libcin

Generated by Doxygen 1.8.13

Contents

Chapter 1

FastCCD Communication Library (libcin)

Introduction

This library, based in C is designed to control the FastCCD detector from Lawrence Berkeley National Laboratory. It controls both camera control functions and data acquisition (frame acquisition). It is separated into two distinct parts, the control part ,`cin_ctl`, and the data (image) part named `cin_data`. It was written in part for use with areaDetector.

Prerequisites

The library relies on the following:

- `libconfig` (Used for nice config files)
- `libpthread` (Used for threading)
- `librt` (Used for time functions)

Installation

Installation of the library is like most unix based source packages:

```
./make
./make doc
./make test
./make install
```

TCP/IP Stack Tuning

In order for the CIN data to operate efficiently, the 10G interface on the host computer needs to be tuned. This needs to be done by adding the following to the file `/etc/sysctl.conf`.

```
# Increase the maximum buffer that user programs can request
# 2147483647 = 2048 Mb
net.core.rmem_max=2147483647
net.core.wmem_max=2147483647
# Set a default value 10 times bigger
net.core.rmem_default=1000000
net.core.wmem_default=1000000
# increase the length of the processor input queue
net.core.netdev_max_backlog = 250000
# recommended for hosts with jumbo frames enabled
net.ipv4.tcp_mtu_probing=1
```

These can be reread by the system without rebooting by entering the command:

```
$sudo sysctl --system
```

Versioning

For the versions available, see the tags on this repository.

Authors

- **Stuart B. Wilkins** - stuwilkins

See also the list of contributors who participated in this project.

License

This project is licensed under the BSD License - see the LICENSE file for details

Acknowledgments

A huge thanks to Peter Dennes, John Joseph and the detector team at LBNL and the team at Sydor Instruments.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Module Documentation

5.1 Cin Control Initialization Routines

Functions

- int cin_ctl_init (cin_ctl_t *cin, char *addr, uint16_t port, uint16_t sport, char *bind_addr, uint16_t bind_port, uint16_t bind_sport)
- int cin_ctl_destroy (cin_ctl_t *cin)
- void cin_ctl_set_msg_callback (cin_ctl_t *cin, cin_ctl_msg_callback callback, void *ptr)
- int cin_data_send_magic (cin_data_t *cin)

5.1.1 Detailed Description

5.1.2 Function Documentation

5.1.2.1 cin_ctl_destroy()

```
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

5.1.2.2 cin_ctl_init()

```
int cin_ctl_init (
            cin_ctl_t * cin,
            char * addr,
            uint16_t port,
            uint16_t sport,
            char * bind_addr,
            uint16_t bind_port,
            uint16_t bind_sport )
```

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
addr	ip address of CIN base address
port	UDP port of cin
sport	stream output UDP port of cin
bind_addr	ip address to bind to
bind_port	input udp port of cin
bind_sport	stream input udp port of cin

Returns

Returns 0 on sucsess non-zero if error

5.1.2.3 cin_ctl_set_msg_callback()

```
void cin_ctl_set_msg_callback (
            cin_ctl_t * cin,
            cin_ctl_msg_callback callback,
            void * ptr )
```

Register a function to recieve status messages

Close connections, free memory and exit library

Parameters

cin	handle to cin library
callback	function pointer to callback function
ptr	user pointer which is passed to callback routine

5.1.2.4 cin_data_send_magic()

```
int cin_data_send_magic (
            cin_data_t * cin )
```

Send a magic packet to the CIN to initialize data

Parameters

cin	handle to cin library

Returns

Returns 0 on sucsess non-zero if error

5.2 Cin Control Read/Rwite Routines

Functions

- 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, unsigned char ∗val, int size)
- int cin_ctl_write_with_readback (cin_ctl_t ∗cin, uint16_t reg, uint16_t val)

5.2.1 Detailed Description

5.2.2 Function Documentation

5.2.2.1 cin_ctl_read()

```
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

5.2.2.2 cin_ctl_stream_write()

```
int cin_ctl_stream_write (
            cin_ctl_t * cin,
            unsigned 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

5.2.2.3 cin_ctl_write()

```
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

5.2.2.4 cin_ctl_write_with_readback()

```
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

5.3 Cin Power Routines

Functions

- 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)

5.3.1 Detailed Description

These routine control power to the CIN for the Frame FPGA and the Front Panel

5.3.2 Function Documentation

5.3.2.1 cin_ctl_fo_test_pattern()

```
int cin_ctl_fo_test_pattern (
            cin_ctl_t * cin,
            int on_off )
```

Control Fiber Optic Interface Test Pattern

Turn on and off the fiber optic test pattern. The FO modules transmit a test pattern to indicate that communication with the data modules is correct. This routine turns on and off the modules. If on_off is 0 then the FO test pattern is turned off, and if on_off is 1 then the FO test pattern is turned on. Note: this routine manipulates the FCRIC mask, so you may need to reconfigure the fCRICs after using it.

cin handle to cin library on_off test pattern status

CIN_OK on sucsess, CIN_ERROR on an error

5.3.2.2 cin_ctl_fp_pwr()

```
int cin_ctl_fp_pwr (
            cin_ctl_t * cin,
            int pwr )
```

Control CIN Front Panel Power

Turn on and off the CIN Front Panel power. The front panel power powers either the fiber optic modules or the LVDS lines to the camera. If pwr is 0 then turn off FP power and if pwr is 1 turn on the FP power.

cin handle to cin library pwr power status

CIN_OK on sucsess, CIN_ERROR on an error

5.3.2.3 cin_ctl_pwr()

```
int cin_ctl_pwr (
            cin_ctl_t * cin,
            int pwr )
```

Control CIN Frame FPGA Power

Turn on and off the frame FPGA power. If pwr is 0 then turn off power. If pwr is 1 turn on power.

cin handle to cin library pwr power status

CIN_OK on sucsess, CIN_ERROR on an error

5.4 CIN Firmware Upload Routines

Functions

- int cin_ctl_load_firmware (cin_ctl_t *cin)
- int cin_ctl_load_firmware_file (cin_ctl_t *cin, char *filename)
- int cin_ctl_load_firmware_data (cin_ctl_t *cin, unsigned char *data, int data_len)
- int cin_ctl_load_config (cin_ctl_t *cin, const char *filename)

5.4.1 Detailed Description

These routines control the upload of firmware to the frame FPGA in the CIN. The firmware can be uploaded using either a external file or an array of unsigned char (bytes). The function cin_ctl_load_firmware() loads the pre-compiled firmware.

5.4.2 Function Documentation

5.4.2.1 cin_ctl_load_config()

```
int cin_ctl_load_config (
            cin_ctl_t * cin,
            const char * filename )
```

Load FPGA config file

Upload a FPGA config file to the CIN. This file is a simple file with each line containing a 4 digit hex value for the register location and a 4 digit hex value for the value to be written to the register.

cin handle to the cin library filename filename to load

CIN_OK on sucsess, CIN_ERROR on an error

5.4.2.2 cin_ctl_load_firmware()

```
int cin_ctl_load_firmware (
            cin_ctl_t * cin )
```

Load the pre-compiled frame FPGA firmware

cin handle to cin library

CIN_OK on sucsess, CIN_ERROR on an error

5.4.2.3 cin_ctl_load_firmware_data()

```
int cin_ctl_load_firmware_data (
             cin_ctl_t * cin,
             unsigned char * data,
             int data_len )
```

Load the frame FPGA firmware from char array

cin handle to cin library data array of binary FPGA firmware data_len length of binary data

CIN_OK on sucsess, CIN_ERROR on an error

5.4.2.4 cin_ctl_load_firmware_file()

```
int cin_ctl_load_firmware_file (
             cin_ctl_t * cin,
             char * filename )
```

Load the frame FPGA firmware from file

cin handle to cin library filename file containing the binary FPGA firmware

CIN_OK on sucsess, CIN_ERROR on an error

5.5 CIN FCLK Configuration Routines

Functions

- int cin_ctl_get_fclk (cin_ctl_t *cin, int *clkfreq)
- int cin_ctl_set_fclk (cin_ctl_t *cin, int clkfreq)

5.5.1 Detailed Description

These routines configure the Internal Frame FPGA Clock (FCLK) frequency.

5.5.2 Function Documentation

5.5.2.1 cin_ctl_get_fclk()

```
int cin_ctl_get_fclk (
              cin_ctl_t * cin,
              int * clkfreq )
```

Get the current frame FPGA clock frequency

cin handle to cin library clkfreq clock frequency

CIN_OK on sucsess, CIN_ERROR on an error

5.5.2.2 cin_ctl_set_fclk()

```
int cin_ctl_set_fclk (
              cin_ctl_t * cin,
              int clkfreq )
```

Set the frame FPGA clock frequency

cin handle to cin library clkfreq clock frequency to set

CIN_OK on sucsess, CIN_ERROR on an error

5.6 CIN Status Routines

Functions

- int cin_ctl_get_id (cin_ctl_t ∗cin, cin_ctl_id_t ∗val)
- int cin_ctl_get_cfg_fpga_status (cin_ctl_t ∗cin, uint16_t ∗_val)
- int **cin_ctl_get_dcm_status** (cin_ctl_t ∗cin, uint16_t ∗_val)
- int **cin_ctl_get_power_status** (cin_ctl_t ∗cin, int full, int ∗pwr, cin_ctl_pwr_mon_t ∗values)

5.6.1 Detailed Description

Group of routines to get the status of the frame and config FPGAs in the CIN.

5.6.2 Function Documentation

5.6.2.1 cin_ctl_get_cfg_fpga_status()

```
int cin_ctl_get_cfg_fpga_status (
            cin_ctl_t * cin,
            uint16_t * _val )
```

cin _val

5.6.2.2 cin_ctl_get_id()

```
int cin_ctl_get_id (
            cin_ctl_t * cin,
            cin_ctl_id_t * val )
```

Get the serial and firmware numbers from the CIN

cin handle to cin library id data structure containing firmware and serial numbers

CIN_OK on sucsess, CIN_ERROR on an error

5.7 CIN Control Bias Routines

Functions

- 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_bias_regs** (cin_ctl_t *cin, uint16_t *vals, int verify)
- int **cin_ctl_get_bias_regs** (cin_ctl_t *cin, uint16_t *vals)
- int **cin_ctl_set_bias_voltages** (cin_ctl_t *cin, float *voltage, int verify)
- int **cin_ctl_get_bias_voltages** (cin_ctl_t *cin, float *voltage, uint16_t *regs)

5.7.1 Detailed Description

Initialization group

5.8 CIN Control Timing Routines

Functions

- int **cin_ctl_set_timing_regs** (cin_ctl_t *cin, uint16_t *vals, int vals_len)
- int **cin_ctl_get_timing_regs** (cin_ctl_t *cin, uint16_t *vals, int vals_len)
- int **cin_config_read_file** (cin_ctl_t *cin, const char *file)
- int cin_config_get_timing_name (cin_ctl_t *cin, int num, char **name)
- int cin_config_get_current_timing_name (cin_ctl_t *cin, char **name)

5.8.1 Detailed Description

Timing setup group

5.8.2 Function Documentation

5.8.2.1 cin_config_get_current_timing_name()

```
int cin_config_get_current_timing_name (
            cin_ctl_t * cin,
            char ** name )
```

Get the name of the current timing mode

cin handle to cin library name char array of name

CIN_OK on sucsess, CIN_ERROR on an error

5.8.2.2 cin_config_get_timing_name()

```
int cin_config_get_timing_name (
            cin_ctl_t * cin,
            int num,
            char ** name )
```

Get the name of the timing config options

cin handle to cin library num number of timing option name char array of name

CIN_OK on sucsess, CIN_ERROR on an error

5.9 CIN Data Initialization Routines

Functions

- int cin_data_init (cin_data_t *cin, char *addr, uint16_t port, char *bind_addr, uint16_t bind_port, int rcvbuf, int packet_buffer_len, int frame_buffer_len, cin_data_callback push_callback, cin_data_callback pop_callback, void *usr_ptr)
- void cin_data_destroy (cin_data_t *cin)

5.9.1 Detailed Description

Initialization group

5.9.2 Function Documentation

5.9.2.1 cin_data_destroy()

```
void cin_data_destroy (
            cin_data_t * cin )
```

Close the cin data library and cleanup

Stop all the processing threads and join them to the main thread. This function blocks until all threads have joined the main thread (program). This should be called to clean up the library before the program is exited

Parameters

cin	Handle to cin data library

5.9.2.2 cin_data_init()

```
int cin_data_init (
            cin_data_t * cin,
            char * addr,
            uint16_t port,
            char * bind_addr,
            uint16_t bind_port,
            int rcvbuf,
            int packet_buffer_len,
            int frame_buffer_len,
            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.

Parameters

cin	Handle to cin data library
addr	IP-Address of cin (if NULL defaults to standard)
port	UDP Port of CIN
bind_addr	IP-Address to bind to (if NULL binds to 0.0.0.0)
bind_port	UDP Port of host
rcvbuf	TCP/IP Kernel recieve buffer size
packet_buffer_len	Length of packet buffer fifo (in units number of packets)
frame_buffer_len	Length of frame (assembler) buffer fifo (in units of number of frames)
push_callback	This function is called when a data structure is needed
pop_callback	This function is called when an image has been processed
usr_ptr	Pointer passed to callback functions

5.10 CIN Data Framestore Functions

Functions

- 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)

5.10.1 Detailed Description

Framestore Group

5.10.2 Function Documentation

5.10.2.1 cin_data_framestore_disable()

```
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

5.10.2.2 cin_data_framestore_skip()

```
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

5.10.2.3 cin_data_framestore_trigger()

```
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 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

5.10.2.4 cin_data_framestore_trigger_enable()

```
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

5.10.2.5 cin_data_get_framestore_counter()

```
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

Chapter 6

Class Documentation

6.1 cin_config_timing Struct Reference

Public Attributes

- uint16_t * data
- int data_len
- char name [CIN_CONFIG_MAX_TIMING_NAME]
- int rows
- int cols
- int overscan
- int fclk_freq
- int framestore

6.1.1 Member Data Documentation

6.1.1.1 cols

```
int cin_config_timing::cols
```

Cols for this timing setup

6.1.1.2 data

```
uint16_t* cin_config_timing::data
```

Pointer to timing data

6.1.1.3 data_len

```
int cin_config_timing::data_len
```

timing data length

6.1.1.4 fclk_freq

```
int cin_config_timing::fclk_freq
```

FCLK Frequency to use

6.1.1.5 framestore

```
int cin_config_timing::framestore
```

Flag (not zero means framestore

6.1.1.6 name

```
char cin_config_timing::name[CIN_CONFIG_MAX_TIMING_NAME]
```

String for config name

6.1.1.7 overscan

```
int cin_config_timing::overscan
```

Number of overscan cols for this setup

6.1.1.8 rows

```
int cin_config_timing::rows
```

Rows for this timing setup

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

- src/cin.h

6.2 cin_ctl Struct Reference

Public Attributes

- char * **addr**
- char * **bind_addr**
- int **port**
- int **bind_port**
- int **sport**
- int **bind_sport**
- cin_port_t **ctl_port**
- cin_port_t **stream_port**
- cin_config_timing_t **timing** [CIN_CONFIG_MAX_TIMING_MODES]
- int **timing_num**
- cin_config_timing_t * **current_timing**
- float fclk_time_factor
- cin_ctl_listener_t * **listener**
- pthread_mutex_t **access**
- pthread_mutexattr_t **access_attr**
- void(* **msg_callback**)(const char *, int, void *)
- void * **msg_callback_ptr**

6.2.1 Member Data Documentation

6.2.1.1 fclk_time_factor

```
float cin_ctl::fclk_time_factor
```

In micro seconds

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

- src/cin.h

6.3 cin_ctl_id Struct Reference

Public Attributes

- uint16_t **base_board_id**
- uint16_t **base_serial_no**
- uint16_t **base_fpga_ver**
- uint16_t **fabric_board_id**
- uint16_t **fabric_serial_no**
- uint16_t **fabric_fpga_ver**

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

- src/cin.h

6.4 cin_ctl_listener Struct Reference

Public Attributes

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

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

- src/cin.h

6.5 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

6.6 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

6.7 cin_data Struct Reference

Public Attributes

- fifo * **packet_fifo**
- fifo * **frame_fifo**
- cin_data_threads_t **listen_thread**
- cin_data_threads_t **assembler_thread**
- cin_data_threads_t **descramble_thread**
- pthread_mutex_t **descramble_mutex**
- pthread_mutex_t **stats_mutex**
- pthread_mutex_t **framestore_mutex**
- cin_data_callbacks_t **callbacks**
- char * **addr**
- char * **bind_addr**
- int **port**

- int **bind_port**
- int **recv_buf**
- cin_port_t **dp**
- struct timespec **framerate**
- unsigned long int **dropped_packets**
- unsigned long int **mallformed_packets**
- uint16_t **last_frame**
- cin_data_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

6.8 cin_data_callbacks Struct Reference

Public Attributes

- void ∗(∗ **push**)(cin_data_frame_t ∗, void ∗usr_ptr)
- void ∗(∗ **pop**)(cin_data_frame_t ∗, void ∗usr_ptr)
- cin_data_frame_t ∗ **frame**
- void ∗ **usr_ptr**

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

- src/cin.h

6.9 cin_data_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

6.10 cin_data_frame Struct Reference

Public Attributes

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

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

- src/cin.h

6.11 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

6.12 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

6.13 cin_data_stats Struct Reference

Public Attributes

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

6.14 cin_data_threads Struct Reference

Public Attributes

- pthread_t **thread_id**
- pthread_barrier_t **barrier**
- int **started**

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

- src/cin.h

6.15 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

6.16 cin_port Struct Reference

Public Attributes

- int **sockfd**
- struct timeval **tv**
- struct sockaddr_in **sin_srv**
- struct sockaddr_in **sin_cli**
- socklen_t **slen**

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

- src/cin.h

6.17 cin_timing_state Struct Reference

`#include <cin.h>`

Public Attributes

- uint8_t passes_per_state
- uint8_t next_state
- uint32_t loop_back_counter
- uint8_t loop_state
- uint8_t total_ticks
- uint8_t initial_state [15]
- uint8_t edge1 [15]
- uint8_t edge2 [15]
- uint8_t **spare1**
- uint8_t **spare2**

6.17.1 Detailed Description

CIN CCD Timing state

Each timing state is made up of 52 parameters

0 passes_per_state 1 next_state 3 loop_backs_for_state When not zero go to loop_state and subtract 1 4 loop_↩ state 5 ccd_clock_cnt_end Number of clock counts for 1 pass of this state 6 -20 initial_clock_value[15] There are 8 vertical, 4 horizontal, convert, save_data and spare 21-35 clock_edge1[15] After this number of ticks the clock signal is inverted from initial_clock_value 36-50 clock_edge2[15] After this number of ticks the clock signal is reverted to

6.17.2 Member Data Documentation

6.17.2.1 edge1

```
uint8_t cin_timing_state::edge1[15]
```

Number of ticks to wait before inverting clock state

6.17.2.2 edge2

```
uint8_t cin_timing_state::edge2[15]
```

Number of ticks to wait before returning to initial_state

6.17.2.3 initial_state

```
uint8_t cin_timing_state::initial_state[15]
```

Initial clock values

6.17.2.4 loop_back_counter

```
uint32_t cin_timing_state::loop_back_counter
```

Number of jumps to loop_state

6.17.2.5 loop_state

```
uint8_t cin_timing_state::loop_state
```

State to jump to when loop_state is non zero

6.17.2.6 next_state

```
uint8_t cin_timing_state::next_state
```

State to jump to upon completion

6.17.2.7 passes_per_state

```
uint8_t cin_timing_state::passes_per_state
```

Number of times to pass through this state

6.17.2.8 total_ticks

```
uint8_t cin_timing_state::total_ticks
```

Total number of ticks for this state

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

- src/cin.h

6.18 fifo Struct Reference

Public Attributes

- void * **data**
- void * **head**
- void * **tail**
- void * **end**
- 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 7

File Documentation

7.1 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 fifo
- struct cin_ctl_listener
- struct cin_port
- struct cin_timing_state
- struct cin_config_timing
- struct cin_ctl
- struct cin_data_frame
- struct cin_data_stats
- struct cin_data_threads
- struct cin_data_callbacks
- struct cin_data_descramble_map_t
- struct cin_data
- struct cin_ctl_id
- struct cin_ctl_pwr_val
- struct cin_ctl_pwr_mon_t

Macros

- #define **CIN_OK** 0
- #define **CIN_ERROR** -1
- #define **CIN_CTL_MSG_OK** 0
- #define **CIN_CTL_MSG_MINOR** 1
- #define **CIN_CTL_MSG_MAJOR** 2
- #define **CIN_CTL_IP** "192.168.1.207"
- #define **CIN_CTL_CIN_PORT** 49200
- #define **CIN_CTL_BIND_PORT** 50200
- #define **CIN_CTL_FRMW_CIN_PORT** 49202
- #define **CIN_CTL_FRMW_BIND_PORT** 50202
- #define **CIN_CTL_RCVBUF** 10
- #define **CIN_CTL_MAX_READ_TRIES** 5
- #define **CIN_CTL_MAX_WRITE_TRIES** 5
- #define **CIN_CTL_WRITE_SLEEP** 100
- #define **CIN_CTL_READ_SLEEP** 100
- #define **CIN_CTL_BIAS_SLEEP** 100000
- #define **CIN_CTL_FO_SLEEP** 500000
- #define **CIN_CTL_CONFIG_SLEEP** 100
- #define **CIN_CTL_DCO_SLEEP** 1000000
- #define **CIN_CTL_FCLK_SLEEP** 200000
- #define **CIN_CTL_STREAM_CHUNK** 512
- #define **CIN_CTL_STREAM_SLEEP** 5
- #define **CIN_CTL_PACKET_WAIT** 1000
- #define **CIN_CTL_PACKET_LOOPS** 250
- #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_DATA_IP** "10.0.5.207"
- #define **CIN_DATA_BIND_PORT** 49201
- #define **CIN_DATA_CIN_PORT** 49203
- #define **CIN_DATA_FRAME_BUFFER_LEN** 1000
- #define **CIN_DATA_PACKET_BUFFER_LEN** 10000
- #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∗1024∗1024)
- #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_CTL_NUM_BIAS** 20
- #define CIN_CTL_BIAS_OFFSET 0x0030
- #define **CIN_CTL_BIAS_POSH** 0
- #define **CIN_CTL_BIAS_NEGH** 1
- #define **CIN_CTL_BIAS_POSRG** 2
- #define **CIN_CTL_BIAS_NEGRG** 3
- #define **CIN_CTL_BIAS_POSSW** 4
- #define **CIN_CTL_BIAS_NEGSW** 5
- #define **CIN_CTL_BIAS_POSV** 6
- #define **CIN_CTL_BIAS_NEGV** 7
- #define **CIN_CTL_BIAS_POSTG** 8
- #define **CIN_CTL_BIAS_NEGTG** 9
- #define **CIN_CTL_BIAS_POSVF** 10
- #define **CIN_CTL_BIAS_NEGVF** 11
- #define **CIN_CTL_BIAS_NEDGE** 12
- #define **CIN_CTL_BIAS_OTG** 13
- #define **CIN_CTL_BIAS_VDDR** 14
- #define **CIN_CTL_BIAS_VDD_OUT** 15
- #define **CIN_CTL_BIAS_BUF_BASE** 16
- #define **CIN_CTL_BIAS_BUF_DELTA** 17
- #define **CIN_CTL_BIAS_SPARE1** 18
- #define **CIN_CTL_BIAS_SPARE2** 19
- #define **DEBUG_PRINT**(fmt, ...) if(_debug_print_flag) { fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, __LI↩ NE__, __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__, __LI↩ NE__, __func__); }
- #define **ERROR_PRINT**(fmt, ...) if(_error_print_flag) { fprintf(stderr, "%s:%d:%s(): " fmt, __FILE__, __LIN↩ E__, __func__, __VA_ARGS__); }
- #define **CIN_CONFIG_MAX_STRING** 40
- #define CIN_CONFIG_MAX_TIMING_DATA 880
- #define CIN_CONFIG_MAX_TIMING_MODES 10
- #define CIN_CONFIG_MAX_TIMING_NAME 40

Typedefs

- typedef struct cin_ctl_listener **cin_ctl_listener_t**
- typedef struct cin_port **cin_port_t**
- typedef struct cin_timing_state cin_timing_state_t
- typedef struct cin_config_timing **cin_config_timing_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 ∗, void ∗usr_ptr)
- typedef void(∗ **cin_ctl_msg_callback**) (const char ∗, int, void ∗)
- 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, char *addr, uint16_t port, uint16_t sport, char *bind_addr, uint16_t bind_port, uint16_t bind_sport)
- int cin_ctl_destroy (cin_ctl_t *cin)
- void cin_ctl_set_msg_callback (cin_ctl_t *cin, cin_ctl_msg_callback callback, void *ptr)
- int cin_data_send_magic (cin_data_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, unsigned 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_firmware (cin_ctl_t *cin)
- int cin_ctl_load_firmware_file (cin_ctl_t *cin, char *filename)
- int cin_ctl_load_firmware_data (cin_ctl_t *cin, unsigned char *data, int data_len)
- int cin_ctl_load_config (cin_ctl_t *cin, const char *filename)
- int cin_ctl_get_fclk (cin_ctl_t *cin, int *clkfreq)
- int cin_ctl_set_fclk (cin_ctl_t *cin, int clkfreq)
- int cin_ctl_get_id (cin_ctl_t *cin, cin_ctl_id_t *val)
- int cin_ctl_get_cfg_fpga_status (cin_ctl_t *cin, uint16_t *_val)
- int **cin_ctl_get_dcm_status** (cin_ctl_t *cin, uint16_t *_val)
- int **cin_ctl_get_power_status** (cin_ctl_t *cin, int full, int *pwr, cin_ctl_pwr_mon_t *values)
- 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_bias_regs** (cin_ctl_t *cin, uint16_t *vals, int verify)
- int **cin_ctl_get_bias_regs** (cin_ctl_t *cin, uint16_t *vals)
- int **cin_ctl_set_bias_voltages** (cin_ctl_t *cin, float *voltage, int verify)
- int **cin_ctl_get_bias_voltages** (cin_ctl_t *cin, float *voltage, uint16_t *regs)
- int **cin_ctl_set_timing_regs** (cin_ctl_t *cin, uint16_t *vals, int vals_len)
- int **cin_ctl_get_timing_regs** (cin_ctl_t *cin, uint16_t *vals, int vals_len)
- 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_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_clamp** (cin_ctl_t *cin, int clamp)

- int **cin_ctl_set_fcric_gain** (cin_ctl_t *cin, int gain)
- int **cin_ctl_set_fcric_regs** (cin_ctl_t *cin, uint16_t *reg, int num_reg)
- int **cin_ctl_set_fcric** (cin_ctl_t *cin)
- int **cin_ctl_set_fabric_address** (cin_ctl_t *cin, char *ip)
- int **cin_ctl_bias_dump** (cin_ctl_t *cin, FILE *fp)
- int **cin_ctl_reg_dump** (cin_ctl_t *cin, FILE *fp)
- int **cin_config_read_file** (cin_ctl_t *cin, const char *file)
- int cin_config_get_timing_name (cin_ctl_t *cin, int num, char **name)
- int cin_config_get_current_timing_name (cin_ctl_t *cin, char **name)
- void **cin_ctl_message** (cin_ctl_t *cin, const char *message, int severity)
- int cin_data_init (cin_data_t *cin, char *addr, uint16_t port, char *bind_addr, uint16_t bind_port, int rcvbuf, int packet_buffer_len, int frame_buffer_len, cin_data_callback push_callback, cin_data_callback pop_callback, void *usr_ptr)
- void cin_data_destroy (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)
- int **cin_com_boot** (cin_ctl_t *cin_ctl, cin_data_t *cin_data, int mode)
- int **cin_com_set_timing** (cin_ctl_t *cin_ctl, cin_data_t *cin_data, int mode)
- int **cin_config_find_timing** (cin_ctl_t *cin, const char *name)
- int **cin_ctl_upload_bias** (cin_ctl_t *cin)

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**

7.1.1 Detailed Description

Author

Stuart B. Wilkins swilkins@bnl.gov

7.1.2 LICENSE

7.1.3 DESCRIPTION

Header file for CIN communications

7.1.4 Macro Definition Documentation

7.1.4.1 CIN_CONFIG_MAX_TIMING_DATA

```
#define CIN_CONFIG_MAX_TIMING_DATA 880
```

Max = 55 per state, 16 states

7.1.4.2 CIN_CONFIG_MAX_TIMING_MODES

```
#define CIN_CONFIG_MAX_TIMING_MODES 10
```

states max

7.1.4.3 CIN_CONFIG_MAX_TIMING_NAME

```
#define CIN_CONFIG_MAX_TIMING_NAME 40
```

Max characters for timing name

7.1.4.4 CIN_CTL_BIAS_OFFSET

```
#define CIN_CTL_BIAS_OFFSET 0x0030
```

Offset in address to read bias

7.1.5 Typedef Documentation

7.1.5.1 cin_timing_state_t

```
typedef struct cin_timing_state cin_timing_state_t
```

CIN CCD Timing state

Each timing state is made up of 52 parameters

0 passes_per_state 1 next_state 3 loop_backs_for_state When not zero go to loop_state and subtract 1 4 loop_↩ state 5 ccd_clock_cnt_end Number of clock counts for 1 pass of this state 6 -20 initial_clock_value[15] There are 8 vertical, 4 horizontal, convert, save_data and spare 21-35 clock_edge1[15] After this number of ticks the clock signal is inverted from initial_clock_value 36-50 clock_edge2[15] After this number of ticks the clock signal is reverted to

7.2 src/cin_register_map.h File Reference

Macros

- #define REG_COMMAND 0x0001
- #define **REG_READ_ADDRESS** 0x0002
- #define REG_STREAM_TYPE 0x0003
- #define **CMD_FCLK_125** 0xB000
- #define **CMD_FCLK_200** 0x7000
- #define CMD_FCLK_250 0x3000
- #define **REG_IF_MAC0** 0x0010
- #define **REG_IF_MAC1** 0x0011
- #define **REG_IF_MAC2** 0x0012
- #define **REG_IF_IP0** 0x0013
- #define **REG_IF_IP1** 0x0014
- #define **REG_IF_CMD_PORT_NUM** 0x001A
- #define **REG_IF_STREAM_IN_PORT_NUM** 0x001C
- #define **REG_IF_STREAM_OUT_PORT_NUM** 0x001D
- #define REG_ETH_RESET 0x0020
- #define REG_ETH_ENABLE 0x0021

- #define REG_PHY1_MDIO_CMD 0x0022
- #define **REG_PHY1_MDIO_CMD_DATA** 0x0023
- #define **REG_PHY1_MDIO_STATUS** 0x0024
- #define **REG_PHY1_MDIO_RD_ADDR** 0x0025
- #define **REG_PHY1_MDIO_RD_DATA** 0x0026
- #define REG_MAC_CFG_VECTOR1 0x0027
- #define **REG_PHY2_MDIO_CMD** 0x0028
- #define **REG_PHY2_MDIO_CMD_DATA** 0x0029
- #define **REG_PHY2_MDIO_STATUS** 0x002A
- #define **REG_PHY2_MDIO_RD_ADDR** 0x002B
- #define **REG_PHY2_MDIO_RD_DATA** 0x002C
- #define REG_MAC_CFG_VECTOR2 0x002D
- #define CMD_PS_ENABLE 0x0021
- #define CMD_PS_POWERDOWN 0x0022
- #define REG_PS_ENABLE 0x0030
- #define REG_PS_SYNC_DIV0 0x0031
- #define REG_PS_SYNC_DIV1 0x0032
- #define REG_PS_SYNC_DIV2 0x0033
- #define REG_PS_SYNC_DIV3 0x0034
- #define REG_PS_SYNC_DIV4 0x0035
- #define **CMD_PROGRAM_FRAME** 0x0041
- #define REG_FRM_RESET 0x0036
- #define REG_FRM_10GbE_SEL 0x0037;
- #define CMD_ENABLE_CLKS 0x0031
- #define CMD_DISABLE_CLKS 0x0032
- #define REG_CLOCK_EN_REG 0x0038
- #define **REG_SI570_REG0** 0x0039
- #define **REG_SI570_REG1** 0x003A
- #define **REG_SI570_REG2** 0x003B
- #define REG_SI570_REG3 0x003C
- #define CMD_MON_STOP 0x0011
- #define CMD_MON_START 0x0012
- #define **REG_VMON_ADC1_CH1** 0x0040 /∗ V12P_BUS Voltage Monitor ∗/
- #define **REG_IMON_ADC1_CH0** 0x0041 /∗ V12P_BUS Current Monitor ∗/
- #define **REG_VMON_ADC0_CH5** 0x0042 /∗ V3P3_MGMT Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH5** 0x0043 /∗ V3P3_MGMT Current Monitor ∗/
- #define **REG_VMON_ADC0_CH4** 0x0044 /∗ V3P3_S3E Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH4** 0x0045 /∗ V3P3_S3E Current Monitor ∗/
- #define **REG_VMON_ADC0_CH7** 0x0046 /∗ V2P5_MGMT Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH7** 0x0047 /∗ V2P5_MGMT Current Monitor ∗/
- #define **REG_VMON_ADC0_CH6** 0x0048 /∗ V1P8_MGMT Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH6** 0x0049 /∗ V1P8_MGMT Current Monitor ∗/
- #define **REG_VMON_ADC0_CH2** 0x004A /∗ V1P2_MGMT Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH2** 0x004B /∗ V1P2_MGMT Current Monitor ∗/
- #define **REG_VMON_ADC0_CH3** 0x004C /∗ V1P0_ENET Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH3** 0x004D /∗ V1P0_ENET Current Monitor ∗/
- #define **REG_VMON_ADC0_CH8** 0x004E /∗ V3P3_GEN Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH8** 0x004F /∗ V3P3_GEN Current Monitor ∗/
- #define **REG_VMON_ADC0_CH9** 0x0050 /∗ V2P5_GEN Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CH9** 0x0051 /∗ V2P5_GEN Current Monitor ∗/
- #define **REG_VMON_ADC0_CHE** 0x0052 /∗ V0P9_V6 Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CHE** 0x0053 /∗ V0P9_V6 Current Monitor ∗/
- #define **REG_VMON_ADC0_CHD** 0x0054 /∗ V2P5_V6 Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CHD** 0x0055 /∗ V2P5_V6 Current Monitor ∗/
- #define **REG_VMON_ADC0_CHB** 0x0056 /∗ V1P0_V6 Voltage Monitor ∗/

- #define **REG_IMON_ADC0_CHB** 0x0057 /∗ V1P0_V6 Current Monitor ∗/
- #define **REG_VMON_ADC0_CHC** 0x0058 /∗ V1P2_V6 Voltage Monitor ∗/
- #define **REG_IMON_ADC0_CHC** 0x0059 /∗ V1P2_V6 Current Monitor ∗/
- #define **REG_VMON_ADC0_CHF** 0x005A /∗ V5P0_FP Voltage Monitor (1/2) ∗/
- #define REG_IMON_ADC0_CHF 0x005B /∗ V5P0_FP Current Monitor (1/2) ∗/
- #define **REG_DCM_STATUS** 0x0080
- #define **REG_FPGA_STATUS** 0x0081
- #define **REG_BOARD_ID** 0x008D
- #define **REG_HW_SERIAL_NUM** 0x008E
- #define REG_FPGA_VERSION 0x008F
- #define **REG_SANDBOX_REG00** 0x00F0
- #define **REG_SANDBOX_REG01** 0x00F1
- #define **REG_SANDBOX_REG02** 0x00F2
- #define **REG_SANDBOX_REG03** 0x00F3
- #define **REG_SANDBOX_REG04** 0x00F4
- #define **REG_SANDBOX_REG05** 0x00F5
- #define **REG_SANDBOX_REG06** 0x00F6
- #define **REG_SANDBOX_REG07** 0x00F7
- #define **REG_SANDBOX_REG08** 0x00F8
- #define **REG_SANDBOX_REG09** 0x00F9
- #define **REG_SANDBOX_REG0A** 0x00FA
- #define **REG_SANDBOX_REG0B** 0x00FB
- #define **REG_SANDBOX_REG0C** 0x00FC
- #define **REG_SANDBOX_REG0D** 0x00FD
- #define **REG_SANDBOX_REG0E** 0x00FE
- #define REG_SANDBOX_REG0F 0x00FF
- #define **REG_FRM_COMMAND** 0x8001
- #define **REG_FRM_READ_ADDRESS** 0x8002
- #define REG_FRM_STREAM_TYPE 0x8003
- #define CMD_SEND_SYNC_PULSE 0x0100
- #define CMD_SYNC_DETECTOR2READOUT 0x0101
- #define CMD_WR_CCD_BIAS_REG 0x0102
- #define CMD_WR_CCD_CLOCK_REG 0x0103
- #define CMD_SEND_FCRIC_CONFIG 0x0105
- #define CMD_RESET_FRAME_COUNT 0x0106
- #define **REG_IF_MAC_FAB1B0** 0x8010
- #define **REG_IF_MAC_FAB1B1** 0x8011
- #define **REG_IF_MAC_FAB1B2** 0x8012
- #define **REG_IF_IP_FAB1B0** 0x8013
- #define **REG_IF_IP_FAB1B1** 0x8014
- #define **REG_IF_CMD_PORT_NUM_FAB1B** 0x8015
- #define **REG_IF_STREAM_IN_PORT_NUM_FAB1B** 0x8016
- #define **REG_IF_STREAM_OUT_PORT_NUM_FAB1B** 0x8017
- #define **REG_XAUI_FAB1B** 0x8018
- #define **REG_MAC_CONFIG_VEC_FAB1B0** 0x8019
- #define **REG_MAC_CONFIG_VEC_FAB1B1** 0x801A
- #define **REG_MAC_STATS1_FAB1B0** 0x801B
- #define **REG_MAC_STATS1_FAB1B1** 0x801C
- #define **REG_MAC_STATS2_FAB1B0** 0x801D
- #define **REG_MAC_STATS2_FAB1B1** 0x801E
- #define **REG_IF_MAC_FAB2B0** 0x8020
- #define **REG_IF_MAC_FAB2B1** 0x8021
- #define **REG_IF_MAC_FAB2B2** 0x8022
- #define **REG_IF_IP_FAB2B0** 0x8023
- #define **REG_IF_IP_FAB2B1** 0x8024

- #define **REG_IF_CMD_PORT_NUM_FAB2B** 0x8025
- #define **REG_IF_STREAM_IN_PORT_NUM_FAB2B** 0x8026
- #define **REG_IF_STREAM_OUT_PORT_NUM_FAB2B** 0x8027
- #define **REG_XAUI_FAB2B** 0x8028
- #define **REG_MAC_CONFIG_VEC_FAB2B0** 0x8029
- #define **REG_MAC_CONFIG_VEC_FAB2B1** 0x802A
- #define **REG_MAC_STATS1_FAB2B0** 0x802B
- #define **REG_MAC_STATS1_FAB2B1** 0x802C
- #define **REG_MAC_STATS2_FAB2B0** 0x802D
- #define REG_MAC_STATS2_FAB2B1 0x802E
- #define REG_SRAM_COMMAND 0x8030
- #define **REG_SRAM_START_ADDR1** 0x8031
- #define **REG_SRAM_START_ADDR0** 0x8032
- #define **REG_SRAM_STOP_ADDR1** 0x8033
- #define **REG_SRAM_STOP_ADDR0** 0x8034
- #define **REG_SRAM_FRAME_DATA_OUT1** 0x8035
- #define **REG_SRAM_FRAME_DATA_OUT0** 0x8036
- #define **REG_SRAM_FRAME_DATA_IN1** 0x8037
- #define **REG_SRAM_FRAME_DATA_IN0** 0x8038
- #define **REG_SRAM_FRAME_DV** 0x8039
- #define **REG_SRAM_STATUS1** 0x803A
- #define REG_SRAM_STATUS0 0x803B
- #define CMD_FCLK_COMMIT 0x0012
- #define REG_FCLK_I2C_ADDRESS 0x8040
- #define REG_FCLK_I2C_DATA_WR 0x8041
- #define REG_FCLK_I2C_DATA_RD 0x8042
- #define **REG_TRIGGERSELECT_REG** 0x8050
- #define REG_TRIGGERMASK_REG 0x8051
- #define **REG_CCDFCLKSELECT_REG** 0x8052
- #define **REG_CDICLKDISABLE_REG** 0x8053
- #define **REG_FCLK_SET0** 0xB007
- #define **REG_FCLK_SET1** 0xB008
- #define **REG_FCLK_SET2** 0xB009
- #define **REG_FCLK_SET3** 0xB00A
- #define **REG_FCLK_SET4** 0xB00B
- #define REG_FCLK_SET5 0xB00C
- #define **REG_FRM_DCM_STATUS** 0x8080
- #define **REG_FRM_FPGA_STATUS** 0x8081
- #define **REG_FRM_BOARD_ID** 0x808D
- #define **REG_FRM_HW_SERIAL_NUM** 0x808E
- #define REG_FRM_FPGA_VERSION 0x808F
- #define **REG_FRM_SANDBOX_REG00** 0x80F0
- #define **REG_FRM_SANDBOX_REG01** 0x80F1
- #define **REG_FRM_SANDBOX_REG02** 0x80F2
- #define **REG_FRM_SANDBOX_REG03** 0x80F3
- #define **REG_FRM_SANDBOX_REG04** 0x80F4
- #define **REG_FRM_SANDBOX_REG05** 0x80F5
- #define **REG_FRM_SANDBOX_REG06** 0x80F6
- #define **REG_FRM_SANDBOX_REG07** 0x80F7
- #define **REG_FRM_SANDBOX_REG08** 0x80F8
- #define **REG_FRM_SANDBOX_REG09** 0x80F9
- #define **REG_FRM_SANDBOX_REG0A** 0x80FA
- #define **REG_FRM_SANDBOX_REG0B** 0x80FB
- #define **REG_FRM_SANDBOX_REG0C** 0x80FC
- #define **REG_FRM_SANDBOX_REG0D** 0x80FD

- #define **REG_FRM_SANDBOX_REG0E** 0x80FE
- #define REG_FRM_SANDBOX_REG0F 0x80FF
- #define **REG_DETECTOR_REVISION_REG** 0x8100
- #define **REG_DETECTOR_CONFIG_REG1** 0x8101
- #define **REG_DETECTOR_CONFIG_REG2** 0x8102
- #define **REG_DETECTOR_CONFIG_REG3** 0x8103
- #define **REG_DETECTOR_CONFIG_REG4** 0x8104
- #define **REG_DETECTOR_CONFIG_REG5** 0x8105
- #define **REG_DETECTOR_CONFIG_REG6** 0x8106
- #define **REG_DETECTOR_CONFIG_REG7** 0x8107
- #define **REG_DETECTOR_CONFIG_REG8** 0x8108
- #define **REG_IMG_PROC_REVISION_REG** 0x8120
- #define **REG_IMG_PROC_CONFIG_REG1** 0x8121
- #define **REG_IMG_PROC_CONFIG_REG2** 0x8122
- #define **REG_IMG_PROC_CONFIG_REG3** 0x8123
- #define **REG_IMG_PROC_CONFIG_REG4** 0x8124
- #define **REG_IMG_PROC_CONFIG_REG5** 0x8125
- #define **REG_IMG_PROC_CONFIG_REG6** 0x8126
- #define **REG_IMG_PROC_CONFIG_REG7** 0x8127
- #define **REG_IMG_PROC_CONFIG_REG8** 0x8128
- #define **REG_BIASANDCLOCKREGISTERADDRESS** 0x8200
- #define **REG_BIASANDCLOCKREGISTERDATA** 0x8201
- #define **REG_CLOCKREGISTERDATAOUT** 0x8202
- #define REG_BIASREGISTERDATAOUT 0x8203
- #define REG_BIASCONFIGREGISTER0_REG 0x8204
- #define REG_CLOCKCONFIGREGISTER0_REG 0x8205
- #define **REG_BIASPARAM_READ_START** 0x3000
- #define REG_EXPOSURETIMEMSB_REG 0x8206
- #define REG_EXPOSURETIMELSB_REG 0x8207
- #define **REG_ALTEXPOSURETIMEMSB_REG** 0x8306
- #define **REG_ALTEXPOSURETIMELSB_REG** 0x8307
- #define REG_TRIGGERREPETITIONTIMEMSB_REG 0x8208
- #define REG_TRIGGERREPETITIONTIMELSB_REG 0x8209
- #define **REG_DELAYTOEXPOSUREMSB_REG** 0x820A
- #define **REG_DELAYTOEXPOSURELSB_REG** 0x820B
- #define **REG_NUMBEROFEXPOSURE_REG** 0x820C
- #define **REG_SHUTTERTIMEMSB_REG** 0x820D
- #define **REG_SHUTTERTIMELSB_REG** 0x820E
- #define **REG_DELAYTOSHUTTERMSB_REG** 0x820F
- #define REG_DELAYTOSHUTTERLSB_REG 0x8210
- #define **REG_FCRIC_MASK_REG1** 0x8211
- #define **REG_FCRIC_MASK_REG2** 0x8212
- #define **REG_FCRIC_MASK_REG3** 0x8213
- #define **REG_LVDS_OVERFLOW_ERROR_REG1** 0x8214
- #define **REG_LVDS_OVERFLOW_ERROR_REG2** 0x8215
- #define **REG_LVDS_OVERFLOW_ERROR_REG3** 0x8216
- #define **REG_LVDS_PARITY_ERROR_REG1** 0x8217
- #define **REG_LVDS_PARITY_ERROR_REG2** 0x8218
- #define **REG_LVDS_PARITY_ERROR_REG3** 0x8219
- #define **REG_LVDS_STOP_BIT_ERROR_REG1** 0x821A
- #define **REG_LVDS_STOP_BIT_ERROR_REG2** 0x821B
- #define **REG_LVDS_STOP_BIT_ERROR_REG3** 0x821C
- #define **REG_FCRIC_WRITE0_REG** 0x821D
- #define **REG_FCRIC_WRITE1_REG** 0x821E
- #define **REG_FCRIC_WRITE2_REG** 0x821F

- #define **REG_FCRIC_READ0_REG** 0x8220
- #define **REG_FCRIC_READ1_REG** 0x8221
- #define **REG_FCRIC_READ2_REG** 0x8222
- #define **REG_DEBUGVIDEO0_REG** 0x8223
- #define **REG_DEBUGVIDEO1_REG** 0x8224
- #define **REG_DEBUGVIDEO2_REG** 0x8225
- #define **REG_DEBUGVIDEO3_REG** 0x8226
- #define **REG_DEBUGVIDEO4_REG** 0x8227
- #define **REG_DEBUGVIDEO5_REG** 0x8228
- #define **REG_DEBUGVIDEO6_REG** 0x8229
- #define **REG_DEBUGVIDEO7_REG** 0x822A
- #define **REG_DEBUGVIDEO8_REG** 0x822B
- #define **REG_DEBUGVIDEO9_REG** 0x822C
- #define **REG_DEBUGVIDEO10_REG** 0x822D
- #define **REG_DEBUGVIDEO11_REG** 0x822E
- #define **REG_DEBUGCOUNTER00_REG** 0x822F
- #define **REG_DEBUGCOUNTER01_REG** 0x8230
- #define **REG_DEBUGCOUNTER02_REG** 0x8231
- #define **REG_DEBUGCOUNTER03_REG** 0x8232
- #define REG_DEBUGCOUNTER04_REG 0x8233
- #define CMD_READ_REG 0x0001

7.2.1 Detailed Description

<

Author

Stuart B. Wilkins swilkins@bnl.gov

7.2.2 LICENSE

7.2.3 DESCRIPTION

Control and Frame FPGA Register Map

7.2.4 TIMING

The exposure time is set through the REG_EXPOSURETIMEMSB_REG and REG_EXPOSURETIMELSB_REG registers. Their value in wall time depends on the fclk frequency. At 200 MHz fclk a register value of 0x00000001 corresponds to 20 us. At 125 MHz, a value of 0x00000001 corresponds to 32 us.

7.2.5 Macro Definition Documentation

7.2.5.1 CMD_DISABLE_CLKS

```
#define CMD_DISABLE_CLKS 0x0032
```

Disable Frame FPGA clock crystals

7.2.5.2 CMD_ENABLE_CLKS

```
#define CMD_ENABLE_CLKS 0x0031
```

Enable selected Frame FPGA clock crystals

7.2.5.3 CMD_FCLK_250

```
#define CMD_FCLK_250 0x3000
```

Ethernet Interface

7.2.5.4 CMD_FCLK_COMMIT

```
#define CMD_FCLK_COMMIT 0x0012
```

Start I2C Write/Read

7.2.5.5 CMD_MON_START

```
#define CMD_MON_START 0x0012
```

Start voltage and current monitor

7.2.5.6 CMD_MON_STOP

```
#define CMD_MON_STOP 0x0011
```

Stop voltage and current monitor

7.2.5.7 CMD_PS_ENABLE

```
#define CMD_PS_ENABLE 0x0021
```

Enable Selected Power Modules

7.2.5.8 CMD_PS_POWERDOWN

```
#define CMD_PS_POWERDOWN 0x0022
```

Start power down sequence

7.2.5.9 CMD_READ_REG

```
#define CMD_READ_REG 0x0001
```

Read Register CIN_REGISTER_MAP_H

7.2.5.10 CMD_RESET_FRAME_COUNT

```
#define CMD_RESET_FRAME_COUNT 0x0106
```

RESET STATISTICS/DEBUG COUNTERS Ethernet Interface

7.2.5.11 CMD_SEND_FCRIC_CONFIG

```
#define CMD_SEND_FCRIC_CONFIG 0x0105
```

SEND CONFIG DATA TO FRIC

7.2.5.12 CMD_SEND_SYNC_PULSE

```
#define CMD_SEND_SYNC_PULSE 0x0100
```

ISSUES A SYNC PULSE

7.2.5.13 CMD_SYNC_DETECTOR2READOUT

```
#define CMD_SYNC_DETECTOR2READOUT 0x0101
```

COMMAND TO SYNC DETECTOR AND READOUT (SEE IMAGE PROCESSING)

7.2.5.14 CMD_WR_CCD_BIAS_REG

```
#define CMD_WR_CCD_BIAS_REG 0x0102
```

WRITE CCD BIAS REGISTERS

7.2.5.15 CMD_WR_CCD_CLOCK_REG

```
#define CMD_WR_CCD_CLOCK_REG 0x0103
```

WRITE CCD CLOCK REGISTER

7.2.5.16 REG_BIASCONFIGREGISTER0_REG

```
#define REG_BIASCONFIGREGISTER0_REG 0x8204
```

Clock Static Registers

7.2.5.17 REG_BIASREGISTERDATAOUT

```
#define REG_BIASREGISTERDATAOUT 0x8203
```

Bias Static Registers

7.2.5.18 REG_CLOCK_EN_REG

```
#define REG_CLOCK_EN_REG 0x0038
```

Clock Enable Register Programmable Si570 Clock Registers

7.2.5.19 REG_CLOCKCONFIGREGISTER0_REG

```
#define REG_CLOCKCONFIGREGISTER0_REG 0x8205
```

Bias Voltage

7.2.5.20 REG_COMMAND

```
#define REG_COMMAND 0x0001
```

<Command Registers

7.2.5.21 REG_DEBUGCOUNTER04_REG

```
#define REG_DEBUGCOUNTER04_REG 0x8233
```

==========

CIN Commands

Common Commands

7.2.5.22 REG_DELAYTOSHUTTERLSB_REG

```
#define REG_DELAYTOSHUTTERLSB_REG 0x8210
```

Digitizer Registers

7.2.5.23 REG_ETH_ENABLE

```
#define REG_ETH_ENABLE 0x0021
```

Enable Eth Hardware 1=Rx, 2=Tx, 3=Both

7.2.5.24 REG_ETH_RESET

```
#define REG_ETH_RESET 0x0020
```

Reset Eth Hardware 1=Rx, 2=Tx, 3=Both

7.2.5.25 REG_EXPOSURETIMELSB_REG

```
#define REG_EXPOSURETIMELSB_REG 0x8207
```

Exposure time LSB

7.2.5.26 REG_EXPOSURETIMEMSB_REG

```
#define REG_EXPOSURETIMEMSB_REG 0x8206
```

Exposure time MSB

7.2.5.27 REG_FCLK_I2C_ADDRESS

```
#define REG_FCLK_I2C_ADDRESS 0x8040
```

[Slave Address(7), RD/WRn(1), Reg Adress(8)] Slave adddress Hx58 -> HxB when shifted up by 1

7.2.5.28 REG_FCLK_I2C_DATA_RD

```
#define REG_FCLK_I2C_DATA_RD 0x8042
```

[Read Failed (1), Write Failed(1), Toggle bit(1), 0(5), Read Data (8)]

7.2.5.29 REG_FCLK_I2C_DATA_WR

```
#define REG_FCLK_I2C_DATA_WR 0x8041
```

[Clock Select(2), Clock Enable (1), 0(5), Write Data (8)] Clock Select: (00): 250 MHz (01): 200 MHz (10): FPGA FCRIC Clk (11): Si570 Programmable

7.2.5.30 REG_FCLK_SET5

```
#define REG_FCLK_SET5 0xB00C
```

FRM Status

7.2.5.31 REG_FPGA_VERSION

```
#define REG_FPGA_VERSION 0x008F
```

Sandbox Registers

7.2.5.32 REG_FRM_10GbE_SEL

```
#define REG_FRM_10GbE_SEL 0x0037;
```

10GbE Link Select Clock Enables

7.2.5.33 REG_FRM_FPGA_VERSION

```
#define REG_FRM_FPGA_VERSION 0x808F
```

Sandbox Registers

7.2.5.34 REG_FRM_RESET

```
#define REG_FRM_RESET 0x0036
```

Frame Reset

7.2.5.35 REG_FRM_SANDBOX_REG0F

```
#define REG_FRM_SANDBOX_REG0F 0x80FF
```

Image Processing Registers

7.2.5.36 REG_FRM_STREAM_TYPE

```
#define REG_FRM_STREAM_TYPE 0x8003
```

List of Commands

7.2.5.37 REG_IMON_ADC0_CHF

```
#define REG_IMON_ADC0_CHF 0x005B /* V5P0_FP Current Monitor (1/2) */
```

Status Registers

7.2.5.38 REG_MAC_CFG_VECTOR1

```
#define REG_MAC_CFG_VECTOR1 0x0027
```

Ethernet Hardware Conf

7.2.5.39 REG_MAC_CFG_VECTOR2

```
#define REG_MAC_CFG_VECTOR2 0x002D
```

Ethernet Hardware Conf Power Supply Control

7.2.5.40 REG_MAC_STATS2_FAB2B1

```
#define REG_MAC_STATS2_FAB2B1 0x802E
```

SRAM Test Interface

7.2.5.41 REG_PHY1_MDIO_CMD

```
#define REG_PHY1_MDIO_CMD 0x0022
```

Start(1), RnW(1), WDRd(1), PHY Addr(5), REG Addr(5)

7.2.5.42 REG_PS_ENABLE

```
#define REG_PS_ENABLE 0x0030
```

Power Supply Enable:

7.2.5.43 REG_PS_SYNC_DIV0

```
#define REG_PS_SYNC_DIV0 0x0031
```

2.5V Gen

7.2.5.44 REG_PS_SYNC_DIV1

```
#define REG_PS_SYNC_DIV1 0x0032
```

3.3V Gen

7.2.5.45 REG_PS_SYNC_DIV2

```
#define REG_PS_SYNC_DIV2 0x0033
```

2.5V Frame

7.2.5.46 REG_PS_SYNC_DIV3

```
#define REG_PS_SYNC_DIV3 0x0034
```

0.9V Frame

7.2.5.47 REG_PS_SYNC_DIV4

```
#define REG_PS_SYNC_DIV4 0x0035
```

5.0V FP Frame FPGA Control

7.2.5.48 REG_SANDBOX_REG0F

```
#define REG_SANDBOX_REG0F 0x00FF
```

----------< Frame FPGA Registers > Command Registers

7.2.5.49 REG_SI570_REG3

```
#define REG_SI570_REG3 0x003C
```

Power Monitor Registers

7.2.5.50 REG_SRAM_COMMAND

```
#define REG_SRAM_COMMAND 0x8030
```

1 bit [0] >> Read NOT Write 2 bits [3:2] >> Modes: – Single RW 0x00 – Burst RW 0x01 – Test/Diagnostic 10 – Sleep 11 1 bit [4] >> start/stop

7.2.5.51 REG_SRAM_STATUS0

```
#define REG_SRAM_STATUS0 0x803B
```

Programmable Clock

7.2.5.52 REG_STREAM_TYPE

```
#define REG_STREAM_TYPE 0x0003
```

FCLK Values

7.2.5.53 REG_TRIGGERMASK_REG

```
#define REG_TRIGGERMASK_REG 0x8051
```

[00]==SW Trigger, [01]==FP TrigIn2, [10]==FP TrigIn1, [11]==FP TrigIn1OR2

7.2.5.54 REG_TRIGGERREPETITIONTIMELSB_REG

```
#define REG_TRIGGERREPETITIONTIMELSB_REG 0x8209
```

Trigger Cycle Time LSB

7.2.5.55 REG_TRIGGERREPETITIONTIMEMSB_REG

```
#define REG_TRIGGERREPETITIONTIMEMSB_REG 0x8208
```

Trigger Cycle Time MSB

Index