

MICCDlibrary

Generated by Doxygen 1.8.8

Sun Jan 11 2015 18:03:00

Contents

1	MI CCD Linux driver	1
1.1	Function list	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	camera_info_h_t Struct Reference	7
4.2	camera_info_t Struct Reference	7
4.2.1	Detailed Description	8
4.3	camera_t Struct Reference	8
4.3.1	Detailed Description	9
5	File Documentation	11
5.1	/home/petr/miccd/include/miccd.h File Reference	11
5.1.1	Detailed Description	12
5.1.2	Function Documentation	12
5.1.2.1	miccd_abort_exposure	12
5.1.2.2	miccd_chip_temperature	13
5.1.2.3	miccd_clear	13
5.1.2.4	miccd_close	13
5.1.2.5	miccd_close_shutter	13
5.1.2.6	miccd_environment_temperature	14
5.1.2.7	miccd_fan	15
5.1.2.8	miccd_filter	15
5.1.2.9	miccd_g1_mode	15
5.1.2.10	miccd_gain	15
5.1.2.11	miccd_hclear	16
5.1.2.12	miccd_hshift_clear	16
5.1.2.13	miccd_info	16

5.1.2.14	miccd_mode	16
5.1.2.15	miccd_open	17
5.1.2.16	miccd_open_shutter	17
5.1.2.17	miccd_power_voltage	17
5.1.2.18	miccd_read_eeprom	17
5.1.2.19	miccd_read_frame	18
5.1.2.20	miccd_set_cooltemp	18
5.1.2.21	miccd_shift	18
5.1.2.22	miccd_shift_to0	19
5.1.2.23	miccd_start_exposure	19
5.1.2.24	miccd_vshift_clear	19

Chapter 1

MI CCD Linux driver

Copyright © 2010 Petr Kubánek petr@kubanek.net

Copyright © 2010 Moravské přístroje s.r.o. (MI) <http://www.mii.cz>

Driver and associated documentation cannot be distributed without prior agreement from MI.

This is driver for MI CCD camera. You will need to link with libmiccd.a to get working binary, e.g. assuming libmiccd.a resides in /usr/local/lib, headers in /usr/local/include, and you are using cc for C compiler, link your code code.c with this library with:

```
cc -o code -I/usr/local/include code.c /usr/local/lib/libmiccd.a
```

With exception of miccd_open all functions takes as the first parameter pointer to structure identifying camera. The return value is 0 for success, negative for system errno value, and positive for error reply from camera itself.

The library does not need any kernel driver - access to USB bus through /dev/bus/usb, available in all recent kernels, is enough.

1.1 Function list

All function are documented under [global function list](#). Following links to their definition in order you will use them in the code.

- [miccd_open\(\)](#)
- [miccd_info\(\)](#)
- [miccd_open_shutter\(\)](#)
- [miccd_close_shutter\(\)](#)
- [miccd_read_frame\(\)](#)
- [miccd_close\(\)](#)

Author

Petr Kubánek petr@kubanek.net

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

camera_info_h_t	7
camera_info_t		
Camera information structure	7
camera_t		
Camera support structure	8

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

/home/petr/miccd/include/miccd.h	
Prototypes for MI CCD library functions	11

Chapter 4

Data Structure Documentation

4.1 camera_info_h_t Struct Reference

Data Fields

- uint32_t [id](#)
camera unique identifier
- uint8_t [hwrevision](#)
camera hardware revision
- uint8_t **filters**
- uint16_t **FIFOlines**
- uint16_t [w](#)
chip width [pixels]
- uint16_t [h](#)
chip height [pixels]
- uint16_t [pw](#)
pixel width [nm]
- uint16_t [ph](#)
pixel height [nm]
- char [description](#) [15]
camera description
- char [serial](#) [15]
camera serial number
- char [chip](#) [14]
camera chip identification

The documentation for this struct was generated from the following file:

- [/home/petr/miccd/include/miccd.h](#)

4.2 camera_info_t Struct Reference

Camera information structure.

```
#include <miccd.h>
```

Data Fields

- uint32_t [id](#)
camera unique identifier
- uint16_t [hwrevision](#)
camera hardware revision
- uint16_t [w](#)
chip width [pixels]
- uint16_t [h](#)
chip height [pixels]
- uint16_t [pw](#)
pixel width [nm]
- uint16_t [ph](#)
pixel height [nm]
- char [description](#) [15]
camera description
- char [serial](#) [15]
camera serial number
- char [chip](#) [14]
camera chip identification

4.2.1 Detailed Description

Camera information structure.

Contains fields retrieved from camera. Can be used to distinguish different cameras.

The documentation for this struct was generated from the following file:

- /home/petr/miccd/include/[miccd.h](#)

4.3 camera_t Struct Reference

Camera support structure.

```
#include <miccd.h>
```

Public Types

- enum { **NORMAL** = 0, **LOW**, **ULTRA_LOW** }
readout mode

Data Fields

- int [fd](#)
file descriptor of the camera
- uint8_t [binx](#)
binning in x
- uint8_t [biny](#)
binning in y
- uint16_t [w](#)

- exposure width*
- uint16_t [h](#)
 - exposure height*
- enum camera_t:: { ... } [mode](#)
 - readout mode*
- [model_t](#) **model**

4.3.1 Detailed Description

Camera support structure.

Holds various camera variables which cannot be retrieved from camera.

The documentation for this struct was generated from the following file:

- [/home/petr/miccd/include/miccd.h](#)

Chapter 5

File Documentation

5.1 /home/petr/miccd/include/miccd.h File Reference

Prototypes for MI CCD library functions.

Data Structures

- struct [camera_info_t](#)
Camera information structure.
- struct [camera_info_h_t](#)
- struct [camera_t](#)
Camera support structure.

Enumerations

- enum [model_t](#) {
 G10300, G10400, G10800, G11200,
 G11400, G12000, G2, G3,
 G3_H }
Camera model.

Functions

- int [miccd_open](#) (int32_t id, [camera_t](#) *camera)
Open connection to camera with given product ID.
- int [miccd_close](#) ([camera_t](#) *camera)
Close connection to camera.
- int [miccd_info](#) ([camera_t](#) *camera, [camera_info_t](#) *info)
Retrieve camera information structure from camera.
- int [miccd_g1_mode](#) ([camera_t](#) *camera, int bit16, int lownoise)
Set G1 camera mode.
- int [miccd_mode](#) ([camera_t](#) *camera, uint8_t mode)
Set camera mode.
- int [miccd_clear](#) ([camera_t](#) *camera)
Clear camera CCD.
- int [miccd_hclear](#) ([camera_t](#) *camera)
Clear horizontal register.

- int [miccd_shift_to0](#) ([camera_t](#) *camera)
Shifts rows till first imagine row.
- int [miccd_shift](#) ([camera_t](#) *camera)
Shift camera rows.
- int [miccd_vshift_clear](#) ([camera_t](#) *camera, uint16_t v)
Shift camera rows.
- int [miccd_hshift_clear](#) ([camera_t](#) *camera, uint16_t h)
Shift camera columns.
- int [miccd_read_frame](#) ([camera_t](#) *camera, uint8_t hbinning, uint8_t vbinning, uint16_t x, uint16_t y, uint16_t w, uint16_t h, char *data)
Readout camera chip.
- int [miccd_read_data](#) ([camera_t](#) *camera, uint32_t data_size, char *data, uint16_t w, uint16_t h)
- int [miccd_open_shutter](#) ([camera_t](#) *camera)
Open camera shutter.
- int [miccd_close_shutter](#) ([camera_t](#) *camera)
Close camera shutter.
- int32_t [miccd_start_exposure](#) ([camera_t](#) *camera, uint16_t x, uint16_t y, uint16_t w, uint16_t h, float exposure)
Start exposure.
- int [miccd_abort_exposure](#) ([camera_t](#) *camera)
Abort exposure on G1 camera.
- int [miccd_filter](#) ([camera_t](#) *camera, uint8_t filter)
Change filter in camera filter wheel.
- int [miccd_set_cooltemp](#) ([camera_t](#) *camera, float temp)
Set camera cooling temperature.
- int [miccd_chip_temperature](#) ([camera_t](#) *camera, float *temp)
Retrieve chip temperature.
- int [miccd_environment_temperature](#) ([camera_t](#) *camera, float *temp)
Retrieve environmental (surrounding) temperature.
- int [miccd_power_voltage](#) ([camera_t](#) *camera, uint16_t *voltage)
Retrieve camera voltage.
- int [miccd_gain](#) ([camera_t](#) *camera, uint16_t *gain)
Retrieve camera gain.
- int [miccd_fan](#) ([camera_t](#) *camera, int8_t fan)
Switch on/off camera fan.
- int [miccd_read_eeprom](#) ([camera_t](#) *camera, uint8_t offset, uint8_t size, void *buf)
Read EEPROM data from the camera.

5.1.1 Detailed Description

Prototypes for MI CCD library functions.

5.1.2 Function Documentation

5.1.2.1 int miccd_abort_exposure (camera_t * camera)

Abort exposure on G1 camera.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.2 int miccd_chip_temperature (camera_t * camera, float * temp)

Retrieve chip temperature.

Parameters

<i>temp</i>	pointer to float to store chip temperature (in degrees C).
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.3 int miccd_clear (camera_t * camera)

Clear camera CCD.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.4 int miccd_close (camera_t * camera)

Close connection to camera.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure.

5.1.2.5 int miccd_close_shutter (camera_t * camera)

Close camera shutter.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.6 `int miccd_environment_temperature (camera_t * camera, float * temp)`

Retrieve environmental (surrounding) temperature.

Parameters

<i>temp</i>	pointer to float to store environmental temperature (in degrees C).
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.7 int miccd_fan (camera_t * camera, int8_t fan)

Switch on/off camera fan.

Parameters

<i>fan</i>	turn fan on/off (boolean)
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.8 int miccd_filter (camera_t * camera, uint8_t filter)

Change filter in camera filter wheel.

Change filter in camera filter wheel. Filters are counted from 0.

Parameters

<i>filter</i>	filter number. Filters are counted from 0.
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.9 int miccd_g1_mode (camera_t * camera, int bit16, int lownoise)

Set G1 camera mode.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
<i>bit16</i>	if 16 bit readout should be enabled
<i>lownoise</i>	lownoise mode enabled

5.1.2.10 int miccd_gain (camera_t * camera, uint16_t * gain)

Retrieve camera gain.

Parameters

<i>gain</i>	pointer to integer to store gain level
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.11 int miccd_hclear (camera_t * camera)

Clear horizontal register.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.12 int miccd_hshift_clear (camera_t * camera, uint16_t h)

Shift camera columns.

This is equal to serial shift - only serial register is shifted by given number of pixels.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
<i>h</i>	number of pixels for serial shift

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.13 int miccd_info (camera_t * camera, camera_info_t * info)

Retrieve camera information structure from camera.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
<i>info</i>	camera descriptor structure, filled on successfull return with camera data

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.14 int miccd_mode (camera_t * camera, uint8_t mode)

Set camera mode.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
<i>mode</i>	camera mode (0 - normal, 1 = low noise; 2 = ultra low noise; 2 is not available on G1)

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.15 `int miccd_open (int32_t id, camera_t * camera)`

Open connection to camera with given product ID.

Parameters

<i>id</i>	camera product ID. If set to 0, the first found camera will be connected. You can use miccd_info(camera,info) call to retrieve its product ID
<i>camera</i>	camera support structure

Returns

opened file descriptor for camera, negative error code on error. Error code is negative errno value.

See also

[miccd_info\(camera,info\)](#)

5.1.2.16 `int miccd_open_shutter (camera_t * camera)`

Open camera shutter.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.17 `int miccd_power_voltage (camera_t * camera, uint16_t * voltage)`

Retrieve camera voltage.

Parameters

<i>voltage</i>	pointer to integer to store voltage level
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.18 `int miccd_read_eeprom (camera_t * camera, uint8_t offset, uint8_t size, void * buf)`

Read EEPROM data from the camera.

Parameters

<i>offset</i>	read EEPROM from this offset
<i>size</i>	size of buffer to read
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.19 `int miccd_read_frame (camera_t * camera, uint8_t hbinning, uint8_t vbinning, uint16_t x, uint16_t y, uint16_t w, uint16_t h, char * data)`

Readout camera chip.

Read data from camera and stored them in provided buffer. You can call this function anytime, but most probably you will prefer to call [miccd_open_shutter\(\)](#) and [miccd_close_shutter\(\)](#) before to accumulate some light on the camera.

Parameters

<i>camera</i>	camera_t structure filled in miccd_open() call
<i>hbinning</i>	frame horizontal binning
<i>vbinning</i>	frame vertical binning
<i>x</i>	ROI x start
<i>y</i>	ROI y start
<i>w</i>	ROI width
<i>h</i>	ROI height
<i>data</i>	pointer to buffer allocated to hold data. Must be large enough to hold all data from camera (e.g. at least $w/hbinning * h/vbinning * 2$ bytes).

Returns

0 on success, negative error code on (-errno) on failure

See also

[miccd_open_shutter](#)

[miccd_close_shutter](#)

5.1.2.20 `int miccd_set_cooltemp (camera_t * camera, float temp)`

Set camera cooling temperature.

Parameters

<i>temp</i>	target cooling temperature in degrees C. Please consult camera documentation for details of its cooling capabilities.
<i>camera</i>	camera structure filled in miccd_open() call

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.21 `int miccd_shift (camera_t * camera)`

Shift camera rows.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.22 int miccd_shift_to0 (camera_t * camera)

Shifts rows till first imagine row.

This jump over prescan region. After calling this function serial register holds first pixel of first non-prescan (image) row.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
---------------	--

Returns

0 on success, negative error code on (-errno) on failure

5.1.2.23 int32_t miccd_start_exposure (camera_t * camera, uint16_t x, uint16_t y, uint16_t w, uint16_t h, float exposure)

Start exposure.

Works only with G1 camera.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
<i>exposure</i>	exposure length in seconds. If it is < 0, then the frame will be read out (no exposure will occurs). Start of the exposure should be triggered by call to miccd_clear.

Returns

0 on success, negative error code on (-errno) on failure

See also

[miccd_clear](#)

5.1.2.24 int miccd_vshift_clear (camera_t * camera, uint16_t v)

Shift camera rows.

This is equal to paraller shift - full rows of camera data are read out and discarded.

Parameters

<i>camera</i>	camera structure filled in miccd_open() call
<i>v</i>	number of rows for paraller shift

Returns

0 on success, negative error code on (-errno) on failure