libcin

Generated by Doxygen 1.8.6

Thu Jul 6 2017 18:56:14

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 Cin Control Routines

Functions

- int cin_ctl_init (cin_ctl_t *cin, const char *ipaddr, uint16_t oport, uint16_t iport, uint16_t soport, uint16_t siport)
- int cin_ctl_destroy (cin_ctl_t *cin)
- int cin_ctl_read (cin_ctl_t *cin, uint16_t reg, uint16_t *val)
- int cin_ctl_write (cin_ctl_t *cin, uint16_t reg, uint16_t val, int wait)
- int cin_ctl_stream_write (cin_ctl_t *cin, char *val, int size)
- int cin_ctl_write_with_readback (cin_ctl_t *cin, uint16_t reg, uint16_t val)
- int **cin_ctl_pwr** (cin_ctl_t *cin, int pwr)
- int **cin_ctl_fp_pwr** (cin_ctl_t *cin, int pwr)
- int **cin_ctl_fo_test_pattern** (cin_ctl_t *cin, int on_off)
- int **cin_ctl_load_config** (cin_ctl_t *cin, char *filename)
- int **cin_ctl_load_firmware** (cin_ctl_t *cin, char *filename)
- int **cin_ctl_set_fclk** (cin_ctl_t *cin, int clkfreq)
- int **cin_ctl_get_fclk** (cin_ctl_t *cin, int *clkfreq)
- int **cin_ctl_freeze_dco** (cin_ctl_t *cin, int freeze)
- int **cin_ctl_get_cfg_fpga_status** (cin_ctl_t *cin, uint16_t *_val)
- int **cin_ctl_get_id** (cin_ctl_t *cin, cin_ctl_id_t *_val)
- void **cin_ctl_display_id** (FILE *out, cin_ctl_id_t val)
- void **cin_ctl_display_fpga_status** (FILE *out, uint16_t val)
- int **cin_ctl_get_dcm_status** (cin_ctl_t *cin, uint16_t *_val)
- void **cin_ctl_display_dcm_status** (FILE *out, uint16_t *_val)
- double **cin_ctl_current_calc** (uint16_t val)
- int **cin_ctl_get_power_status** (cin_ctl_t *cin, int full, int *pwr, cin_ctl_pwr_mon_t *values)
- void **cin_ctl_display_pwr** (FILE *out, cin_ctl_pwr_mon_t *values)
- void **cin_ctl_display_pwr_line** (FILE *out, const char *msg, cin_ctl_pwr_val_t val)
- int **cin_ctl_calc_vi_status** (cin_ctl_t *cin, uint16_t vreg, uint16_t ireg, double vfact, cin_ctl_pwr_val_t *vi)
- int **cin_ctl_get_camera_pwr** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_camera_pwr** (cin_ctl_t *cin, int val)
- int **cin_ctl_set_bias** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_bias** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_clocks** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_clocks** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_trigger** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_trigger** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_focus** (cin_ctl_t *cin, int val)

- int **cin_ctl_get_focus** (cin_ctl_t ∗cin, int ∗val)
- int **cin_ctl_get_triggering** (cin_ctl_t ∗cin, int ∗trigger)
- int **cin_ctl_int_trigger_start** (cin_ctl_t ∗cin, int nimages)
- int **cin_ctl_int_trigger_stop** (cin_ctl_t ∗cin)
- int **cin_ctl_ext_trigger_start** (cin_ctl_t ∗cin, int trigger_mode)
- int **cin_ctl_ext_trigger_stop** (cin_ctl_t ∗cin)
- int **cin_ctl_set_exposure_time** (cin_ctl_t ∗cin, float e_time)
- int **cin_ctl_set_trigger_delay** (cin_ctl_t ∗cin, float t_time)
- int **cin_ctl_set_cycle_time** (cin_ctl_t ∗cin, float ftime)
- int **cin_ctl_frame_count_reset** (cin_ctl_t ∗cin)
- int **cin_ctl_set_mux** (cin_ctl_t ∗cin, int setting)
- int **cin_ctl_get_mux** (cin_ctl_t ∗cin, int ∗setting)
- int **cin_ctl_set_fcric_gain** (cin_ctl_t ∗cin, int gain)
- int **cin_ctl_set_fabric_address** (cin_ctl_t ∗cin, char ∗ip)
- int **cin_ctl_reg_dump** (cin_ctl_t ∗cin, FILE ∗fp)
- int **cin_ctl_get_bias_voltages** (cin_ctl_t ∗cin, float ∗voltage)
- int **cin_ctl_set_bias_voltages** (cin_ctl_t ∗cin, float ∗voltage)
- int **cin_ctl_set_fcric_clamp** (cin_ctl_t ∗cin, int clamp)

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 int cin_ctl_destroy (cin_ctl_t ∗ *cin*)

Destroy (close) the cin control library

Close connections, free memory and exit library

Parameters

cin	handle to cin library

Returns

Returns 0 on sucsess non-zero if error

4.1.2.2 int cin_ctl_init (cin_ctl_t ∗ *cin*, const char ∗ *ipaddr*, uint16_t *oport*, uint16_t *iport*, uint16_t *soport*, uint16_t *siport*)

Initialize the cin control library

Initialize the control structures and communications with the CIN via the control interface. This function opens the UDP ports and starts a listening thread to recieve packets from the CIN.

Parameters

cin	handle to cin library
ipaddr	ip address of CIN base address
oport	output udp port of cin
iport	input udp port of cin
soport	stream output udp port of cin
siport	stream input udp port of cin

Returns

Returns 0 on sucsess non-zero if error

4.1.2.3 int cin_ctl_read (cin_ctl_t * *cin*, uint16_t *reg*, uint16_t * *val*)

Read register from CIN

Parameters

cin	handle to cin library
reg	register to read
val	variable to read value of register to

Returns

Returns 0 on sucsess non-zero if error

4.1.2.4 int cin_ctl_stream_write (cin_ctl_t * *cin*, char * *val*, int *size*)

Write stream data to CIN

Parameters

cin	handle to cin library
val	array of values to write
size	size of array pointed to by val

Write stream data to cin in form of 16 bit array.

Returns

Returns 0 on sucsess non-zero if error

4.1.2.5 int cin_ctl_write (cin_ctl_t * *cin*, uint16_t *reg*, uint16_t *val*, int *wait*)

Write register to CIN

Parameters

cin	handle to cin library
reg	register to write to
val	value to write to register
wait	if non-zero

Write register value to CIN. If wait is non-zero then wait a sleep time of i CIN_CTL_WRITE_SLEEP before releasing the mutex to add flow control to the cin.

Returns

Returns 0 on sucsess non-zero if error

4.1.2.6 int cin_ctl_write_with_readback (cin_ctl_t * *cin*, uint16_t *reg*, uint16_t *val*)

Write register to CIN with readback verification

Parameters

cin	handle to cin library
reg	register to write to
val	value to write to register

Write register value to CIN. Follow write with read of register and compare value. CIN_CTL_WRITE_SLEEP before releasing the mutex to add flow control to the cin.

Returns

Returns 0 on sucsess non-zero if error

4.2 CIN Data Framestore Functions

Data group

Chapter 5

Class Documentation

5.1 cin_ctl Struct Reference

Public Attributes

- cin_port_t **ctl_port**
- cin_port_t **stream_port**
- cin_ctl_config_t **config**
- cin_ctl_listener_t * **listener**
- pthread_mutex_t **access**
- pthread_mutexattr_t **access_attr**

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

- src/cin.h

5.2 cin_ctl_config Struct Reference

Public Attributes

- char **name** [CIN_CONFIG_MAX_STRING]
- char **firmware_filename** [CIN_CONFIG_MAX_STRING]
- int **overscan**
- int **columns**
- int **fclk**
- uint16_t **timing** [CIN_CONFIG_MAX_DATA][2]
- int **timing_len**
- uint16_t **fcric** [CIN_CONFIG_MAX_DATA][2]
- int **fcric_len**
- uint16_t **bias** [CIN_CONFIG_MAX_DATA][2]
- int **bias_len**

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

- src/cin.h

5.3 cin_ctl_id Struct Reference

Public Attributes

- uint16_t **board_id**
- uint16_t **serial_no**
- uint16_t **fpga_ver**

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

- src/cin.h

5.4 cin_ctl_listener Struct Reference

Public Attributes

- struct cin_port * **cp**
- fifo **ctl_fifo**
- pthread_t **thread_id**

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

- src/cin.h

5.5 cin_ctl_pwr_mon_t Struct Reference

Public Attributes

- cin_ctl_pwr_val_t **bus_12v0**
- cin_ctl_pwr_val_t **mgmt_3v3**
- cin_ctl_pwr_val_t **mgmt_2v5**
- cin_ctl_pwr_val_t **mgmt_1v2**
- cin_ctl_pwr_val_t **enet_1v0**
- cin_ctl_pwr_val_t **s3e_3v3**
- cin_ctl_pwr_val_t **gen_3v3**
- cin_ctl_pwr_val_t **gen_2v5**
- cin_ctl_pwr_val_t **v6_0v9**
- cin_ctl_pwr_val_t **v6_1v0**
- cin_ctl_pwr_val_t **v6_2v5**
- cin_ctl_pwr_val_t **fp**

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

- src/cin.h

5.6 cin_ctl_pwr_val Struct Reference

Public Attributes

- double **i**

- double **v**

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

- src/cin.h

5.7 cin_data Struct Reference

Public Attributes

- fifo * **packet_fifo**
- fifo * **frame_fifo**
- fifo * **image_fifo**
- cin_data_threads_t **listen_thread**
- cin_data_threads_t **assembler_thread**
- cin_data_threads_t **descramble_thread**
- pthread_mutex_t **listen_mutex**
- pthread_mutex_t **assembler_mutex**
- pthread_mutex_t **descramble_mutex**
- pthread_mutex_t **stats_mutex**
- pthread_mutex_t **framestore_mutex**
- cin_data_callbacks_t **callbacks**
- cin_port_t **dp**
- struct timespec **framerate**
- unsigned long int **dropped_packets**
- unsigned long int **mallformed_packets**
- uint16_t **last_frame**
- descramble_map_t **map**
- int **framestore_mode**
- struct timespec **framestore_trigger**
- int **framestore_counter**

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

- src/cin.h

5.8 cin_data_callbacks Struct Reference

Public Attributes

- void *(* **push**)(cin_data_frame_t *)
- void *(* **pop**)(cin_data_frame_t *)
- cin_data_frame_t * **frame**

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

- src/cin.h

5.9 cin_data_frame Struct Reference

Public Attributes

- uint16_t * **data**
- uint16_t **number**
- struct timespec **timestamp**
- int **size_x**
- int **size_y**
- void * **usr_ptr**

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

- src/cin.h

5.10 cin_data_packet Struct Reference

Public Attributes

- unsigned char * **data**
- int **size**
- struct timespec **timestamp**

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

- src/data.h

5.11 cin_data_proc Struct Reference

Public Attributes

- void *(* **input_get**)(void *, int)
- void *(* **input_put**)(void *, int)
- void * **input_args**
- int **reader**
- void *(* **output_put**)(void *)
- void *(* **output_get**)(void *)
- void * **output_args**
- cin_data_t * **parent**

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

- src/data.h

5.12 cin_data_stats Struct Reference

Public Attributes

- int **last_frame**
- double **framerate**

- double **datarate**
- double **packet_percent_full**
- double **frame_percent_full**
- double **image_percent_full**
- long int **packet_overruns**
- long int **frame_overruns**
- long int **image_overruns**
- long int **packet_used**
- long int **frame_used**
- long int **image_used**
- long int **dropped_packets**
- long int **mallformed_packets**

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

- src/cin.h

5.13 cin_data_threads Struct Reference

Public Attributes

- pthread_t **thread_id**
- int **started**

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

- src/cin.h

5.14 cin_map_t Struct Reference

Public Attributes

- char * **name**
- uint16_t **reg**

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

- src/cinregisters.h

5.15 cin_port Struct Reference

Public Attributes

- char * **srvaddr**
- char * **cliaddr**
- uint16_t **srvport**
- uint16_t **cliport**
- int **sockfd**
- struct timeval **tv**
- struct sockaddr_in **sin_srv**
- struct sockaddr_in **sin_cli**

- socklen_t **slen**
- int **rcvbuf**
- int **rcvbuf_rb**

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

- src/cin.h

5.16 descramble_map_t Struct Reference

Public Attributes

- uint32_t * **map**
- int **size_x**
- int **size_y**
- int **overscan**
- int **rows**

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

- src/cin.h

5.17 fifo Struct Reference

Public Attributes

- void * **data**
- void * **head**
- void * **tail** [FIFO_MAX_READERS]
- void * **end**
- int **readers**
- long int **size**
- int **elem_size**
- int **full**
- long int **overruns**
- pthread_mutex_t **mutex**
- pthread_cond_t **signal**

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

- src/cin.h

Chapter 6

File Documentation

6.1 src/cin.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <sys/time.h>
#include <pthread.h>
```

Classes

- struct cin_ctl_config
- struct fifo
- struct cin_ctl_listener
- struct cin_port
- struct cin_ctl
- struct cin_data_frame
- struct cin_data_stats
- struct cin_data_threads
- struct cin_data_callbacks
- struct descramble_map_t
- struct cin_data
- struct cin_ctl_id
- struct cin_ctl_pwr_val
- struct cin_ctl_pwr_mon_t

Macros

- #define **CIN_CTL_IP** "192.168.1.207"
- #define **CIN_CTL_SVR_PORT** 49200
- #define **CIN_CTL_CLI_PORT** 50200
- #define **CIN_CTL_SVR_FRMW_PORT** 49202
- #define **CIN_CTL_CLI_FRMW_PORT** 50202
- #define **CIN_CTL_MAX_READ_TRIES** 10
- #define **CIN_CTL_MAX_WRITE_TRIES** 5
- #define **CIN_CTL_WRITE_SLEEP** 2000

- #define **CIN_CTL_POWER_ENABLE** 0x001F
- #define **CIN_CTL_POWER_DISABLE** 0x0000
- #define **CIN_CTL_FP_POWER_ENABLE** 0x0020
- #define **CIN_CTL_DCM_LOCKED** 0x0001
- #define **CIN_CTL_DCM_PSDONE** 0x0002
- #define **CIN_CTL_DCM_STATUS0** 0x0004
- #define **CIN_CTL_DCM_STATUS1** 0x0008
- #define **CIN_CTL_DCM_STATUS2** 0x0010
- #define **CIN_CTL_DCM_TX1_READY** 0x0020
- #define **CIN_CTL_DCM_TX2_READY** 0x0040
- #define **CIN_CTL_DCM_ATCA_ALARM** 0x0080
- #define **CIN_CTL_TRIG_INTERNAL** 0x0000
- #define **CIN_CTL_TRIG_EXTERNAL_1** 0x0001
- #define **CIN_CTL_TRIG_EXTERNAL_2** 0x0002
- #define **CIN_CTL_TRIG_EXTERNAL_BOTH** 0x0003
- #define **CIN_CTL_FOCUS_BIT** 0x0002
- #define **CIN_CTL_FCLK_125** 0x0000
- #define **CIN_CTL_FCLK_200** 0x0001
- #define **CIN_CTL_FCLK_250** 0x0002
- #define **CIN_CTL_FCLK_125_C** 0x0003
- #define **CIN_CTL_FCLK_200_C** 0x0004
- #define **CIN_CTL_FCLK_250_C** 0x0005
- #define **CIN_CTL_FCLK_156_C** 0x0006
- #define **CIN_CTL_FPGA_STS_CFG** 0x8000
- #define **CIN_CTL_FPGA_STS_FP_PWR** 0x0008
- #define **CIN_CTL_DCM_STS_ATCA** 0x0080
- #define **CIN_CTL_DCM_STS_LOCKED** 0x0001
- #define **CIN_CTL_DCM_STS_OVERIDE** 0x0800
- #define **CIN_CTL_MUX1_VCLK1** 0x0001
- #define **CIN_CTL_MUX1_VCLK2** 0x0002
- #define **CIN_CTL_MUX1_VCLK3** 0x0003
- #define **CIN_CTL_MUX1_ATG** 0x0004
- #define **CIN_CTL_MUX1_VFSCLK1** 0x0005
- #define **CIN_CTL_MUX1_VFSCLK2** 0x0006
- #define **CIN_CTL_MUX1_VFSCLK3** 0x0007
- #define **CIN_CTL_MUX1_HCLK1** 0x0008
- #define **CIN_CTL_MUX1_HCLK2** 0x0009
- #define **CIN_CTL_MUX1_OSW** 0x000A
- #define **CIN_CTL_MUX1_RST** 0x000B
- #define **CIN_CTL_MUX1_CONVERT** 0x000C
- #define **CIN_CTL_MUX1_SHUTTER** 0x000D
- #define **CIN_CTL_MUX1_SWTRIGGER** 0x000E
- #define **CIN_CTL_MUX1_TRIGMON** 0x000F
- #define **CIN_CTL_MUX1_EXPOSE** 0x0000
- #define **CIN_CTL_MUX2_VCLK1** 0x0010
- #define **CIN_CTL_MUX2_VCLK2** 0x0020
- #define **CIN_CTL_MUX2_VCLK3** 0x0030
- #define **CIN_CTL_MUX2_ATG** 0x0040
- #define **CIN_CTL_MUX2_VFSCLK1** 0x0050
- #define **CIN_CTL_MUX2_VFSCLK2** 0x0060
- #define **CIN_CTL_MUX2_VFSCLK3** 0x0070
- #define **CIN_CTL_MUX2_HCLK1** 0x0080
- #define **CIN_CTL_MUX2_HCLK2** 0x0090
- #define **CIN_CTL_MUX2_HCLK3** 0x00A0
- #define **CIN_CTL_MUX2_OSW** 0x00B0

- #define **CIN_CTL_MUX2_RST** 0x00C0
- #define **CIN_CTL_MUX2_CONVERT** 0x00D0
- #define **CIN_CTL_MUX2_SAVE** 0x00E0
- #define **CIN_CTL_MUX2_HWTRIG** 0x00F0
- #define **CIN_CTL_MUX2_EXPOSE** 0x0000
- #define **CIN_CTL_FO_REG1** 0x821D
- #define **CIN_CTL_FO_REG2** 0x821E
- #define **CIN_CTL_FO_REG3** 0x821F
- #define **CIN_CTL_FO_REG4** 0x8001
- #define **CIN_CTL_FO_REG5** 0x8211
- #define **CIN_CTL_FO_REG6** 0x8212
- #define **CIN_CTL_FO_REG7** 0x8213
- #define **CIN_DATA_IP** "10.0.5.207"
- #define **CIN_DATA_PORT** 49201
- #define **CIN_DATA_CTL_PORT** 49203
- #define **CIN_DATA_MAX_MTU** 9000
- #define **CIN_DATA_UDP_HEADER** 8
- #define **CIN_DATA_MAGIC_PACKET** UINT64_C(0x0000F4F3F2F1F000)
- #define **CIN_DATA_MAGIC_PACKET_MASK** UINT64_C(0x0000FFFFFFFFFF00)
- #define **CIN_DATA_TAIL_MAGIC_PACKET** UINT64_C(0x010DF0ADDEF2F1F0)
- #define **CIN_DATA_TAIL_MAGIC_PACKET_MASK** UINT64_C(0xFFFFFFFFFFFFFFFF)
- #define **CIN_DATA_DROPPED_PACKET_VAL** 0x2000
- #define **CIN_DATA_DATA_MASK** 0x1FFF
- #define **CIN_DATA_CTRL_MASK** 0xE000
- #define **CIN_DATA_SIGN_MASK** 0x1000
- #define **CIN_DATA_GAIN_8** 0xC000
- #define **CIN_DATA_GAIN_4** 0x4000
- #define **CIN_DATA_PACKET_LEN** 8184
- #define **CIN_DATA_MAX_PACKETS** 542
- #define **CIN_DATA_RCVBUF** 100
- #define **CIN_DATA_MAX_FRAME_X** 1152
- #define **CIN_DATA_MAX_FRAME_Y** 2050
- #define **CIN_DATA_MAX_STREAM** 2400000
- #define **CIN_DATA_CCD_COLS** 96
- #define **CIN_DATA_CCD_COLS_PER_CHAN** 10
- #define **CIN_DATA_PIPELINE_FLUSH** 1344
- #define **CIN_DATA_MODE_CALLBACK** 0x01
- #define **NUM_BIAS_VOLTAGE** 20
- #define **pt_posH** 0
- #define **pt_negH** 1
- #define **pt_posRG** 2
- #define **pt_negRG** 3
- #define **pt_posSW** 4
- #define **pt_negSW** 5
- #define **pt_posV** 6
- #define **pt_negV** 7
- #define **pt_posTG** 8
- #define **pt_negTG** 9
- #define **pt_posVF** 10
- #define **pt_negVF** 11
- #define **pt_NEDGE** 12
- #define **pt_OTG** 13
- #define **pt_VDDR** 14
- #define **pt_VDD_OUT** 15
- #define **pt_BUF_Base** 16

- #define **pt_BUF_Delta** 17
- #define **pt_Spare1** 18
- #define **pt_Spare2** 19
- #define **DEBUG_PRINT**(fmt,...) if(_debug_print_flag) { fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, __LINE__, __func__, __VA_ARGS__); }
- #define **DEBUG_COMMENT**(fmt) if(_debug_print_flag) { fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, __LINE__, __func__); }
- #define **ERROR_COMMENT**(fmt) if(_error_print_flag) { fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, __LINE__, __func__); }
- #define **ERROR_PRINT**(fmt,...) if(_error_print_flag) { fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, __LINE__, __func__, __VA_ARGS__); }
- #define **CIN_CONFIG_MAX_STRING** 256
- #define **CIN_CONFIG_MAX_DATA** 5000
- #define **FIFO_MAX_READERS** 10

Typedefs

- typedef struct cin_ctl_config **cin_ctl_config_t**
- typedef struct cin_ctl_listener **cin_ctl_listener_t**
- typedef struct cin_port **cin_port_t**
- typedef struct cin_ctl **cin_ctl_t**
- typedef struct cin_data_frame **cin_data_frame_t**
- typedef struct cin_data_stats **cin_data_stats_t**
- typedef struct cin_data_threads **cin_data_threads_t**
- typedef struct cin_data_callbacks **cin_data_callbacks_t**
- typedef struct cin_data **cin_data_t**
- typedef void(∗ **cin_data_callback**)(cin_data_frame_t ∗)
- typedef struct cin_ctl_id **cin_ctl_id_t**
- typedef struct cin_ctl_pwr_val **cin_ctl_pwr_val_t**

Functions

- void **cin_set_debug_print** (int debug)
- void **cin_set_error_print** (int error)
- void **cin_report** (FILE ∗fp, int details)
- int cin_ctl_init (cin_ctl_t ∗cin, const char ∗ipaddr, uint16_t oport, uint16_t iport, uint16_t soport, uint16_t siport)
- int cin_ctl_destroy (cin_ctl_t ∗cin)
- int cin_ctl_read (cin_ctl_t ∗cin, uint16_t reg, uint16_t ∗val)
- int cin_ctl_write (cin_ctl_t ∗cin, uint16_t reg, uint16_t val, int wait)
- int cin_ctl_stream_write (cin_ctl_t ∗cin, char ∗val, int size)
- int cin_ctl_write_with_readback (cin_ctl_t ∗cin, uint16_t reg, uint16_t val)
- int **cin_ctl_pwr** (cin_ctl_t ∗cin, int pwr)
- int **cin_ctl_fp_pwr** (cin_ctl_t ∗cin, int pwr)
- int **cin_ctl_fo_test_pattern** (cin_ctl_t ∗cin, int on_off)
- int **cin_ctl_load_config** (cin_ctl_t ∗cin, char ∗filename)
- int **cin_ctl_load_firmware** (cin_ctl_t ∗cin, char ∗filename)
- int **cin_ctl_set_fclk** (cin_ctl_t ∗cin, int clkfreq)
- int **cin_ctl_get_fclk** (cin_ctl_t ∗cin, int ∗clkfreq)
- int **cin_ctl_freeze_dco** (cin_ctl_t ∗cin, int freeze)
- int **cin_ctl_get_cfg_fpga_status** (cin_ctl_t ∗cin, uint16_t ∗_val)
- int **cin_ctl_get_id** (cin_ctl_t ∗cin, cin_ctl_id_t ∗_val)
- void **cin_ctl_display_id** (FILE ∗out, cin_ctl_id_t val)
- void **cin_ctl_display_fpga_status** (FILE ∗out, uint16_t val)

- int **cin_ctl_get_dcm_status** (cin_ctl_t *cin, uint16_t *_val)
- void **cin_ctl_display_dcm_status** (FILE *out, uint16_t *_val)
- double **cin_ctl_current_calc** (uint16_t val)
- int **cin_ctl_get_power_status** (cin_ctl_t *cin, int full, int *pwr, cin_ctl_pwr_mon_t *values)
- void **cin_ctl_display_pwr** (FILE *out, cin_ctl_pwr_mon_t *values)
- void **cin_ctl_display_pwr_line** (FILE *out, const char *msg, cin_ctl_pwr_val_t val)
- int **cin_ctl_calc_vi_status** (cin_ctl_t *cin, uint16_t vreg, uint16_t ireg, double vfact, cin_ctl_pwr_val_t *vi)
- int **cin_ctl_get_camera_pwr** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_camera_pwr** (cin_ctl_t *cin, int val)
- int **cin_ctl_set_bias** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_bias** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_clocks** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_clocks** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_trigger** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_trigger** (cin_ctl_t *cin, int *val)
- int **cin_ctl_set_focus** (cin_ctl_t *cin, int val)
- int **cin_ctl_get_focus** (cin_ctl_t *cin, int *val)
- int **cin_ctl_get_triggering** (cin_ctl_t *cin, int *trigger)
- int **cin_ctl_int_trigger_start** (cin_ctl_t *cin, int nimages)
- int **cin_ctl_int_trigger_stop** (cin_ctl_t *cin)
- int **cin_ctl_ext_trigger_start** (cin_ctl_t *cin, int trigger_mode)
- int **cin_ctl_ext_trigger_stop** (cin_ctl_t *cin)
- int **cin_ctl_set_exposure_time** (cin_ctl_t *cin, float e_time)
- int **cin_ctl_set_trigger_delay** (cin_ctl_t *cin, float t_time)
- int **cin_ctl_set_cycle_time** (cin_ctl_t *cin, float ftime)
- int **cin_ctl_frame_count_reset** (cin_ctl_t *cin)
- int **cin_ctl_set_mux** (cin_ctl_t *cin, int setting)
- int **cin_ctl_get_mux** (cin_ctl_t *cin, int *setting)
- int **cin_ctl_set_fcric_gain** (cin_ctl_t *cin, int gain)
- int **cin_ctl_set_fabric_address** (cin_ctl_t *cin, char *ip)
- int **cin_ctl_reg_dump** (cin_ctl_t *cin, FILE *fp)
- int **cin_ctl_get_bias_voltages** (cin_ctl_t *cin, float *voltage)
- int **cin_ctl_set_bias_voltages** (cin_ctl_t *cin, float *voltage)
- int **cin_ctl_set_fcric_clamp** (cin_ctl_t *cin, int clamp)
- int **cin_config_read_file** (cin_ctl_t *cin, const char *file)
- int cin_data_init (cin_data_t *cin, int mode, int packet_buffer_len, int frame_buffer_len, char *ipaddr, uint16_t port, char *cin_ipaddr, uint16_t cin_port, int rcvbuf, cin_data_callback push_callback, cin_data_callback pop_callback, void *usr_ptr)
- void **cin_data_wait_for_threads** (cin_data_t *cin)
- void **cin_data_stop_threads** (cin_data_t *cin)
- void cin_data_framestore_trigger (cin_data_t *cin, int count)
- void cin_data_framestore_skip (cin_data_t *cin, int count)
- int cin_data_get_framestore_counter (cin_data_t *cin)
- void cin_data_framestore_disable (cin_data_t *cin)
- void cin_data_framestore_trigger_enable (cin_data_t *cin)
- struct cin_data_frame * **cin_data_get_next_frame** (cin_data_t *cin)
- void **cin_data_release_frame** (cin_data_t *cin, int free_mem)
- struct cin_data_frame * **cin_data_get_buffered_frame** (void)
- void **cin_data_release_buffered_frame** (void)
- void **cin_data_compute_stats** (cin_data_t *cin, cin_data_stats_t *stats)
- void **cin_data_show_stats** (FILE *fp, cin_data_stats_t stats)
- void **cin_data_reset_stats** (cin_data_t *cin)
- int **cin_data_set_descramble_params** (cin_data_t *cin, int rows, int overscan)
- void **cin_data_get_descramble_params** (cin_data_t *cin, int *rows, int *overscan, int *xsize, int *ysize)

Variables

- const char * **cin_build_git_time**
- const char * **cin_build_git_sha**
- const char * **cin_build_version**
- int **_debug_print_flag**
- int **_error_print_flag**

6.1.1 Detailed Description

Author

Stuart B. Wilkins swilkins@bnl.gov

6.1.2 LICENSE

6.1.3 DESCRIPTION

header file for CIN communications

6.1.4 Function Documentation

6.1.4.1 void cin_data_framestore_disable (cin_data_t * *cin*)

Disable the framestore modes

This function disables the framestore modes (software trigger and skip). If the camera is hardware triggering then the images will start to be processed.

Parameters

cin	Handle to the cin library

6.1.4.2 void cin_data_framestore_skip (cin_data_t * *cin*, int *count*)

Enable framestore skip mode

Enable the framestore skip mode. This function should be called before hardware triggering the camera. This causes the data processing to skip

Parameters

count	frames from the first images to be read. This is usually done to stop the first few frames from being over exposed.
cin	handle to the cin_data library

6.1.4.3 void cin_data_framestore_trigger (cin_data_t * *cin*, int *count*)

Send a framestore (software) trigger

Send a software trigger to the CIN by timestamping the request time and allow images to be processed when recieved after this time. The function is enabled by setting the framestore mode to CIN_DATA_FRAMESTORE_TRIGGER. The count option sets the number of frames to trigger. A value of -1 indicated that the trigger should not count images but run indefinately after the trigger has occured.

Parameters

cin	handle to the cin_data library
count	number of frames to trigger

6.1.4.4 void cin_data_framestore_trigger_enable (cin_data_t * *cin*)

Enable the framestore trigger mode

This function enables the framestore trigger mode. It cases the images to not be processed pending a call to the function to (software) trigger the camera.

Parameters

cin	Handle to the cin library

6.1.4.5 int cin_data_get_framestore_counter (cin_data_t * *cin*)

Get the value of the framestore counter

Return the number of frames in the framestore counter. In trigger mode, this returns the number of frames to go. In skip mode, this returns the number of frames that have to be skipped.

Parameters

cin	handle to the cin_data library

Returns

Number of frames to go in trigger

6.1.4.6 int cin_data_init (cin_data_t * *cin*, int *mode*, int *packet_buffer_len*, int *frame_buffer_len*, char * *ipaddr*, uint16_t *port*, char * *cin_ipaddr*, uint16_t *cin_port*, int *rcvbuf*, cin_data_callback *push_callback*, cin_data_callback *pop_callback*, void * *usr_ptr*)

Initialize the cin data library

Initialize the data handeling routines and start the threads for listening. mode should be set for the desired output. The packet_buffer_len in the length of the packet FIFO in number of packets. The frame_buffer_len is the number of data frames to buffer.

Parameters

cin	Handle to cin data library

Index