

Writing configuration files for the ELC sideKick Fader10

by ELC lighting

The sidekick fader10 is a fully programmable fader/button panel.
The way the sidekick works is determined by the configuration files loaded into the unit.

The best way to start is read this document once and then again, but then while looking into the configuration text files on the CD-rom.

The configuration files are a combination of text files, with the main file being config.txt.

```
#include = "universal.txt"  
#include = "grandma_DMX_Wing1.txt"  
#include = "grandma_DMX_Wing2.txt"  
#include = "grandma_DMX_MIDI.txt"  
#include = "grandma_OnPC.txt"  
#include = "hog2_DMX_Wing1.txt"  
#include = "hog2_DMX_Wing2.txt"  
#include = "hog2pc.txt"  
#include = "hog2_wing1.txt"  
#include = "hog2_wing2.txt"  
#include = "sunlite.txt"
```

Description of the configuration file basics

Each configuration file is a list lines. Each line is build up in elements.

[identifier] = [parameter 1], [parameter 2], [parameter 3] ,

The identifier describes the function of the line, and 1 or more parameters set the values.

Special identifiers are marked with a '#' at the front, like the #include lines in for example the config.txt.

Examples:

The first line would be an event line, describing the event to what to act to

```
event = button, 12, press
```

identifier *event* > the line marks that the line describes an event

parameter 1 *button* > the event is a button event

parameter 2 *12* > the button event belongs to button 12

parameter 3 *press* > the complete event describes a button 12 press

The line that would follow the event would be 1 or more action lines

```
action = dmx, 101, 255
```

identifier *action* > the line marks that the line describes an action

parameter 1 *dmx* > the action is to set an DMX output channel to a value

parameter 2 *101* > the DMX channel to set is channel 101

parameter 3 *255* > the DMX value to set is 255 (100%)

To include another text file at this position

```
#include = "textfile.txt" > this will insert the textfile at this position, like it was part of  
the current textfile. Each textfile (even included) can also  
include 1 or more files
```

Using variable parameters (this will be explained further on)

```
#someparameter = 1 > create a variable parameter and set it to value 1
```

Small examples:

event = button, 1, press describe the event of button 1 being pressed
action = dmx, 1, 255 the action to the event is to set DMX channel 1 to 255 (100%)
event = button, 1, release describe the event of button 1 being released
action = dmx, 1, 0 the action the event is to set DMX channel 1 to 0 (0%)

also multiple actions can be added to and event, example:

event = button, 2, press describe the event of button 2 being pressed
action = dmx, 1, 255 the action to the event is to set DMX channel 1 to 255 (100%)
action = dmx, 2, 128 the action to the event is to set DMX channel 2 to 128 (50%)
action = midi_note_on, 1, 23, 127
the action is to send midi note on channel 1, note 23, val. 127

This is then done for all buttons and faders to be programmed.

List of events

event = button, [button number], press
the event describes the button press of [button number]
event = button, [button number], press, shift
the event describes the button press of [button number], but only is the shift value is active
event = button, [button number], release
the event describes the button release of [button number]
event = button, [button number], release, shift
the event describes the button release of [button number], but only is the shift value is active
event = fader, [fader number], move
event = encoder, [encoder number], up
event = encoder, [encoder number], down

List of actions

DMX actions

action = dmx, [dmx channel number], [value]
the action describes to set dmx channel [dmx channel number] to the decimal value of [value]. When you want to send the current fader value, on a fader move event, then don't put a value. Example
event = fader, 1, move
action = dmx, 1
the dmx channel will be set to the value of the fader sending the event.

MIDI actions

action = midi_note_on, [midi channel], [note number], [velocity]
the action sends out a midi note on command with the values [midi channel], [note number] and [velocity]. When you want to send the fader value of the fader sending the event, put the velocity to 128. Example
event = fader, 2, move
action = midi_note_on, 1, 12, 128

when you move fader 2, it will send a midi note on to channel 1, note 12 with the velocity (value 0-127) proportional to the fader value

action = midi_note_off, [midi channel, [note number], [velocity]

this action acts the same like midi note on, except that it sends a midi note off.

action = midi_controller, [midi channel, [controller number], [value]

this action sends a controller value to midi channel, controller number. If value is 0-127, it will send this fixed value. If value is 128, then it will send the value of the fader sending the event.

Memory actions

Each configuration can have up to 10 memories. To use memories in a configuration, place the line

memory = 10

in the beginning of the configuration. Please note that memories consume some configuration space.

All memories will be HTP merged with the DMX input to the DMX output.

action = memory_fader, [memory number], [value]

set the master of memory [memory number] to value. If you do not add a value parameter, it will be set to the value of the fader generating the event.

action = memory_flash, [memory number], [value]

set a temporary flash (the master value remains saved) to memory [memory number]. Flash on > value = 1 / Flash off > value = 0.

action = memory_store, [memory number]

store the DMX input values to memory [memory number]

Special Actions

action = shift, 1

Set the shift value to 1

action = shift, 0

Set the shift value to 0

Configuration File Header

The beginning of each configuration file begins with

title = [text to display on the screen]

next set the serial port mode

serialport = midi_in_out

the current version only supports midi input / output. please note that the midi input signal is merged to the midi output.

Next all events with these actions follow.

Using variable parameters.

When you need to write a configuration file it would be a lot of copy/paste work as a lot of buttons and faders do the same action, except the some parameter values are different. To help you, we added the option to create and use variable parameters.

First you need to create the parameter, this is normally done at the beginning of the configuration.

```
#[parameter name] = [start value]
```

parameter name > name used to identify the parameter in the rest of the configuration. Please note that "include" cannot be used as a parameter name.

Example

```
#dmxchannel = 1
```

To use the parameter you place it instead of a fixed value in the parameter field of an event of action line, for example

```
action = dmx, #dmxchannel, 255      set dmx channel 1 to 255, as #dmxchannel  
                                     has the value of 1
```

There are a few ways to change the value of the variable parameter.

```
#[parameter name] = +      increment the parameter by 1  
#[parameter name] = +, 10  add 10 to the parameter  
#[parameter name] = -      decrement the parameter by 1  
#[parameter name] = -, 5   subtract 5 from the parameter  
#[parameter name] = 6      set the parameter to 6
```

Longer example

```
#dmxchannel = 1
```

```
#button = 1
```

```
event = button, #button, pressed
```

```
action = dmx, #dmxchannel, 255
```

```
#button = +
```

```
#dmxchannel = +
```

```
event = button, #button, pressed
```

```
action = dmx, #dmxchannel, 255
```

Please read this document again, but do this while looking in the configuration files found on the CD-rom in the config directory.