

DMXControl

Language Description and Tutorial for Device- and Forms Configuration

Version 2.8.1

Status: beta



PopSoft

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

Authors:

Frank Burghardt, Berlin
Stefan Krupop, Braunschweig
e-mail: info@dmxcontrol.de

The document was translated with friendly aid by <http://ets.freetranslation.com/> ;-)

Content

1	INTRODUCTION	2
1.1	Conditions of utilization	2
1.2	System requirements	2
2	Language description	2
2.1	General	2
2.2	Example - Dimmed spots	3
2.3	Tasks	4
2.4	Overview	5
2.4.1	Graphic Elements	5
2.4.2	Functional Elements	7
2.5	Syntax	8
2.5.1	Generic Attributes	8
2.5.2	Description of devices	9
2.5.3	Description of channels	9
2.5.4	Description of menus	10
2.5.5	Control elements	11
2.5.6	Passive layout elements	13
2.5.7	Hilfe	14
2.5.8	Sequences	14
3	Extended Programming	15
3.1	General principles	15
3.2	Elementary language elements and conventions	15
3.3	XML Syntax	17
3.4	A Procedure for Example	17
4	Tutorial	19
4.1	Step 1: Studying the operation instructions	19
4.2	Step 2: Determine the Functionality of Form	19
4.3	Step 3: Generation of the XML file	20
4.3.1	Description of Device	20
4.3.2	Description of the Channels	20
4.3.3	Description of the Forms	21
4.3.4	Further controls	23
	Step 4: Advanced Programming	26
5	Error catalogue	28
5.1	Known Problems	28
6	Appendix	29
6.1	Example for the possible complexity of device forms	29
6.2	XML Schema for DMXControl	30
6.3	Device Description from Tutorial	33

Please ask for spezial help for creation of device definition files fro your equipment:

Daniel Miertz (DDF support)

e-mail: ddf-support@dmxcontrol.de

Version info	new	
Rel. 2.8	Help tag Color picker	- e.g. for DMX table of device - RGB control

1 INTRODUCTION

DMXControl is a tool for the arrangement of your light show.

With this program you can steer your DMX capable equipment of the computer and need only minimal knowledge about the DMX concept.

This tutorial deals with the construction of equipment descriptions for DMX equipment for the integration in DMXControl.

It is described how specific menus can be assigned to this equipment also.

1.1 *Conditions of utilization*

DMXControl is freeware. The passing on of the program and the documentations without changes is permitted.

We ask all users to register themselves for purposes of feedback in the DMXControl forum free of charge. The authors receive no responsibility for possible damages which arise from the use of the software.

1.2 *System requirements*

For the production of configuration and form-descriptions you need only a text editor in principle. A view of the files in the Internet Explorer can be advisable.

Configuration and form descriptions are text files with the extension ".xml". Of course, these files can also be provided by known XML tools. The XML schema for DMXControl is contained for information in the appendix. If you want to use it for your XML editor, find it also as a xds file in our download area.

2 Language description

2.1 *General*

The description of DMX equipment is carried out at DMXControl in a XML format which is transformed by a Parser into the internal equipment description.

DMXControl doesn't support any graphic equipment editor at the moment yet.

This document therefore gives instructions for the manual construction of equipment descriptions by means of a normal editor or XML tools.

XML is very common language syntax which e.g. is used in other dialects at web pages or WAP sides in the Internet.

But you don't have to be an expert to make a new equipment description.

The following simple notes suffice:

- You should take care that your editor doesn't save any (invisible) control characters. There won't be any problems at the Windows editor or emacs. Please save the file in the text format, if you want to use Word or like tool.
- The DMXControl dialect of XML demands to each opening tag („<tag>“) always a closing tag („</tag>“). Only the tag of the last level are implicitly finalized (e. g. <item caption= "white" value= "0" />)
- The closing separators „/>“ and </tag> are semantically equivalent, i.e. also <item caption= "white" value= "0" > </item> is valid.
- Each tag can possess attributes, that you can find in the table below. Each parameter value is initiated with an equals sign and the value must always be included in double apostrophes.
- All day and attribute identifiers are -- noted down at DMXControl in a small letter.
- The tags are hierarchically defined. You should make that visible by means of corresponding indents.
- As a rule, the order of the attributes of a tag doesn't care. Where it for once depends on the order anyway we describe it below explicitly.
- Comment lines are noted in the form <! -This is a comment -->
- You can have a look at xml files (therefore also the DMX DEVICE files) in clear view with the Internet Explorer.

So this was it already, we should have a look at a simple example.

2.2 Example - Dimmed spots

Every device should get an appropriate icon to receive an adapted representation in the stage representation. This icon is put down to images in the subdirectory.

Note:

The icons still can be changed in DMXC afterwards (context menu stage).



Figure 1: Icon for spots equipment

The placed numbers before lines of the following XML-files don't have to be inserted; they serve only for description of the example:

```

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      <deviceaddress 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>

```

Line 1 announces the parser the related XML-version. Line 2 refers to the global description of the new device type inclusive the icon to be used and an initial signal sequence of the device (the optional initsequence was demonstrated here, but is only limited reasonable).

Line 4 as a part of the channel description assigns a name to the specified device channel and defines the allowed scope of DMX-values. Hence, the channel numbers must be given always by 0 climbing up without interruption for all channels. The lines 6-9 describe the form, its graphical coordinates in the context menu in pixel for the device picture, name and start address in the unit "pixel".



Figure 2: Context menu of dimmed floodlight

After that the controls with its properties follow, in this example a slider with graphic coordinates. For other devices types, also controls such as radio buttons, dropdowns or normal buttons could be declared here.

The buttons at the right top corner (Pin-button and Move-button) are generated by default.

2.3 Tasks

Before you define your equipment type, please have a look on our web page whether there is already a suitable or similar definition for your device. In order to define a new device, you can modify an existing file of a similar device.

Of course the authors don't have access to all DMX equipment of the various manufacturers. If you should have defined a new device, we would be happy if you would make available the result to the general public world, in that you send us the corresponding file (info@dmxcontrol.de or use the upload-menu of our web page later).

If you have completed your work, you simply save the xml file in the "devices" subdirectory in your DMX installation and store the corresponding gif picture (in the format 32 x32 pixels, the transparency fashion is favourable) in the subdirectory "images" - and the device should be visible at the next start of DMXControl.

2.4 Overview

Every description contains four main parts (see illustration 1):

- description of global properties (in the above example line 1-2)
- description of the individual DMX channels (in the example line 3-5),
- description of the graphic context menu (in the example line 6-11),
- procedure code (optional, not contained in the example):

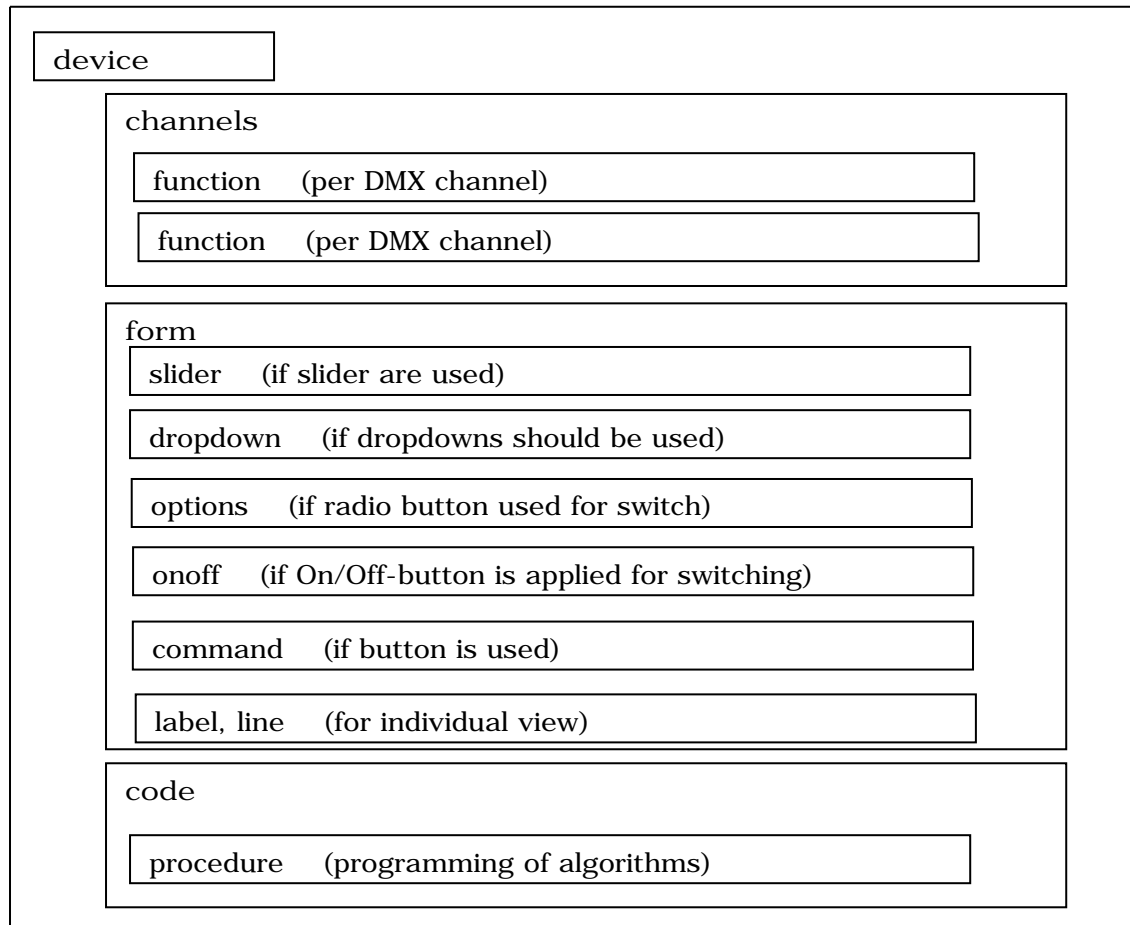


Figure 3: Parts of des Equipment and Forms Configuration file

All form elements must be provided with graphic coordinates to the positioning in the context menu. The elements "label" and "line" are passive and serve only the design of the surface. The remaining active elements of the form (also called control elements) can be used for the active control of the DMX device, e.g. onoff.

2.4.1 Graphic Elements

A survey of all supported graphic elements shows illustration 4.

An assignment to the required syntax elements is in carried out the explanation texts.

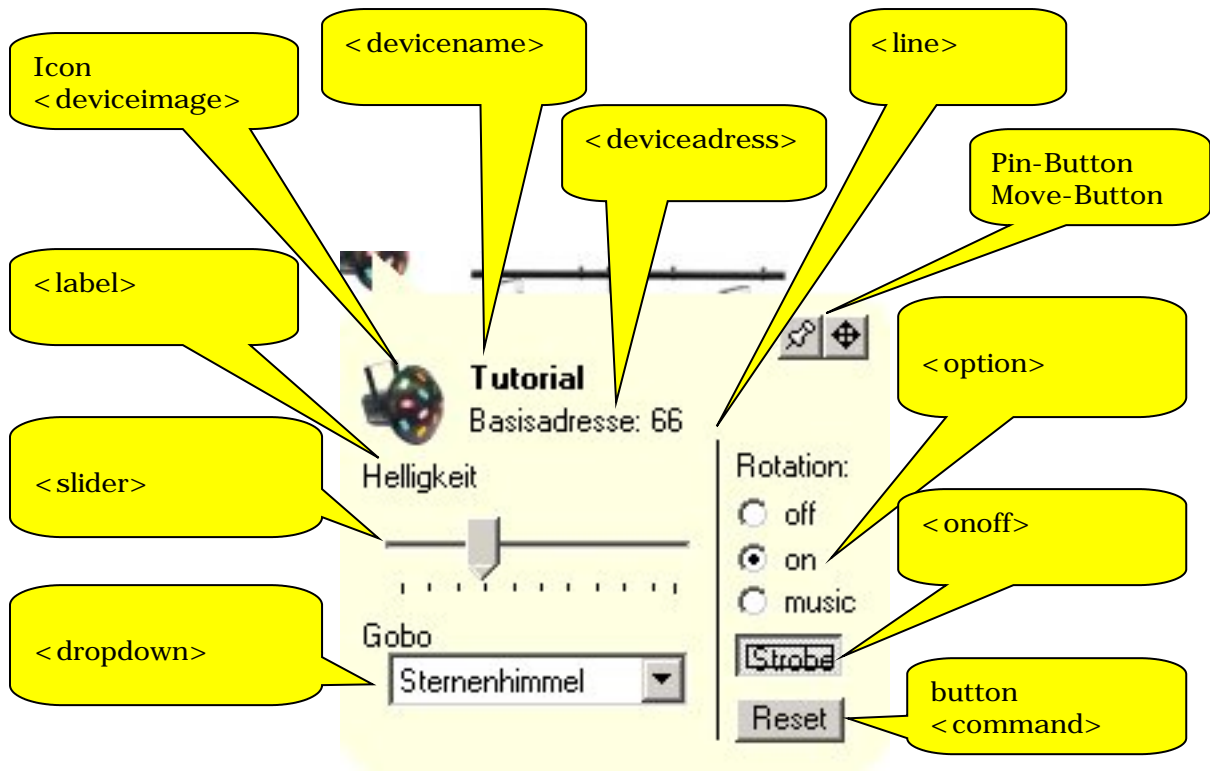


Figure 4: All supported graphic elements

The following illustration illustrates the meaning of the graphic coordinates:

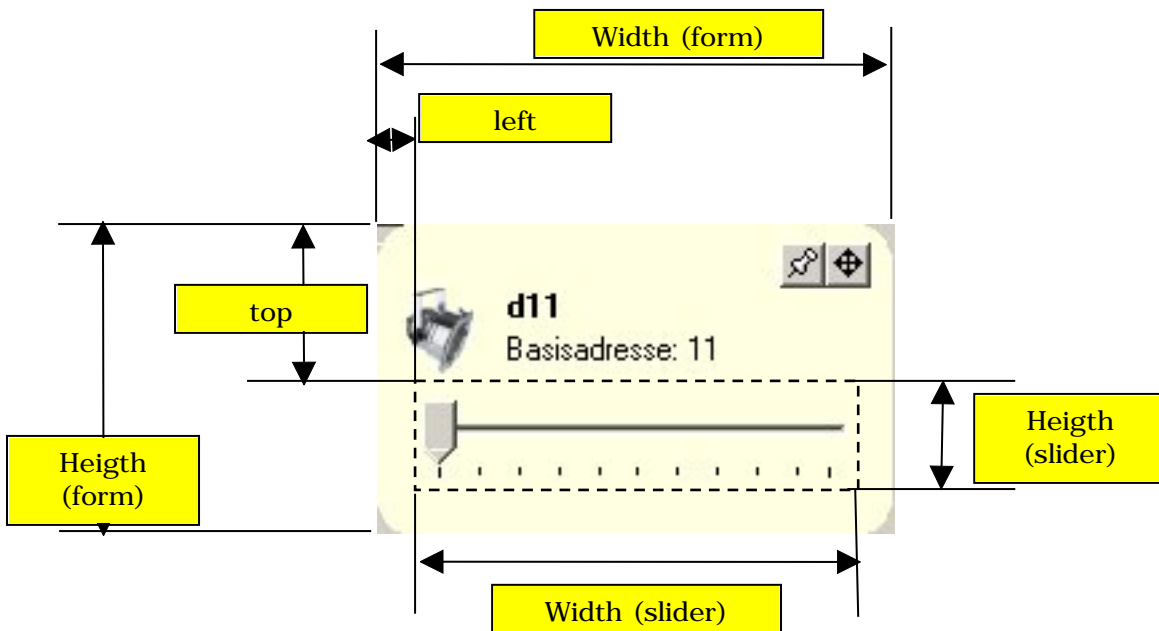


Figure 5: graphic coordinates

2.4.2 Functional Elements

The active control elements interoperate in 3 possible ways with DMXControl or the device and with the definition of a concrete control exactly one concept is selected:

	concept	description	example
1	Channel/value concept	By the channel parameter the control is assigned to a concrete channel. In the control element concrete values or scopes of values are defined which are put with the activity of the control element.	- option list caption= "Star sky" value= "165" - scope of slider startvalue= "0" endvalue= "255"
2	Sequences	An instruction set is defined in the control as string which is executed while operating the control.	clicksequence= "set 1 75; set 2 100"
3	Action/procedures	A separately defined procedure contains more complicated program instructions. The procedure is assigned about the action parameter and is executed with every operating of the control.	action= "SetGobo"

This particularly means, you have to use the channel attribute and the action attribute in the control elements alternatively.

Please take into account: For many applications the first two concepts are sufficient. As well as the complexity of the 3 concepts grows, the processing time requirements also increase. Hence, the simplest variant for the solution of a task should always be chosen.

An action attribute can be used also with the channel description. This means that the procedure is called by every change of the channel value.

An example is shown in illustration 6 how different control elements can interact with the channels. The channel values are addressed about "channel_n" and can be set to over the 3 mentioned above concepts. The current value which is assigned to a control element is addressed by a name assigned to the control element that is freely eligible (reference), marked by "control_n" here.

In the example "actions" are defined for three control elements and additionally "reference names" are assigned. These actions can set one or several channel values and therefore should be called "Setnnnnn", as a rule. Furthermore an "action" is assigned to the channel_3. This is intended to read the current channel values and to inform the control elements about to. Such a procedure will therefore get the name "Getnnnnn", as a rule, since it must read current channel values.

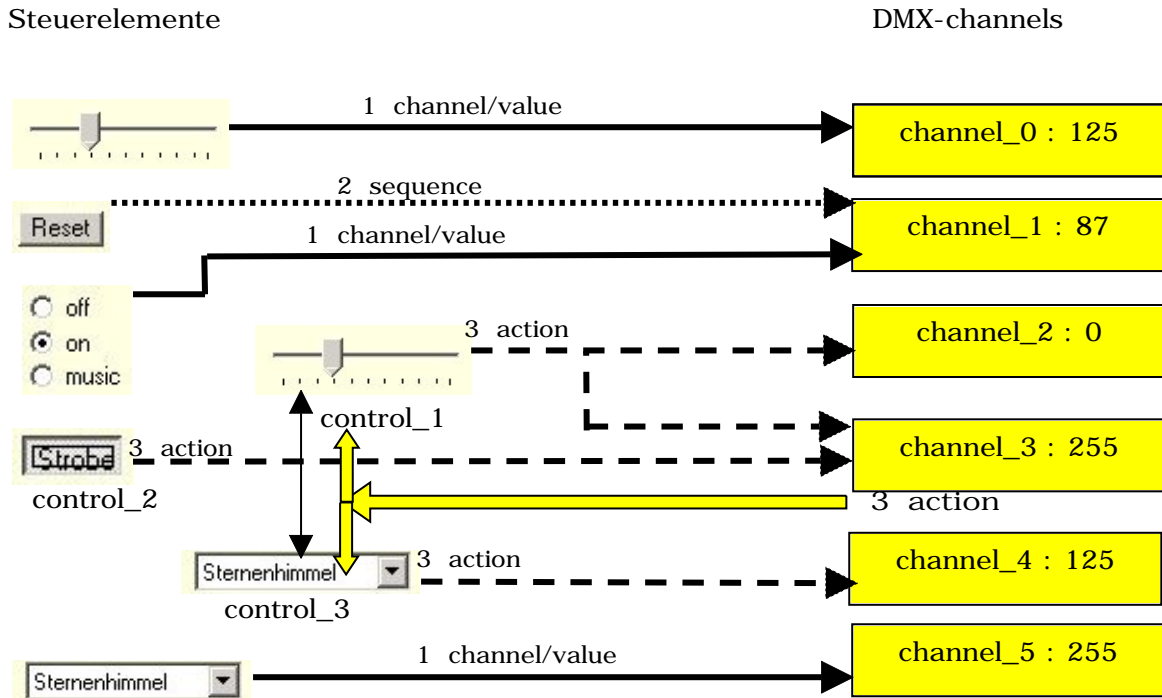


Figure 6: Interaction of control elements

Tip: the inscriptions of the badges have here no importance.

2.5 Syntax

This chapter describes the syntax of the device and form configuration files of DMXControl. A XML scheme is given in the appendix.

2.5.1 Generic Attributes

To the shortening of the following tables, often repeating parameters with same meaning are described once globally here.

reference	Attribute	hierachy and property	comment	example
GP1	value	assigned DMX-value		0, 128, 255
GP2	caption	Describing enum value for explaining a scene		green Star
GP3	top	Relative y-coordinate of the elementes from left upper corner		
GP4	left	Relative x-coordinate of the elementes from left upper corner		
GP5	width	Width of element		
GP6	hight	Hight of Element		

Coordinates are given in pixel

2.5.2 Description of devices

tag	Attribute	hierachy and property	comment	example
<device>		Level 1		
	image	File name of icon	String, Filename .gif	Moon.gif
	initsequence	set initial values for individual DMX channels of the equipment	Optional Use it e.g. for scanner start position	set 0 15; set 7 128
<information>		Level 2 Additional informal comment		
<name>		Level 3 Arbitrary text		Custom Scanner
<vendor>		Level 3 Arbitrary text only for information	optional	ShowTec
<deviceidentifier>		Level 3 Arbitrary text only for information	optional	TG-3
<author>		Level 3 Arbitrary text only for information	optional	T.Tutor
<panarea>		Scanner spezific Information	optional	
	degree	Degree of illumination area	optional	85
	offset	Tbd.	optional	0
<tiltarea>		Scanner spezific Information	optional	
	degree	Degree of illumination area	optional	176
	offset	Tbd.	optional	0
<help>		Level 2 Help text (ASCII)	optional	

2.5.3 Description of channels



Tag	Attribute	hierachy and property	comment	example
<channels>		Per DMX-channel one "function" element must be defined		
<function>		Level 3 Subtag of „channels“		
	channel	Internal DMX channel-number the channel numbers must be given always by 0 climbing up without interruption for the single channels	integer	
	minvalue	Minimal DMX value	integer	
	maxvalue	Maximal DMX value	integer	
	name	describing name for	string	Brightnes

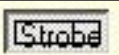
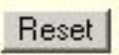
		channel		s
	fade	Fadeable property of channel	string	no, yes
	type	type of channel	string	dimmer, color, gobo, pan, panfine, tilt, tiltfine
	action	Call of code (procedure); Is called, if the channel value is changed	optional string	


2.5.4 Description of menus

Tag	Attribute	hierachy and property	comment	example
<form>		Level 2		
	width height	See generic attributes	integer	
<deviceimage>		Level 3; Subtag of „form“ relative position of shown image		
	top left width height	See generic attributes	integer	
<devicename>		Level 3 Subtag of „form“; relative position of shown name		
	top left width height	See generic attributes	integer	
<deviceaddress>		Level 3 Subtag of „form“; relative position of shown Basic address		
	top left width height	See generic attributes	integer	
<position>		Coordinate system for Moving Lights		
	top left height width	See generic attributes Tip: make start position with initsequence	integer	

2.5.5 Control elements

Tag	Attribute	Hierarchy and property	comment	example
				
< slider>		Level 3 Subtag of „form“; Creates a slider		
	top left height width	See generic parameters	integer	
	channel	Assigned channel Alternatively to action	optional	
	startvalue	Lower value	integer	0
	endvalue	Upper value	integer	255
	tickfreq	distance of the scales partitioning	integer	50
	smallchange	Change installment, e.g., with the arrow keys	integer; VB- Eigenschaft	20
	largechange	Change installment, e.g., with the mouse click (do not pull)	Integer; VB- Eigenschaft	50
	name	reference name for procedure code (variable)	Optional string	
	action	procedure call at changing the slider	optional	SetColor
				
< drop down>		Level 3 Subtag of „form“; Erzeugt Klappmenü		
	top	see GP3	integer	
	left	see GP4	integer	
	width	see GP5	integer	
	channel	Assigned channel Alternatively to action	optional integer	
	name	Reference name for procedure code (variable)		
	action	procedure call at changing the dropdown element		
< item>		Level 4 Subtag of „dropdown“		
	caption	Describing name		
	value	Assigned value		
< colorlist>		inserts all entries of the color list assigned to the device	optional replacement for item	
< gobolist>		inserts all entries of the gobo list assigned to the device	optional replacement for item	

		<input type="radio"/> off <input checked="" type="radio"/> on <input type="radio"/> music		
< options >		Level 3 creates Radiobox Subtag of „form“;		
	top left	see generic attributes	integer	
	channel	Assigned channel	integer	
	action	procedure call at changing the radiobox element		
< option>		Level 4 Subtag of „options“		
	caption	see GP2	string	
	value top left	see generic attributes	integer	
				
< onoff>		Level 3 Subtag of „form“; creates On/Off –Button (State is kept)		
	top left width height	see generic attributes		
	caption	See GP2		
	name	Reference nname for procedure code		
	channel	Assigned channel Alternatively to action	integer	
	onvalue	Value for ON	integer	
	offvalue	Value for OFF	integer	
	onsequence	Sequence, to be started for ON	string	
	offsequence	Sequence, to be started for OFF	string	
	action	procedure call at changing the on/off element		SetFog
				
< command>		Level 3 Subtag of „form“ Defines a new button		
	top left width	Position see generic attributes		
	caption	GP2 Button label		"Lampe zünden"
	clicksequence	Sequence to be started at click		"save 0;set 0 230;hold 5500;res tore 0"

	downsequence			Dto.
	upsequence	Sequence to be started at up button		Dto.
	action	procedure call at clicking the button	string	
				
< color picker >		Color mixer for RGB-devices		
	mode	"rgb" or "cym"	string	
	channel1	Assignment to 1. channel	integer	0
	channel2	Assignment to 2. channel	integer	1
	channel3	Assignment to 3. channel	integer	2
	layout	Choice between „1“ or „2“ (two Layout possibilities)	integer	1

Example:

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

Please hold down the mouse key, if you want to change the color via the color bar of dropdown menu. You can also enter the numeric color value directly.

2.5.6 Passive layout elements

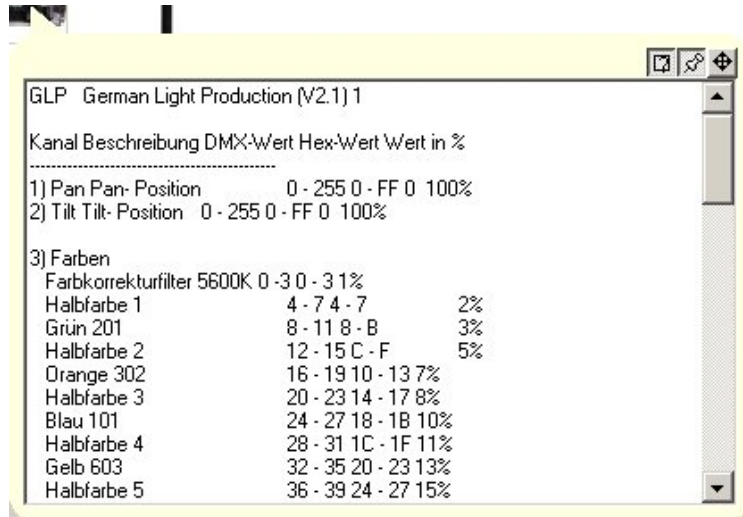
Tag	Attribut	Hierachy and property	comment	example
< line >		Level 3 Subtag of „form“; Layout element; Creates line with refered coordinates		
	x1 y1 x2 y2	coordinats	integer	
< label >		Level 3 Subtag of „form“; Layout element; Additional label)e.g. for controls)		
	top left	position, see GP3 and GP4	integer	
	caption	Text of label		Brightness

2.5.7 Hilfe

Following figure shows an example for help menu. Help menus are optional, but we recommended to display the DMX table of the device here.

The help is described in ASCII format only. Please improve the layout by using underlines and tabulator signs.

Help text can be switched on and off by clicking the questionare button (left from pin button)



2.5.8 Sequences

Following sequence operations are supported (all parameters of type integer):

Operation	Bedeutung
save >channel<	Saves current value of DMX-channel >channel<
set >channel< >value<	Set channel >channel< to value >value<
hold >time<	Timer (holds >time< in msec.)
restore >channel<	restores the internal saved value of DMX-channel >channel<

3 Extended Programming

The Device and form configuration of DMXControl also permits the programming of algorithmic changes of the DMX signals, that are executed automatically with the operation of control elements like on/off badge, dropdown or slider (see actions in chapter 2.4.2.) .

Therefore you can set the values which are assigned to the control elements via logical conditions and formulas. The procedures serve to be able to treat multiple occupations of a channel (e.g. Gobo rotation lies on the same channel as the Gobo selection) so that the value can be calculated differently depending on rotational speed.

These procedures also permit you the programming of dependences of the channels or control elements of a device, e.g. you connect certain colors with Gobos in algorithmic assignment or you organize colors to certain speeds.

3.1 General principles

The procedures are assigned by action attribute to the channels in <function> tags or to control elements <onoff>, <slider> or <dropdown>.

With operation of corresponding control elements the code of the procedure is interpreted and executed automatically.

If, on the other hand, the action is assigned to a channel (function), then it is executed at every change of the belonging channel value.

The procedures use "references" into the values of the control elements which are declared by the "name" attribute within the form definition of the control elements, e.g.

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

This means: The procedure SetColor is executed with operation of the dropdown menu and the value of the dropdown menu to be changed is addressed as "color_color".

Furthermore implicit references as a default assignment to the channels exist.

So, "channel_2" is implicitly related with the channel (function) with channel = 2, the action "GetColor" is also assigned to.

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

Till now, variables in the real meaning as freely verifiable values aren't defined.

3.2 Elementary language elements and conventions

The references are marked by their name ("color_color"), and the current value of the assigned control element is addressed by curly brackets ("{ color_color}").

To simplify the interpretation of the procedure code by a parser, some conventions are agreed you unfortunately must observe although they don't seem very user kindly in the present program release. For comparison purposes and to the ease of learning the following tables always deliver a generic language variant at the right side.

Following characters serve as a separators:

character	meaning
!	Starts an operation
	separates the parts of an operation
\$	Closes an operation

Following arithmetic operations are supported:

Operation	meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
mod	Modulo-Operation

Every expression must be surrounded by a bracket to be calculated.
The following examples illustrate the use of arithmetical operations:

DMXControl code	Generic language
!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

The defined standard operators have following semantics:

Operation	meaning
set_control >cntr< >val<	Set the referred control >cntr< to the value of expression >val<
set_channel >ch< >val<	Set the referred channel >ch< to the value of expression >val<

Following comparison operators can be used:

Operation	Alternative presentation	meaning
>	> gt	greater
<	< lt	lower
=	&eq; eq	equals

By means of the comparison operations, conditions can be formulated because DMXControl supports the if-direction.

DMXControl code	Generic language
!if {channel_3} < 1	if channel_3 < 1 then

The procedure "INITCONTROLS" is executed at opening the form, due to setting the controls on the values which are given by the current DMX values. Only "Get" functions should be called, only read! Initcontrols must not be declared, however it is recommended, in order the controls should always illustrate the current state of the device.

3.3 XML Syntax

Tag	Attribute	Hierarchy and property	comment	example
<code>		Level 2 Subtag of „device“; Contains all procedures		
<procedure>		Defines code for a action		
	name	Name of procedure	string	

3.4 A Procedure for Example

This example shows a procedure which puts the values of the control elements "color_color" and "color_speed" of a Moving Heads into dependence of the current DMX value of the channel 2 (color change machine).

The Mac of 250+ has 4 possibilities for the color wheel:

- celebrations color (first If in the example below)
- turn clockwise (2nd If in the example below)
- turn anticlockwise (3rd If in the example below) as well as
- coincidental colors with different speed (4th If).
-

Here the declarations in the configuration file, the relevant variables are stressed fatly:

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

“color_color” is the reference to the dropdown control for color selection.

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

“color_speed” is the reference to the slider control for speed selection.

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

The procedure subdivides the DMX_values of channel 2 in 4 intervals and executes several statements. The current value gets therefore checked and the Controls get appropriately adapted.

DMXControl code	Generic language
<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)\$ \$ \$ </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; ; </pre>
</procedure>	end

4 Tutorial

This Tutorial demonstrates the creation and practical usage of DMX configuration files on example of the Gobo change machine TG-3.

4.1 Step 1: Studying the operation instructions

The TG-3 uses 2 DMX-channels. In the description, you find following DMX-information:

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: snippet
128-255	white	160-167	Gobo 2: stars sky
		168-175	Gobo 3: sun
		176-183	Gobo 4: pies-triangles
		184-191	Gobo 5: triangle
		192-199	Gobo 6: Wind Wheel with 8 leafs
		200-207	Gobo 7: 5-angular star
		208-215	Gobo 8: three part circle ring
		216-231	Gobo 9: shovel wheel with 4 leafs
		232-255	Gobo 10: 4 squares multicolor

4.2 Step 2: Determine the Functionality of Form

For this device it is an obvious idea to lay out one dropdown menu each for the color and for the Gobos.

Furthermore there shall be a On/Off and a push button to start and stop the stroboscope effect. In all cases we also lay out a reset badge. This becomes symbolizes corresponding form with the following outline:

ima
ge

Gobo Changer

Basic address

color
 off

Dropdown color

 on

Gobo

Strobe

dropdown Gobo

reset

Figure 7: outline for TG-3 context menu

4.3 Step 3: Generation of the XML file

Advantageous manner again an existing XML-filed is reused. We begin with the device-description. The lines numbering serves only the explanation of the example.

4.3.1 Description of Device

We need the line 1 with the XML version and the <device> tag by default.

Either a new icon is laid out in the pixel of format 32 x32 in image subdirectory of the devices-directory or one checks the reusability of the existing pictures.

In our case the picture resembles moon.gif strongly our TG 3, so that we decide in favor of it.

As a precaution becomes an initialization of the device provided but still not fully filled out (line 2).

Within the <information> tag we describe the device (line 3-5) and immortalize ourselves as an author.

```

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 Channel Feature
     -----
     0-15 white
     16-31 red
     32-47 dark blue
     2. DMX Channel Feature
     -----
     0-5 Lamp off
     6-63 open
     64-127 Strobe-effect (5 flashes/second)
   </help>
x </device>

```

Now this file is saved first once under the name GoboChangerTG3.xml in the device-index.

4.3.2 Description of the Channels

This part begins with the <chanel>-tag. We have 2 channls and require therefore two <function>- tags. Of an existing scanner-device, we modify in addition the channels „color“ and „Gobo“.

```

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>

```

The color channel gets the internal channel address "0" and the Gobo channel the number "1". We see the minimal and maximum values for the DMX channels (0 and 255 each, lines 7 and 9) in the table above. Since we have to do it here with purely discreet

values, so fading doesn't make sense – we set it to "no". Both channels still must be assigned the type, what is self-explanatory (line 8 and 10).

The expanded file is stored again now. When starting DMXControl you can already recognize the TG 3 as an equipment now. One can already see the two defined channels in the channel assignment tool (for the facilities in the Submaster or at effects tool).

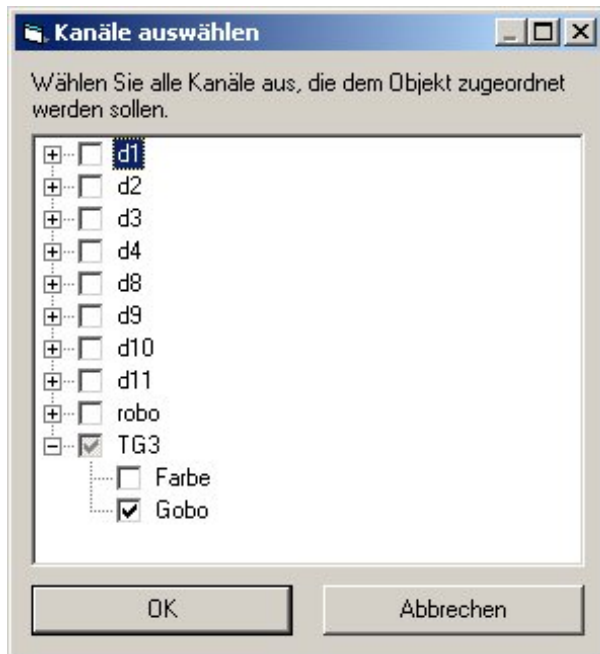


Figure 8: channel selection window

Now we can supplement also line 2 and can introduce the reset command as an init sequence. In addition we send a signal (white and open) on both channels.

```
2 <device image="moon.gif" initsequence="set 0 10; set 1 150" >
```

4.3.3 Description of the Forms

Until now we have defined all relevant channel properties and published to DMXControl. The following modifications are exclusively related to the user interface. These user interface properties are immediately available at re-opening the forms. There is no need to restart DMXControl during testing your forms from now.

At this test you also will have noticed it that the context menu was empty for the device and the operation is very circumstantial over the Submaster tool. Therefore we convert our menu outline in the form description fast.

To this we start with the <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>
```

We now already see the icon, the device name and the base address in the context menu now.

We define the dropdown menus in the next step. For the graphic coordinates we take into account the sizes in pixel. Both dropdown menus get a label (line 16 and 27).

Please, do not forget to define the channel attribute within the <dropdown> tag, so that DMXControl knows to which channel the value assignment refers.

The description of the opening menus can be derived from the table in the chapter 4.1 practically above (lines 18-25 and 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"/>
26 </dropdown>

27 <label top="80" left="0" caption="Gobo"/>
28   <dropdown top="103" left="0" width="113" name="gobo_gobo"
29     channel="1">
30     <item caption="Schnipsel" value="155"/>
31     <item caption="Sternenhimmel" value="165"/>
32     <item caption="Sonne" value="170"/>
33     <item caption="Torten-Dreiecke" value="180"/>
34     <item caption="Dreieck" value="188"/>
35     <item caption="8-flügliges Windrad" value="195"/>
36     <item caption="5-eckiger Stern" value="205"/>
37     <item caption="dreigeteilter Kreisring" value="210"/>
38     <item caption="4-flügliges Schaufelrad" value="220"/>
39     <item caption="4 Quadrate" value="235"/>
40     <item caption="Open" value="140"/>
41     <item caption="closed" value="130"/>
41 </dropdown>

```

Now the context menu looks already as follows:



Figure 9: Dropdown menu in context menu

You can already adjust now the color and the Gobo mode.

4.3.4 Further controls

Now we expand the menu around the remaining controls. We reach the turning off of the device with a radio button by the `<option>` tag. The essential information herewith is that channel 1 is assigned the value „0“ at clicking on the „off“ option (line 44 and 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>
```

We start the Strobe-effect with a button by using the `<onoff>` tag. The button is kept “pressed” until it is clicked again. In line 48 a value is assigned to each state.

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

Lastly the reset button is implemented. We have to decide for the `<command>` tag, because the reset activity should only be started by button click one time without keeping any status of control element.

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

Here the channel is addressed in the parameter `clicksequence` (line 50). In detail, following actions are carried out „on click“ :

- saving the current value of channel 1
- setting of the signal for the reset
- set validity of the value for 5500 msec
- at the end of this time origin value is restored again.

Now it is created. We generated following form, that completely transferred our design:



Figure 10: complete context menu of TG3

Here once again the complete XML-filed to the survey:

```

<?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"/>
<deviceadress 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>
```

Step 4: Advanced Programming

Our form already has become a beautiful thing, however, we can in principle set only static values in the two channels. Furthermore, there also are other disadvantages:

After changes in a strange element, e.g. Submaster or onoff badge not all new values are switched over into the dropdown menus automatically. In this chapter it is also shown how dependences between the control elements can be programmed.

We want to implement the following effects:

1. Only trying out how to set values with action instead of channel/value
2. the dropdown elements shall show the current value any time. (Note: They should anyway if the provided current value is defined as an item, this has therefore to be checked.)
3. If the Gobo "sun" is selected, it always shall be represented in "yellow" automatically color (because our GoboChanger is rather stupid in comparison with scanners we cannot provide any better example although every reader perhaps doesn't attach importance to this effect).

To do this we declare the actions "GetColor" and "GetGobo" which shall respectively read the current channel values.

Furthermore we need a reference value at the Gobo dropdown box (named as "gobo_gobo" and an action about which we want to put values of the DMX channels).

The color dropdown control doesn't get any action (it steers still about channel/value), however, we define a reference value for updating the dropdown value also here, so we call this control element "color_color."

```
<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" >
```

For the implementing the requirements we must define in the <code> part the procedure bodies to the "actions".

We start with the GET procedure. The lines 101 and 107 read only the current channel values actually and assign these to the control elements. The respectively following two lines shall demonstrate the intelligence of the procedure mechanism:

If we don't like the current values because they cannot be assigned to an appropriate list element, the values are "immediately manipulated" (a more complex expression turns on the "70" usually stands). One must check whether these values possibly have to be sent also on the channel.

```
100 <procedure name='GetColor'>
101 !set_control|color_color|({channel_0})$
102 !if|{channel_0} > 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} < 136|
109 !set_control|gobo_gobo|(199)$
110 $
111 </procedure>
```

In the procedure INITCONTROLS both Get procedures are called, so that by every opening of the pop-up menu the values are updated.

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

Now the procedure with which we put the Gobo value is still absent. Because it also should fulfil the 3rd requirement, it is called "SetSunColor".

In line 117 the value of the control element is simply sent the channel 1. In line 118 it is checked whether this value corresponds just to the "sun" Gobo. If yes, the color channel is informed of that the color wished is "yellow".

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

In the appendix you find once more the entire xml file for this "intelligent" GoboChanger.

We wish you much success at the construction of the equipment descriptions of your DMX equipment now.

And you please let us participate by Feedback in your success.

5 Error catalogue

5.1 *Known Problems*

Here well known problems of the parser and the form-configuration are supposed to be held.

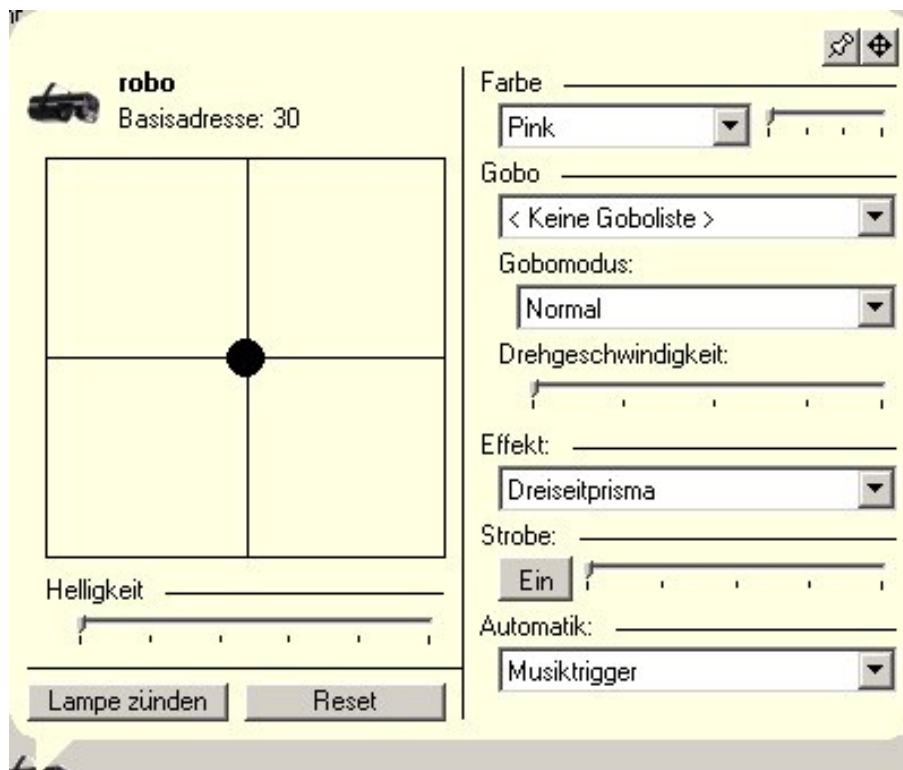
Problem/Error message	Possible reason

6 Appendix

6.1 Example for the possible complexity of device forms

In addition to the previously applied form elements you see following elements in this example (Martin Roboscan 518 Pro):

- Line
- coordinate cross for scanner positioning



6.2 XML Schema for 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 Description from 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"/>
    <deviceadress 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>

```