Documentation

Matrix-Devices

Author: *Volker Christian*

Linz, Oct. 2000

Contents

1 Midi Device

Matrix API of

devmidi v0.1

Author: *Volker Christian*

Linz, Oct. 2000

1.1 Introduction

This device provides most of the common midi-commands as actions and also as events. The device is capable of supporting more than one midi devices.

1.2 Requirements

None

1.3 Connection

```
use midiAlias = midi@midi.host("/dev/midixy");
```

1.4 Actions

1.4.1 Note On

```
midiAlias:NoteOn(int channel, int note, int velocity);
```

`int channel`: The Midi-Channel; range: {0, ... 15}

`int note`: The Midi-Note; range: {0, ... 127}

`int velocity`: The Velocity; range: {0, ... 127}

1.4.2 Note Off

```
midiAlias:NoteOff(int channel, int note, int velocity);
```

`int channel`: The Midi-Channel; range: {0, ... 15}

`int note`: The Midi-Note; range: {0, ... 127}

`int velocity`: The Velocity; range: {0, ... 127}

1.4.3 Polyphonic Key Pressure

```
midiAlias:PKPressure(int channel, int note, int velocity);
```

`int channel`: The Midi-Channel; range: {0, ... 15}

`int note`: The Midi-Note; range: {0, ... 127}

`int velocity`: The Velocity; range: {0, ... 127}

1.4.4 Control Change

```
midiAlias:ContChange(int channel, int controller, int value);
```

`int channel`: The Midi-Channel; range: {0, ... 15}

`int controller`: The Controller; range: {0, ... 127}

`int value`: The new Controller Value; range: {0, ... 127}

1.4.5 Program Change

`midiAlias:ProgChange(int channel, int prognr);`

`int channel`: The Midi-Channel; range: {0, ... 15}

`int prognr`: The Program Number; range: {0, ... 127}

1.4.6 Channel Pressure (After Touch)

`midiAlias:ChanPress(int channel, int pressure);`

`int channel`: The Midi-Channel; range: {0, ... 15}

`int pressure`: The Pressure Value; range: {0, ... 127}

1.4.7 Pitch Wheel Change

`midiAlias:PWChange(int channel, int value);`

`int channel`: The Midi-Channel; range: {0, ... 15}

`int value`: The Pitch-Wheel Value; range: {0, ... 127}

1.4.8 Song Position Pointer

`midiAlias:SPPointer(int position);`

`int position`: The Position in the Song; range: {0, ... 127}

1.4.9 Song Select

`midiAlias:SongSel(int song);`

`int song`: The Song Number; range: {0, ... 127}

1.4.10 Tune Request

`midiAlias:TuneReq(void);`

1.4.11 Send Timeing Clock

`midiAlias:STClock(void);`

1.4.12 Start Sequence

`midiAlias:StrtSeq(void);`

1.4.13 Stop Sequence

`midiAlias:StopSeq(void);`

1.4.14 Continue Sequence

`midiAlias:ContSeq(void);`

1.4.15 Reset

`midiAlias:Reset(void);`

1.5 Events

Every parameter of an Event can be prefixed by "^" to read the value comeing in.

1.5.1 Note On

[`state1`[|`state2`[...]]]`-> midiAlias:NoteOn(int channel, int note, int velocity)`

`int channel`: The Midi-Channel; range: {0, ... 15}

`int note`: The Midi-Note; range: {0, ... 127}

`int velocity`: The Velocity; range: {0, ... 127}

1.5.2 Note Off

[`state1`[|`state2`[...]]]`-> midiAlias:NoteOff(int channel, int note, int velocity)`

`int channel`: The Midi-Channel; range: {0, ... 15}

`int note`: The Midi-Note; range: {0, ... 127}

`int velocity`: The Velocity; range: {0, ... 127}

1.5.3 Polyphonic Key Pressure

[`state1`[|`state2`[...]]]`-> midiAlias:PKPressure(int channel, int note, int velocity)`

`int channel`: The Midi-Channel; range: {0, ... 15}

`int note`: The Midi-Note; range: {0, ... 127}

`int velocity`: The Velocity; range: {0, ... 127}

1.5.4 Control Change

[state1[|state2[...]]]-> midiAlias:ContChange(int channel, int controller, int value)

int channel: The Midi-Channel; range: {0, ... 15}

int controller: The Controller; range: {0, ... 127}

int value: The new Controller Value; range: {0, ... 127}

1.5.5 Program Change

[state1[|state2[...]]]-> midiAlias:ProgChange(int channel, int prognr)

int channel: The Midi-Channel; range: {0, ... 15}

int prognr: The Program Number; range: {0, ... 127}

1.5.6 Channel Pressure (After Touch)

[state1[|state2[...]]]-> midiAlias:ChanPress(int channel, int pressure)

int channel: The Midi-Channel; range: {0, ... 15}

int pressure: The Pressure Value; range: {0, ... 127}

1.5.7 Pitch Wheel Change

[state1[|state2[...]]]-> midiAlias:PWChange(int channel, int value)

int channel: The Midi-Channel; range: {0, ... 15}

int value: The Pitch-Wheel Value; range: {0, ... 127}

1.5.8 Song Position Pointer

[state1[|state2[...]]]-> midiAlias:SPPointer(int position)

int position: The Position in the Song; range: {0, ... 127}

1.5.9 Song Select

[state1[|state2[...]]]-> midiAlias:SongSel(int song)

int song: The Song Number; range: {0, ... 127}

1.5.10 Tune Request

[state1[|state2[...]]]-> midiAlias:TuneReq(void)

1.5.11 Send Timeing Clock

[state1[|state2[...]]]-> midiAlias:STClock(void)

1.5.12 Start Sequence

[state1[|state2[...]]]-> midiAlias:StrtSeq(void)

1.5.13 Stop Sequence

[state1[|state2[...]]]-> midiAlias:StopSeq(void)

1.5.14 Continue Sequence

[state1[|state2[...]]]-> midiAlias:ContSeq(void)

1.5.15 Reset

[state1[|state2[...]]]-> midiAlias:Reset(void)

2 CD-Player Device

Matrix API of

devcdplayer v0.1

Author: *Volker Christian*

Linz, Oct. 2000

2.1 Introduction

This device provides the ability to control a cd-rom drive to act as a cd-player.

2.2 Requirements

You need the libcdaudio library in the version >= 0.99.4 from `http://cdcd.undergrid.net/libcdaudio/`

2.3 Connection

```
use cdAlias = cdplayer@cdplayer.host("/dev/cdromx");
```

2.4 Actions

2.4.1 Play

```
cdAlias:play(int track);
```

`int track`: The CD-Track to play

2.4.2 Play Position

```
cdAlias:playpos(int track, int position);
```

`int track`: The CD-Track to play

`int position`: The Position to play from

2.4.3 Play Track

```
cdAlias:playtrack(int starttrack, int endtrack);
```

`int starttrack`: The CD-Track to start playing

`int endtrack`: The CD-Track to end playing

2.4.4 Play Track Position

```
cdAlias:playtrackpos(int starttrack, int endtrack, int position);
```

`int starttrack`: The CD-Track to start playing

`int endtrack`: The CD-Track to end playing

`int position`: The Position in the starttrack to play from

2.4.5 Play Frame

`cdAlias:playframe(int startframe, int endframe);`

`int startframe`: The Frame Position to start playing

`int endframe`: The Frame Position to end playing

2.4.6 Stop

`cdAlias:stop(void);`

2.4.7 Pause

`cdAlias:pause(void);`

2.4.8 Resume

`cdAlias:resume(void);`

2.4.9 Close

`cdAlias:close(void);`

2.4.10 Eject

`cdAlias:eject(void);`

2.5 Events

Every parameter of an Event can be prefixed by "^" to read the value comeing in.

2.5.1 Started Track

[`state1`[|`state2`[...]]]`-> cdAlias:startedtrack(int track)`

`int track`: The Tracknumber of the currently started Track

2.5.2 Ended Track

[`state1`[|`state2`[...]]]`-> cdAlias:endedtrack(int track)`

`int track`: The Tracknumber of the currently ended Track

2.5.3 Play

[`state1`[|`state2`[...]]]`-> cdAlias:play(void)`

2.5.4 Stop

[state1[|state2[...]]]-> cdAlias:stop(void)

2.5.5 Pause

[state1[|state2[...]]]-> cdAlias:pause(void)

2.5.6 Resume

[state1[|state2[...]]]-> cdAlias:resume(void)

2.5.7 Close

[state1[|state2[...]]]-> cdAlias:close(int number_of_tracks)

`int number_of_tracks`: The Number of Tracks on the inserted CD

2.5.8 Eject

[state1[|state2[...]]]-> cdAlias:eject(void)

3 Text-To-Speech (TTS) Device

Matrix API of

devtts v0.1

Author: *Volker Christian*

Linz, Oct. 2000

3.1 Introduction

This device provides an interface to some Text-To-Speech functions of the ViaVoice from IBM.

3.2 Requirements

You need the ViaVoice SDK from IBM installed.

3.3 Connection

```
use ttsAlias = tts@tts.host("");
```

3.4 Actions

3.4.1 Add String

```
ttsAlias:addstring(string text);
```

`string text:` The Sentence to add to the internal Buffer

3.4.2 Talk

```
ttsAlias:talk(void);
```

This command outputs the contents of the internal Buffer as spoken text.

3.4.3 Say

```
ttsAlias:say(string text);
```

`string text:` The text immediately spoken.

3.4.4 Stop

```
ttsAlias:stop(void);
```

Stop speeking and flush all Text in the internal Buffer.

3.4.5 Pause

```
ttsAlias:pause(void);
```

Pause speeking.

3.4.6 Continue

```
ttsAlias:continue(void);
```

Continue speeking

3.4.7 Reset

```
ttsAlias:reset(void);
```

Reset the Speech-Engine to the default state.

3.5 Events

None

4 Speech-Recognition Device

Matrix API of

devspeach v0.1

Author: *Volker Christian*

Linz, Oct. 2000

4.1 Introduction

This device provides an simple speech-recognition device for the matrix. It sends every recognized word as a command-event to the matrix.

4.2 Requirements

You need the ViaVoice SDK from IBM installed.

4.3 Connection

```
use speachAlias = speach@speach.host("");
```

4.4 Actions

4.4.1 Microphon On

```
speachAlias:micon(void);
```

Turn the microphon on

4.4.2 Microphon Off

```
speachAlias:micoff(void);
```

Turn the microphon off

4.5 Events

4.5.1 Command

[state1[|state2[...]]]-> speachAlias:command(string word)

`string work:` The recognized word.

5 Kramer Video-Matrix Device

Matrix API of

devkramer v0.1

Author: *Volker Christian*

Linz, Oct. 2000

5.1 Introduction

This device provides an interface to most functions of the "Kramer"-Video Matrix VS-2516 `http://www.kramer-electrics.com/group2/vs2516.html`.

5.2 Requirements

You need libkramer distributed with libvoc installed.

5.3 Connection

```
use kramerAlias = kramer@kramer.host("/dev/ttySxy"");
```

5.4 Actions

5.4.1 Switch Video

`kramerAlias:switch_video(int input, int output);`

`int input:` The Video-Input which should be connected to the Video-Output

`int output:` The Video-Output the Video-Input should be connected to

5.4.2 Switch Audio

`kramerAlias:switch_audio(int input, int output);`

`int input:` The Audio-Input which should be connected to the Audio-Output

`int output:` The Audio-Output the Audio-Input should be connected to

5.4.3 Store Video

`kramerAlias:store_video(int location);`

`int location:` Store the current Video-Setup in an internal Location; range {0,... 15}

5.4.4 Recall Video

`kramerAlias:recall_video(int location);`

`int location:` Recall a stored Video-Setup of an internal Location; range {0,... 15}

5.5 Events

5.5.1 Error

[`state1`[|`state2`[...]]]`-> kramerAlias:Error(int num)`

`int num:` If the matrix doesn't respond - the number of retries.

6 DMX-512 Device

Matrix API of

devdmx v0.1

Author: *Volker Christian*

Linz, Oct. 2000

6.1 Introduction

This device provides the possibility to control dmx-capable light-devices.

6.2 Requirements

A dmx-1512B Dmx-PC-Card from `http://www.soundlight.de/` and the `dmx512-2.2.tar.gz` package provided on the matrix-homepage.

6.3 Connection

```
use dmxAlias = dmx@midi.host("");
```

6.4 Actions

6.4.1 Start

```
dmxAlias:start(void);
```

Start the dmx-device

6.4.2 Dimming

```
dmxAlias:dimm(int channel, int startbright, int endbright, float
duration);
```

`int channel`: The Channel which should be dimmed

`int startbright`: Start Brightness of the dimm-sequence

`int endbright`: End Brightness of the dimm-sequence

`float duration`: Duration of the dim-sequence

6.4.3 Relative Dimming

```
dmxAlias:dimm_rel(int channel, int endbright, float duration);
```

`int channel`: The Channel which should be dimmed

`int endbright`: End Brightness of the dimm-sequence

`float duration`: Duration of the dim-sequence

6.4.4 Switching

```
dmxAlias:swith_to(int channel, int value);
```

`int channel`: The Channel which should be switched

`int value`: The value of the switching

6.4.5 Triangle

`dmxAlias:triangle(int channel, int start, int end, float duration, int count);`

`int channel`: The Channel which should be effected

`int start`: Start Brightness of the dimm-sequence

`int end`: End Brightness of the dimm-sequence

`float duration`: Duration of the dim-sequence

`int count`: Count for repeditive dimming

6.4.6 SawTooth

`dmxAlias:sawtooth(int channel, int start, int end, float duration, int count);`

`int channel`: The Channel which should be effected

`int start`: Start Brightness of the dimm-sequence

`int end`: End Brightness of the dimm-sequence

`float duration`: Duration of the dim-sequence

`int count`: Count for repeditive dimming

6.5 Events

None

7 Utility (clac) Plugin

Matrix API of

libd.calc.so v0.1

Author: *Volker Christian*

Linz, Oct. 2000

7.1 Introduction

This plugin provides many usefull utility-functions.

7.2 Requirements

None

7.3 Connection

```
include calcAlias = calc("");
```

7.4 Actions

7.4.1 Integer Multiplication

```
int calcAlias:itimes(int v1, int v2);
```

Multiplies the two integers v_1 and v_2.

7.4.2 Integer Division

```
int calcAlias:idiv(int v1, int v2);
```

Divides the two integers v_1 and v_2, $\frac{v_1}{v_2}$

7.4.3 Integer Modulo

```
int calcAlias:stop(int v, int m);
```

Calculates the modulo of m from v

7.4.4 Integer Addition

```
int calcAlias:stop(int v1, int v2);
```

Adds the two integers v_1 and v_2.

7.4.5 Integer Subtraction

```
int calcAlias:isub(int v1, int v2);
```

Subtracts the integer v_2 from v_1.

7.4.6 Integer to String conversion

```
string calcAlias:i2s(int v);
```

Converts the integer value v into a string.

7.4.7 Integer to Floatingpoint conversion

```
float calcAlias:i2f(int v);
```

Converts the integer value v into a float.

7.4.8 Floatingpoint Addition

```
float calcAlias:fadd(float v1, float v2);
```

Adds the two floats v_1 and v_2.

7.4.9 Floatingpoint Subtraction

```
float calcAlias:fsub(float v1, float v2);
```

Subtracts the float v_2 from v_1.

7.4.10 Floatingpoint Multiplication

```
float calcAlias:ftimes(float v1, float v2);
```

Multiplies the two floats v_1 and v_2.

7.4.11 Floatingpoint Division

```
float calcAlias:fdiv(float v1, float v2);
```

Divides v_1 by v_2

7.4.12 Floatingpoint Modulo

```
float calcAlias:fmod(float v, float m);
```

Calculates the modulo of m from v

7.4.13 Floatingpoint to String conversion

```
string calcAlias:f2s(float f);
```

Converts the float f to string.

7.4.14 String Concatenation

```
string calcAlias:sadd(string s1, string s2);
```

Concatenates s_2 to s_1.

7.4.15 String to Integer conversion

```
int calcAlias:s2i(string s);
```

Converts the string s into an integer

7.4.16 String to Float conversion

```
float calcAlias:s2f(string s);
```

Converts the string s into a float.

7.4.17 The Number Π

```
float calcAlias:pi(void);
```

Returns the number $3.14159\ldots$

7.4.18 Integer Random

```
int calcAlias:irandom(int p1, int p2);
```

Returns a random number in the range $[p1, p1 + p2]$

7.4.19 Uptime

```
float calcAlias:uptime(void);
```

Returns the uptime of the matrix in seconds.

7.4.20 Year

```
int calcAlias:get_data_year(void);
```

Returns the current year.

7.4.21 Month

```
int calcAlias:get_data_month(void);
```

Returns the current year.

7.4.22 Month-Day

```
int calcAlias:get_data_mday(void);
```

Returns the day in month.

7.4.23 Week-Day

```
int calcAlias:get_data_wday(void);
```

Returns the day in week.

7.4.24 Year-Day

```
int calcAlias:get_data_yday(void);
```

Returns the day in year.

7.4.25 Week-Day

```
int calcAlias:get_data_wday(void);
```

Returns the day in week.

7.4.26 Minutes

```
int calcAlias:get_data_min(void);
```

Returns the actual minute.

7.4.27 Seconds

```
int calcAlias:get_data_sec(void);
```

Returns the actual second.

7.4.28 Triggering a lable

```
int calcAlias:trigger_lable_insec(string lable, float secs);
```

Triggers the lable *lable* in *secs* seconds.

7.5 Events

7.5.1 Triggering a lable

[state1[|state2[...]]]-> calcAlias:lable(string lable)

This event gets triggered when the timer of the *Triggering a lable* action is timed out.