- und Maximalwerte für die DMX-Kanäle sehen wir in der Tabelle oben (jeweils 0 und 255, Zeilen 7 und 9). Da wir es hier mit rein diskreten Werten zu tun haben, macht Fading

keinen Sinn, also „no“. Beiden Kanälen muß nun noch der Typ zugewiesen werden, was selbsterklärend ist (Zeile 8 und 10).

Nun wird die ergänzte Datei wieder abgespeichert. Beim Starten von DMXControl können Sie nun schon das TG-3 als Gerät anlegen. Im Kanalzuweisungs-Tool (bei der Einrichtung im Submaster oder bei Effekten) kann man bereits die beiden definierten Kanäle sehen.

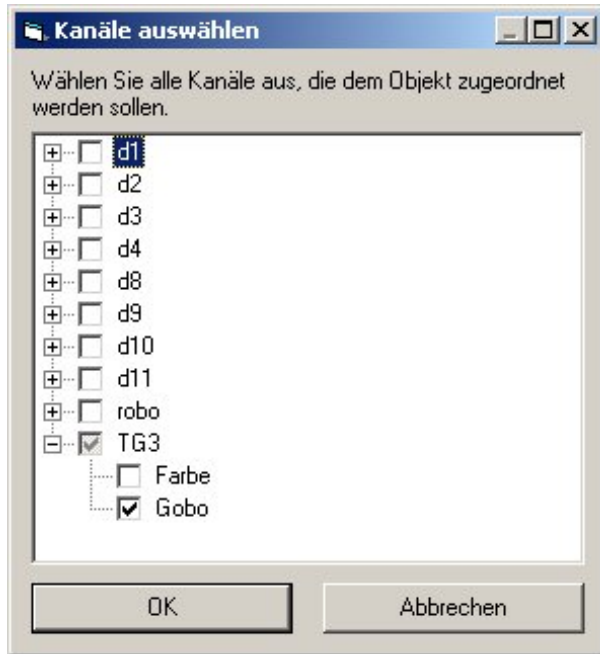


Abbildung 8: Kanalauswahl

Nun können wir auch Zeile 2 ergänzen und das reset-Kommando als „initsequence“ einführen. Dazu senden wir auf beiden Kanälen ein Signal (weiss und open).

```
2 <device image="moon.gif" initsequence="set 0 10; set 1 50" >
```

4.3.3 Beschreibung der Forms

Bisher haben wir alle relevanten DMX-Eigenschaften des Gerätes definiert und damit DMXControl bekannt gemacht. Alle folgenden Änderungen beziehen sich nur noch auf das User interface. Diese sind sofort beim Öffnen des Popup-Menüs verfügbar und sie müssen während ihrer Testphase für das Beschreibungsfile nicht mehr ständig DMXControl neu starten.

Beim obigen Test wird Ihnen aufgefallen sein, dass das Kontextmenü für das Gerät leer war und die Bedienung über das Submaster-Tool sehr umständlich ist. Also setzen wir unseren Menü-Entwurf schnell in die Formbeschreibung um.

Dazu beginnen wir mit dem <form>-tag.

```
12 <form width="177" height="134">
13   <deviceimage top="0" left="0"/>
14   <devicename top="0" left="40"/>
15   <deviceaddress top="16" left="40"/>
...
xx </form>
```

Nun sehen wir schon das Icon, den Gerätenamen und die Basisadresse im Kontextmenü.

Im nächsten Schritt definieren wir die Klappmenüs. Für die grafischen Koordinaten berücksichtigen wir die Angabe in Pixel. Beide Klappmenüs bekommen ein Label (Zeile 16 und 27). Vergessen Sie bitte bei den <dropdown>-Tags nicht das channel-Attribut, damit DMXControl weiss, auf welchen Kanal sich die Wertzuweisung bezieht.

Die Beschreibung der Klappmenüs kann man praktisch aus der Tabelle oben im Kapitel 4.1 ableiten (Zeilen 18-25 und 29-40).

```

16 <label top="33" left="0" caption="Farbe"/>
17   <dropdown top="53" left="0" width="113" name="color_color"
18     channel="0">
19     <item caption="white" value="10"/>
20     <item caption="red" value="20"/>
21     <item caption="dark blue" value="40"/>
22     <item caption="green" value="50"/>
23     <item caption="yellow" value="70"/>
24     <item caption="orange" value="90"/>
25     <item caption="pink" value="105"/>
26     <item caption="light blue" value="120"/>
27   </dropdown>
28 <label top="80" left="0" caption="Gobo"/>
29   <dropdown top="103" left="0" width="113" name="gobo_gobo"
30     channel="1">
31     <item caption="Schnipsel" value="155"/>
32     <item caption="Sternenhimmel" value="165"/>
33     <item caption="Sonne" value="170"/>
34     <item caption="Torten-Dreiecke" value="180"/>
35     <item caption="Dreieck" value="188"/>
36     <item caption="8-flügliges Windrad" value="195"/>
37     <item caption="5-eckiger Stern" value="205"/>
38     <item caption="dreigeteilter Kreisring" value="210"/>
39     <item caption="4-flügliges Schaufelrad" value="220"/>
40     <item caption="4 Quadrate" value="235"/>
41     <item caption="Open" value="140"/>
42     <item caption="closed" value="130"/>
43   </dropdown>

```

Nun sieht das Kontextmenü schon folgendermaßen aus:



Abbildung 9: Dropdown Menüs im Kontextmenü

Sie können jetzt schon die Farbe und das Gobo über das Menü einstellen.

4.3.4 Weitere Bedienelemente

Nun erweitern wir das Menü um die restlichen Bedienelemente. Das Ausschalten des Gerätes erreichen wir mit einem Radiobutton über das `<option>`-Tag. Die wesentliche Information hierbei ist, dass dem Kanal 1 der Wert „0“ beim Anklicken der „off“-Option zugewiesen wird (Zeile 44 und 45).

```
42 <label top="33" left="133" caption="Off:"/>
43   <options channel="1" top="50" left="133">
44     <option value="0" caption="off" left="0" top="0"/>
45     <option value="155" caption="on" left="0" top="250"/>
46   </options>
```

Den Strobe-Effekt steuern wir mit einem OnOff-Button. Dazu wird das `<onoff>`-Tag verwendet. Der Button bleibt „gedrückt“, bis er erneut bedient wird. In Zeile 48 wird jedem Zustand ein DMX-Wert zugeordnet.

```
47   <onoff caption="Strobe" top="97" left="133" width="41" channel="1"
48     onvalue="110" offvalue=" 140" />
```

Zuletzt wird der Reset-Button programmiert. Dazu verwenden wir das `<command>`-Tag, weil die reset-Aktivität nur gestartet werden soll, aber keine Zustandsänderung eines Steuerungselementes gewünscht ist.

```
49   <command top="120" left="133" width="41" caption="Reset"
50     clicksequence="save 1;set 1 134;hold 5500;restore 1"/>
```

Hier wird der Kanal im Attribut `clicksequence` angesprochen (Zeile 50). Im einzelnen werden folgende Aktionen „auf Buttondruck“ durchgeführt:

- Sichern des aktuellen Wertes von Kanal 1
- Setzen des Signals für Reset
- Gültigkeit des Wertes für 5500 msec setzen
- Nach Ablauf dieser Zeit wird der Ursprungswert wieder angenommen.

Nun ist es geschafft. Wir haben folgende Form erstellt, die unseren Entwurf vollständig umgesetzt hat:



Abbildung 10: Das komplette Kontextmenü

Hier noch einmal das vollständige XML-File zur Übersicht:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<device image="moon.gif" initsequence="set 0 10;set 1 10">
  <information>
    <name>EUROLITE Gobo Changer TG-3) von Theo Tutorial </name>
  </information>
<channels>
  <function channel="0" minvalue="0" maxvalue="255" name="Farbe" fade="no"
colorchannel="yes"/>
  <function channel="1" minvalue="0" maxvalue="255" name="Gobo" fade="no"
gobochannel="yes"/>
</channels>

<form width="177" height="134">
<deviceimage top="0" left="0"/>
<devicename top="0" left="40"/>
<deviceaddress top="16" left="40"/>

  <label top="33" left="0" caption="Farbe"/>
  <dropdown top="53" left="0" width="113" channel="0">
    <item caption="white" value="10"/>
    <item caption="red" value="20"/>
    <item caption="dark blue" value="40"/>
    <item caption="green" value="50"/>
    <item caption="yellow" value="70"/>
    <item caption="orange" value="90"/>
    <item caption="pink" value="105"/>
    <item caption="light blue" value="120"/>
  </dropdown>

  <label top="80" left="0" caption="Gobo"/>

  <dropdown top="103" left="0" width="113" channel="1">
    <item caption="Schnipsel" value="155"/>
    <item caption="Sternenhimmel" value="165"/>
    <item caption="Sonne" value="170"/>
    <item caption="Torten-Dreiecke" value="180"/>
    <item caption="Dreieck" value="188"/>
    <item caption="8-flügliges Windrad" value="195"/>
    <item caption="5-eckiger Stern" value="205"/>
    <item caption="dreigeteilter Kreisring" value="210"/>
    <item caption="4-flügliges Schaufelrad" value="220"/>
    <item caption="4 Quadrate" value="235"/>
    <item caption="Open" value="140"/>
    <item caption="closed" value="130"/>
  </dropdown>

  <label top="50" left="133" caption="Off:"/>

  <options channel="1" top="50" left="133">
    <option value="0" caption="off" left="0" top="0"/>
    <option value="155" caption="on" left="0" top="250"/>
  </options>
```

```

<onoff caption="Strobe" top="97" left="113" width="41" channel="1"
  onsequence="set 1 110" offsequence="set 1 140" />

<command top="120" left="113" width="41" caption="Reset"
  clicksequence="save 1;set 1 134;hold 5500;restore 1"/>
</form>
</device>

```

4.4 Schritt 4: Die höhere Schule

Unser Form ist ja schon eine schöne Sache geworden, aber im Prinzip können wir damit nur statische Werte in den beiden Kanälen setzen. Auch gibt es andere Nachteile: Nach Änderungen über ein fremdes Element, z.B. Submaster oder onoff-Button werden nicht alle neuen Werte automatisch in den Klappmenüs umgestellt. In diesem Kapitel wird gezeigt, wie auch Abhängigkeiten zwischen den Steuerungselementen programmiert werden können.

Wir wollen folgende Effekte programmieren:

1. Nur ausprobieren, wie man Werte anstelle channel/value mit action setzt.
2. Die dropdown-Elemente sollen jederzeit den aktuellen Wert anzeigen. (Anm.: Sollten sie eigentlich auch so, sofern der aktuelle Wert auch als item definiert ist, daher ist das zu prüfen.)
3. Wenn das Gobo „Sonne“ gewählt wird, soll es automatisch immer in „gelb“ dargestellt werden (weil unser GoboChanger im Vergleich zu Scannern ziemlich dumm ist, können wir kein besseres Beispiel liefern, obwohl vielleicht nicht jeder Nutzer Wert auf diesen Effekt legt).

Dazu deklarieren wir die actions „GetColor“ und „GetGobo“, die jeweils den aktuellen Kanalwert lesen sollen. Weiterhin benötigen wir an der Gobo-Klappbox einen Referenzwert (als gobo_gobo bezeichnet) und eine action, über die wir Werte der DMX-Kanäle setzen wollen. Die Farb-Klappbox bekommt zwar keine action (sie steuert weiter über channel/value), aber wir definieren auch hier einen Referenzwert zum Aktualisieren des dropdown-Wertes, wir sprechen dieses Steuerelement also über „color_color“ an.

```

<function channel="0" minvalue="0" maxvalue="255" name="Farbe" fade="no"
  type="color" action="GetColor" />
<function channel="1" minvalue="0" maxvalue="255" name="Gobo" fade="no"
  type="gobo" action="GetGobo"/>

<dropdown top="103" left="0" width="113" name="gobo_gobo"
  action="SetSunColor">
<dropdown top="53" left="0" width="113" name="color_color" channel="0" >

```

Zur Umsetzung der Anforderungen müssen wir im <code>-Teil die Prozeduren zu den „actions“ definieren. Beginnen wir mit den GET-Prozeduren. Die Zeilen 101 und 107 lesen eigentlich nur die aktuellen channel-Werte und weisen diese den Steuerelementen zu. Die jeweils folgenden beiden Zeilen sollen die Intelligenz des Prozedurmechanismus demonstrieren: Wenn uns die aktuellen Werte nicht gefallen, weil sie z.B. gar keinem Listenelement zuzuordnen sind, werden sofort „manipulierte“ Werte berechnet (anstelle der „70“ steht normalerweise ein komplexerer Ausdruck). Man muss prüfen, ob diese Werte ggfs. auch auf den Kanal zu senden sind.

```

100 <procedure name='GetColor'>
101 !set_control|color_color|({channel_0})$

```

```

102 !if|{channel_0} &gt; 127|
103   !set_control|color_color|(70)$
104 $
105 </procedure>

106 <procedure name='GetGobo'>
107   !set_control|gobo_gobo|({channel_1})$
108   !if|{channel_1} &lt; 136|
109     !set_control|gobo_gobo|(199)$
110   $
111 </procedure>

```

In der Prozedur INITCONTROLS werden beide Get-Prozeduren aufgerufen, damit bei jedem Öffnen des Kontextmenüs die Werte aktualisiert werden.

```

112 <procedure name='INITCONTROLS'>
113   !call|GetColor$
114   !call|GetGobo$
115 </procedure>

```

Nun fehlt noch die Prozedur, mit der wir den Gobo-Wert setzen. Weil diese auch die 3. Anforderung erfüllen soll, wird Sie "SetSunColor" genannt.

In Zeile 117 wird einfach der Wert des Steuerelementes an den Kanal 1 gesendet. In Zeile 118 wird geprüft, ob dieser Wert gerade dem Gobo „Sonne“ entspricht. Falls ja, wird dem Farb-Kanal mitgeteilt, dass die Farbe „gelb“ gewünscht wird.

```

116 <procedure name="SetSunColor">
117   !set_channel|1|({gobo_gobo})$
118   !if|(({gobo_gobo} &gt; 167) and ({gobo_gobo} &lt; 176))|
119     !set_channel|0|70$
120   $
121 </procedure>

```

Im Anhang finden Sie noch einmal das vollständige xml-file für diesen „intelligenten“ GoboChanger.

Wir wünschen Ihnen nun viel Erfolg bei der Erstellung der Gerätebeschreibungen Ihrer DMX-Geräte.

Und bitte lassen Sie uns durch Feedback an Ihrem Erfolg teilhaben.

5 Fehlerhinweise

5.1 Bekannte Probleme

Hier sollen bekannte Probleme des Parsers und der Forms-Konfiguration festgehalten werden.

Problem/Fehlermeldung	Mögliche Ursache
Absturz bei Prozeduraufruf	Überprüfen Sie, ob alle „Klammern“ im Code gesetzt sind
Werte ändern sich nicht beim Prozeduraufruf	Möglicherweise fehlen geschweifte Klammern falsch: <code>!if color_mode eq -2 !set_channel 2 color_speed+60\$</code> richtig: <code>!if{ color_mode} eq -2 !set_channel 2 {color_speed}+60\$\$</code>

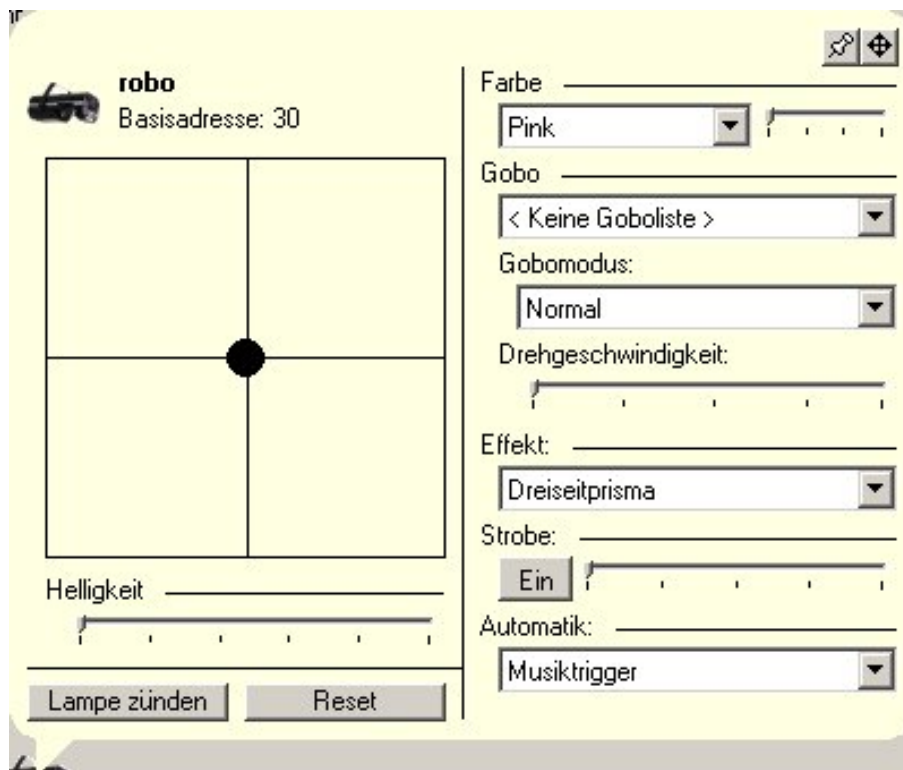
6 Appendix

6.1 Beispiel für die mögliche Komplexität von Kontextmenüs

Zusätzlich zu den bisher verwendeten Forms-Elementen sehen Sie in diesem Beispiel (Martin Roboscan 518 Pro) folgende Elemente:

- Linie
- Koordinatenkreuz für Scannerpositionierung

Das zugehörige xml-file finden Sie im device-Verzeichnis und auf unserer Webseite.



6.2 XML Schema für DMXControl

```

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      XML Schema for DMXControl
      PopSoft
    </xs:documentation>
  </xs:annotation>
<!-- Device===== -->
  <xs:element name="device">
    <xs:annotation>
      <xs:documentation>description of device and form
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="information" type="information_type"
minOccurs="1" maxOccurs="1"/>
        <xs:element name="panarea" type="degoff" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="tiltarea" type="degoff" minOccurs="0"
maxOccurs="1"/>
        <xs:element ref="channels" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="form" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="code" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="image" type="xs:string" use="required"/>
      <xs:attribute name="initsequence" type="xs:string"/>
    </xs:complexType>
  </xs:element>
<!-- Types ===== -->
  <xs:complexType name="degoff">
    <xs:attributeGroup ref="degoff_ag"/>
  </xs:complexType>
  <xs:attributeGroup name="degoff_ag">
    <xs:attribute name="degree" type="xs:positiveInteger"/>
    <xs:attribute name="offset" type="xs:positiveInteger"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="form_point">
    <xs:attribute name="top" type="xs:positiveInteger"/>
    <xs:attribute name="left" type="xs:positiveInteger"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="form_el_size">
    <xs:attribute name="width" type="xs:positiveInteger"/>
    <xs:attribute name="height" type="xs:positiveInteger"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="form_area">
    <xs:attributeGroup ref="form_point"/>
    <xs:attributeGroup ref="form_el_size"/>
  </xs:attributeGroup>
  <xs:simpleType name="yes_no">
    <xs:restriction base="xs:string">
      <xs:enumeration value="yes"/>
      <xs:enumeration value="no"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="device_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="dimmer"/>
      <xs:enumeration value="color"/>
      <xs:enumeration value="gobo"/>
      <xs:enumeration value="pan"/>
      <xs:enumeration value="panfine"/>
      <xs:enumeration value="tilt"/>
      <xs:enumeration value="pantilt"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="dummy">
    <xs:complexContent>
      <xs:restriction base="xs:anyType"/>
    </xs:complexContent>
  </xs:complexType>
<!-- Channels ===== -->

```

```

<xs:complexType name="function_t">
  <xs:attribute name="channel" type="xs:nonNegativeInteger"/>
  <xs:attribute name="minvalue" type="xs:nonNegativeInteger" default="0"/>
  <xs:attribute name="maxvalue" type="xs:nonNegativeInteger" default="256"/>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="fade" type="yes_no" default="yes"/>
  <xs:attribute name="action" type="xs:string"/>
  <xs:attribute name="type" type="device_type" use="required"/>
</xs:complexType>
<xs:element name="channels">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="function" type="function_t" minOccurs="0"
maxOccurs="256"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- Type information ===== -->
<xs:complexType name="information_type">
  <xs:sequence>
    <xs:element name="vendor" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="deviceidentifier" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="author" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string"/>
</xs:complexType>
<!-- Type Form element===== -->
<xs:complexType name="form_element">
  <xs:attributeGroup ref="form_area"/>
</xs:complexType>
<!-- Form ===== -->
<xs:element name="form">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="deviceimage" type="form_element" minOccurs="1"
maxOccurs="1"/>
      <xs:element name="devicename" type="form_element" minOccurs="1"
maxOccurs="1"/>
      <xs:element name="deviceadress" type="form_element" minOccurs="1"
maxOccurs="1"/>
      <xs:element name="position" type="form_element" minOccurs="0"
maxOccurs="1"/>
      <xs:element ref="slider" minOccurs="0"/>
      <xs:element ref="dropdown" minOccurs="0"/>
      <xs:element ref="onoff" minOccurs="0"/>
      <xs:element ref="command" minOccurs="0"/>
      <xs:element ref="options" minOccurs="0"/>
      <xs:element ref="label" minOccurs="0"/>
      <xs:element ref="line" minOccurs="0"/>
    </xs:sequence>
    <xs:attributeGroup ref="form_el_size"/>
  </xs:complexType>
</xs:element>
<!-- Slider ===== -->
<xs:element name="slider">
  <xs:complexType>
    <xs:attributeGroup ref="form_area"/>
    <xs:attribute name="channel" type="xs:nonNegativeInteger"/>
    <xs:attribute name="action" type="xs:string"/>
    <xs:attribute name="startvalue" type="xs:nonNegativeInteger"
default="0"/>
    <xs:attribute name="endvalue" type="xs:nonNegativeInteger"/>
    <xs:attribute name="tickfreq" type="xs:positiveInteger"/>
    <xs:attribute name="smallchange" type="xs:positiveInteger"/>
    <xs:attribute name="largechange" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
<!-- OnOff ===== -->
<xs:element name="onoff">
  <xs:complexType>
    <xs:attributeGroup ref="form_area"/>
    <xs:attribute name="caption" type="xs:string"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="action" type="xs:string"/>
    <xs:attribute name="channel" type="xs:positiveInteger"/>
    <xs:attribute name="onvalue" type="xs:nonNegativeInteger"/>

```

```

        <xs:attribute name="offvalue" type="xs:nonNegativeInteger"/>
        <xs:attribute name="onsequence" type="xs:string"/>
        <xs:attribute name="offsequence" type="xs:string"/>
    </xs:complexType>
</xs:element>
<!-- Dropdown ===== -->
    <xs:element name="dropdown">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="item" minOccurs="1">
                    <xs:annotation>
                        <xs:documentation>
                            elements of dropdown
                        </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:attribute name="caption" type="xs:string"/>
                        <xs:attribute name="value"
type="xs:nonNegativeInteger"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="form_area"/>
            <xs:attribute name="name" type="xs:string"/>
            <xs:attribute name="action" type="xs:string"/>
        </xs:complexType>
    </xs:element>
<!-- Options ===== -->
    <xs:element name="options">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="option" minOccurs="1">
                    <xs:annotation>
                        <xs:documentation>
                            elements of options
                        </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:attribute name="value"
type="xs:nonNegativeInteger"/>
                        <xs:attribute name="caption" type="xs:string"/>
                        <xs:attributeGroup ref="form_point"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="form_area"/>
            <xs:attribute name="channel" type="xs:nonNegativeInteger"/>
            <xs:attribute name="action" type="xs:string"/>
        </xs:complexType>
    </xs:element>
<!-- Command ===== -->
    <xs:element name="command">
        <xs:complexType>
            <xs:attributeGroup ref="form_area"/>
            <xs:attribute name="caption" type="xs:string"/>
            <xs:attribute name="clicksequence" type="xs:string"/>
            <xs:attribute name="downsequence" type="xs:string"/>
            <xs:attribute name="upsequence" type="xs:string"/>
            <xs:attribute name="channel" type="xs:nonNegativeInteger"/>
        </xs:complexType>
    </xs:element>
<!-- Label ===== -->
    <xs:element name="label">
        <xs:annotation>
            <xs:documentation>
                layout element
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:attributeGroup ref="form_point"/>
            <xs:attribute name="caption" type="xs:string">
                <xs:annotation>
                    <xs:documentation>
                        text of label
                    </xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:complexType>

```

```

    </xs:element>
<!-- Line ===== -->
  <xs:element name="line">
    <xs:annotation>
      <xs:documentation>
        layout element
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="x1" type="xs:nonNegativeInteger"/>
      <xs:attribute name="y1" type="xs:nonNegativeInteger"/>
      <xs:attribute name="x2" type="xs:nonNegativeInteger"/>
      <xs:attribute name="y2" type="xs:nonNegativeInteger"/>
    </xs:complexType>
  </xs:element>
<!-- Code ===== -->
  <xs:element name="code">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="procedure" minOccurs="0">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

6.3 Device-Beschreibung aus dem Tutorial

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<device image="moon.gif" initsequence="set 0 10;set 1 50">
  <information>
    <name>EUROLITE Gobo Changer TG-3) von Theo Tutorial </name>
  </information>
  <channels>
    <function channel="0" minvalue="0" maxvalue="255" name="Farbe" fade="no"
colorchannel="yes" action="GetColor" />
    <function channel="1" minvalue="0" maxvalue="255" name="Gobo" fade="no"
gobochannel="yes" action="GetGobo"/>
  </channels>

  <form width="177" height="134">
    <deviceimage top="0" left="0"/>
    <devicename top="0" left="40"/>
    <deviceaddress top="16" left="40"/>

    <label top="33" left="0" caption="Farbe"/>
    <dropdown top="53" left="0" width="113" name="color_color" channel="0" >
      <item caption="white" value="10"/>
      <item caption="red" value="20"/>
      <item caption="dark blue" value="40"/>
      <item caption="green" value="50"/>
      <item caption="yellow" value="70"/>
      <item caption="orange" value="90"/>
      <item caption="pink" value="105"/>
      <item caption="light blue" value="120"/>
    </dropdown>

    <label top="80" left="0" caption="Gobo"/>

    <dropdown top="103" left="0" width="113" name="gobo_gobo"
action="SetSunColor">
      <item caption="Schnipsel" value="155"/>

```

```

    <item caption="Sternenhimmel" value="165"/>
    <item caption="Sonne" value="170"/>
    <item caption="Torten-Dreiecke" value="180"/>
    <item caption="Dreieck" value="188"/>
    <item caption="8-flügliges Windrad" value="195"/>
    <item caption="5-eckiger Stern" value="205"/>
    <item caption="dreigeteilter Kreisring" value="210"/>
    <item caption="4-flügliges Schaufelrad" value="220"/>
    <item caption="4 Quadrate" value="235"/>
    <item caption="Open" value="140"/>
    <item caption="closed" value="130"/>
</dropdown>

<label top="33" left="133" caption="Off:"/>

<options channel="1" top="50" left="133">
  <option value="0" caption="off" left="0" top="0"/>
  <option value="155" caption="on" left="0" top="250"/>
</options>

<onoff caption="Strobe" top="97" left="133" width="41" channel="1"
onsequence="set 1 110" offsequence="set 1 140" />

<command top="120" left="133" width="41" caption="Reset"
clicksequence="save 1;set 1 134;hold 6000;set 0 10; set 1 10"/>
</form>

<code>

<procedure name='INITCONTROLS'>
  !call|GetColor$
  !call|GetGobo$
</procedure>

<procedure name="SetSunColor">

  !set_channel|1|({gobo_gobo})$
  !if|(({gobo_gobo} &gt; 167) and ({gobo_gobo} &lt; 176))|
    !set_channel|0|70$
  $
</procedure>

<procedure name='GetColor'>
  !set_control|color_color|({channel_0})$
  !if|{channel_0} &gt; 127|
    !set_control|color_color|(70)$
  $
</procedure>

<procedure name='GetGobo'>
  !set_control|gobo_gobo|({channel_1})$
  !if|{channel_1} &lt; 136|
    !set_control|gobo_gobo|(199)$
  $
</procedure>

</code>

</device>

```