SOFI server Reference Manual

1

Generated by Doxygen 1.5.3

Mon Mar 29 15:52:23 2010

Contents

Chapter 1

SOFI server Main Page

1.1 Project description

This project contains code packages for RCEs that are configured for use in the Sea Of Flash Implementation of the Petacache server. All the code for this project lives in namespace "sofi".

1.2 Project history

1.2.1 2009/12/19 V00_00_00

Used

```
cvs import petacache PETACACHE V00_00_00
```

to create the project and its package "fmc".

1.2.2 2010/02/10

Used cvs2svn to bring the code into Subversion.

1.2.3 2010/02/16

Renamed the project to "sofi". Moved logging code to project quarks.

Chapter 2

SOFI server Namespace Index

2.1 SOFI server Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

SOFI server Hierarchical Index

3.1 SOFI server Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

SOFI server Class Index

4.1 SOFI server Class List

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

Chapter 5

SOFI server File Index

5.1 SOFI server File List

Here is a list of all files with brief descriptions:

Chapter 6

SOFI server Page Index

6.1 SOFI server Related Pages

Here is a list of all related documentation pages:

Chapter 7

SOFI server Namespace Documentation

7.1 anonymous_namespace{findbadblocks.cc} Namespace Reference

Variables

- unsigned LANES_PER_RCE = Params::LanesPerRce
- unsigned FMCS_PER_LANE = Params::FmcsPerLane
- unsigned CHIPS_PER_DEVICE = Params::ChipsPerDevice
- unsigned BLOCKS_PER_CHIP = Params::BlocksPerChip

7.1.1 Variable Documentation

7.1.1.1 unsigned anonymous_namespace{findbadblocks.cc}::BLOCKS_PER_CHIP = Params::BlocksPerChip

Definition at line 136 of file findbadblocks.cc.

Referenced by findbad().

7.1.1.2 unsigned anonymous_namespace{findbadblocks.cc}::CHIPS_PER_DEVICE = Params::ChipsPerDevice

Definition at line 135 of file findbadblocks.cc.

Referenced by findbad().

7.1.1.3 unsigned anonymous_namespace{findbadblocks.cc}::FMCS_PER_LANE = Params::FmcsPerLane

Definition at line 134 of file findbadblocks.cc.

7.1.1.4 unsigned anonymous_namespace{findbadblocks.cc}::LANES_PER_RCE = Params::LanesPerRce

Definition at line 133 of file findbadblocks.cc.

Referenced by createLaneManagers().

7.2 anonymous_namespace{OpDescr.cc} Namespace Reference

Functions

- void dumpBytes (const unsigned char *data, unsigned nbytes)

7.2.1 Function Documentation

7.2.1.1 void anonymous_namespace{OpDescr.cc}::dumpBytes (const unsigned char * *data*, unsigned *nbytes*)

Definition at line 50 of file OpDescr.cc.

Referenced by sofi::fmc::OpDescr::dumpExported(), and sofi::fmc::OpDescr::dumpImported().

7.3 sofi Namespace Reference

7.3.1 Detailed Description

The top-level namespace for the Sea Of Flash source code.

This namespace contains all the C++ entities defined for project "sofi", which is the RCE-with-FMC implementation of the Petacache server.

Namespaces

- namespace fmc

  The namespace for FMC code.

7.4 sofi::fmc Namespace Reference

7.4.1 Detailed Description

The namespace for FMC code.

This namespace contains all the C++ entities used to communicate with an RCE's Flash Memory Controllers directly via PGP.

Classes

- class Damage

  The structure of the "damage" word at the end of FMC-type import frames.

- class FlashAddr

  Used to build the Address field used in the second word of FMC-type PGP frames.

- class LaneManager

  Construct and transmit (export) FMC commands, receive (import) replies. Non-copyable.

- class OpBarrier

  Waits for completion to be posted for a set of OpDescr instances. Non-copyable.

- class OpDescr

  For one FMC operation hold the transaction descriptors and buffers for both the import and export transactions as well as the transaction ID. Non-copyable.

- struct Params

  Publicly accessible constants mostly describing the structure of flash memory.

- class PgpDriverList

  Manage a set of active PGP drivers. A Singleton class.

- struct PgpHeader

  The first word of all PGP frames, both import and export.

- struct FmcHeader

  The first three words of all frames for Flash Core operations and Loopback, import and export.

- struct GetFlashAttributesHeader

  Header for the GetFlashAttributes command.

- struct GetFmcAttributesHeader

  Header for the GetFmcAttributes command.

- struct GetFmcCounterHeader

  Header for the GetCounter command.

- struct EraseBlockHeader

  Header for the Erase Block command.

- struct GetPageletsHeader

  Header for the Get Blocks (Get Pagelets) command.

- struct SetPageHeader

  Header for the Set Page command.

- struct LoopbackHeader

  Header for the Loopback command.

- struct LaneRegisterHeader

  The PGP frame used for register access (import and export).

- class PgpConfiguration

  Abstract base class for a description of the way PGP is set up.

- class PetacachePgpConfig

  Describes the PGP configuration used for Petacache RCE boards.

Namespaces

- namespace Cmd

  Contains constants used to create FMC comand codes.

- namespace Ctr

  Contains constants named for FMC register numbers.

- namespace Reg

  FMC integration-level register names, opcodes, bits, etc.

Functions

- void init_pgp (bool verbose=false)

  Initialize all PGP drivers for the Petacache. Installs the single instance of PgpDriverList.

- size_t maxPgpHeaderSize ()

  The maximum number of bytes in any of the PGP headers defined in this file.

- unsigned numExportBuffers ()

  Return the number of export frame buffers used.

- unsigned numImportBuffers ()

  Return the number of export frame buffers used.

- unsigned maxExportPayloadSize ()

  Return the maximum payload size to be used for any PGP frame used for export.

- unsigned maxImportPayloadSize ()

  Return the maximum payload size to be used for any PGP frame used for import.

- const RcePic::Params & importBufferInfo ()

  Return a description of the frame buffer allocation for imports.

- const RcePic::Params & exportBufferInfo ()

  Return a description of the frame buffer allocation for exports.

- port_number_t operator++ (port_number_t &p)

Variables

- PgpDriverList * _pgp_driver_list = 0

7.4.2 Function Documentation

7.4.2.1 const RcePic::Params& sofi::fmc::exportBufferInfo () [inline]

Return a description of the frame buffer allocation for exports.

Definition at line 343 of file pgp_frames.hh.

References maxExportPayloadSize(), maxPgpHeaderSize(), and numExportBuffers().

Referenced by createLaneManagers().

7.4.2.2 const RcePic::Params& sofi::fmc::importBufferInfo () [inline]

Return a description of the frame buffer allocation for imports.

Definition at line 331 of file pgp_frames.hh.

References maxImportPayloadSize(), maxPgpHeaderSize(), and numImportBuffers().

Referenced by createLaneManagers(), and sofi::fmc::PetacachePgpConfig::picParams().

7.4.2.3 void sofi::fmc::init_pgp (bool *verbose* = false)

Initialize all PGP drivers for the Petacache. Installs the single instance of PgpDriverList.

We need to replace namespace RcePgp's function init_pgp() and its class Driverlist because they make wrong assumptions about how many PGP ports there are and which PIC blocks are assigned to each port.

Definition at line 277 of file PgpDriverList.cc.

Referenced by findbad().

7.4.2.4 unsigned sofi::fmc::maxExportPayloadSize () [inline]

Return the maximum payload size to be used for any PGP frame used for export.

The largest payload will be the contents of an entire FMC page with "extra" data if Reed-Solomon encoding is disabled.

Definition at line 316 of file pgp_frames.hh.

References sofi::fmc::Params::FmcPageSizeWithExtra.

Referenced by exportBufferInfo().

7.4.2.5 unsigned sofi::fmc::maxImportPayloadSize () [inline]

Return the maximum payload size to be used for any PGP frame used for import.

The largest payload will be the contents of an entire FMC page with "extra" data if Reed-Solomon encoding is disabled, plus four bytes for the "damage" word. This number is the maximum payload size allowed by the protocol plugin; exceeding it will cause PIC block errors and may cause exceptions in the PGP driver tasks.

Definition at line 327 of file pgp_frames.hh.

References sofi::fmc::Params::FmcPageSizeWithExtra.

Referenced by importBufferInfo().

7.4.2.6 size_t sofi::fmc::maxPgpHeaderSize () [inline]

The maximum number of bytes in any of the PGP headers defined in this file.

All frame headers used with a given PGP port must have the same size so there's no need to have separate definitions for import and export.

Definition at line 280 of file pgp_frames.hh.

Referenced by exportBufferInfo(), and importBufferInfo().

7.4.2.7 unsigned sofi::fmc::numExportBuffers () [inline]

Return the number of export frame buffers used.

Definition at line 303 of file pgp_frames.hh.

Referenced by exportBufferInfo(), and numImportBuffers().

7.4.2.8 unsigned sofi::fmc::numImportBuffers () [inline]

Return the number of export frame buffers used.

Since one command produces at most one reply buffer we allocate the same number of import buffers as export buffers.

Definition at line 309 of file pgp_frames.hh.

References numExportBuffers().

Referenced by importBufferInfo().

7.4.2.9 port_number_t sofi::fmc::operator++ (port_number_t & *p*) [inline]

Definition at line 148 of file PgpDriverList.cc.

7.4.3 Variable Documentation

7.4.3.1 PgpDriverList* sofi::fmc::_pgp_driver_list = 0

Definition at line 146 of file PgpDriverList.cc.

Referenced by sofi::fmc::PgpDriverList::create(), and sofi::fmc::PgpDriverList::instance().

7.5 sofi::fmc::Cmd Namespace Reference

7.5.1 Detailed Description

Contains constants used to create FMC comand codes.

Enumerations

- enum Code {
  GetFlashAttributes = 0, GetFmcAttributes = 1, GetCounter = 2, EraseBlock = 3,
  GetBlocks = 4, SetPage = 5, Loopback = 7 }
  The legal FMC-frame command code values.

7.5.2 Enumeration Type Documentation

7.5.2.1 enum sofi::fmc::Cmd::Code

The legal FMC-frame command code values.

Enumerator:

GetFlashAttributes

GetFmcAttributes

GetCounter

EraseBlock

GetBlocks

SetPage

Loopback Implemented by lane firmware, not part of core set.

Definition at line 48 of file fmc_const.hh.

7.6 sofi::fmc::Ctr Namespace Reference

7.6.1 Detailed Description

Contains constants named for FMC register numbers.

Enumerations

- enum Counter {
  Reads, Writes, Erasures, DeviceErrors,
  ArbTime, BusyTime, ArbTimeouts, CommandCongestion }
  The set of FMC counters.

7.6.2 Enumeration Type Documentation

7.6.2.1 enum sofi::fmc::Ctr::Counter

The set of FMC counters.

All counters have 16 bits of data, except for BusyTime and ArbTimeouts, which have 32 bits of data.

Enumerator:

Reads

Writes

Erasures

DeviceErrors

ArbTime

BusyTime

ArbTimeouts

CommandCongestion

Definition at line 67 of file fmc_const.hh.

7.7 sofi::fmc::Reg Namespace Reference

7.7.1 Detailed Description

FMC integration-level register names, opcodes, bits, etc.

Enumerations

- enum Addr {
  Version = 0, Status = 1, IdValid = 2, Fmc01IdLow = 4,
  Fmc01IdHigh = 5, Fmc23IdLow = 6, Fmc23IdHigh = 7, Fmc0Csr = 16,
  Fmc1Csr = 17, Fmc2Csr = 18, Fmc3Csr = 19 }
  The set of known register addresses for each lane.
- enum Opcode { Read, Write, Set, Clear }
  Lane register opcodes.
- enum CsrBits { Reset = 0x1, Enable = 0x8, PageSize = 0x01000000 }
  Bits in each FMC CSR.
- enum StatusBits { RotateDisplay = 0x40000000, EnableFlashWrite = 0x80000000 }
  Bits in the lane status register.

Functions

- Addr operator++ (Addr &a)

7.7.2 Enumeration Type Documentation

7.7.2.1 enum sofi::fmc::Reg::Addr

The set of known register addresses for each lane.

All registers are 32 bits wide. It's guaranteed that FmcnCsr = Fmc0Csr + n, 0 <= n < Params::FmcsPerLane

Enumerator:

Version

Status

IdValid

Fmc01IdLow Low order part of daughter card 0 ID.

Fmc01IdHigh High order part of daughter card 0 ID.

Fmc23IdLow Low order part of daughter card 1 ID.

Fmc23IdHigh High order part of daughter card 1 ID.

Fmc0Csr

Fmc1Csr

Fmc2Csr

Fmc3Csr

Definition at line 49 of file lane_const.hh.

7.7.2.2 enum sofi::fmc::Reg::CsrBits

Bits in each FMC CSR.

Enumerator:

Reset

Enable

PageSize Flash devices have big pages (4K + 128 bytes).

Definition at line 76 of file lane_const.hh.

7.7.2.3 enum sofi::fmc::Reg::Opcode

Lane register opcodes.

Enumerator:

Read Get the contents of the entire register.

Write Alter the contents of the entire register.

Set Set those bits that are set in the contents field.

Clear Clear those bits that are set in the contents field.

Definition at line 67 of file lane_const.hh.

7.7.2.4 enum sofi::fmc::Reg::StatusBits

Bits in the lane status register.

Enumerator:

RotateDisplay

EnableFlashWrite

Definition at line 84 of file lane_const.hh.

7.7.3 Function Documentation

7.7.3.1 Addr sofi::fmc::Reg::operator++ (Addr & *a*) [inline]

Definition at line 63 of file lane_const.hh.

Chapter 8

SOFI server Class Documentation

8.1 sofi::fmc::Damage Class Reference

`#include <Damage.hh>`

8.1.1 Detailed Description

The structure of the "damage" word at the end of FMC-type import frames.

Definition at line 45 of file Damage.hh.

Public Member Functions

- Damage (unsigned value)
- bool writeDataError () const
- bool writeProtectionError () const
- bool flashChipError () const
- bool flashTimeout () const
- bool rsDecodeFailure () const
- bool rsDecodeError () const
- unsigned rsErrorCount () const
- bool error () const

Private Attributes

- union {
  unsigned word
  struct {
  unsigned dnc1:7
  unsigned writeDataError:1
  unsigned writeProtectionError:1
  unsigned flashChipError:1
  unsigned flashTimeout:1
  unsigned rsDecodeFailure:1
  unsigned rsDecodeError:1

```
        unsigned rsErrorCount:3
        unsigned dnc2:16
      } bits
    } _value
```

8.1.2 Constructor & Destructor Documentation

8.1.2.1 sofi::fmc::Damage::Damage (unsigned *value*) [inline]

Definition at line 64 of file Damage.hh.

References _value.

8.1.3 Member Function Documentation

8.1.3.1 bool sofi::fmc::Damage::writeDataError () const [inline]

Definition at line 69 of file Damage.hh.

References _value.

8.1.3.2 bool sofi::fmc::Damage::writeProtectionError () const [inline]

Definition at line 71 of file Damage.hh.

References _value.

8.1.3.3 bool sofi::fmc::Damage::flashChipError () const [inline]

Definition at line 73 of file Damage.hh.

References _value.

8.1.3.4 bool sofi::fmc::Damage::flashTimeout () const [inline]

Definition at line 75 of file Damage.hh.

References _value.

8.1.3.5 bool sofi::fmc::Damage::rsDecodeFailure () const [inline]

Definition at line 77 of file Damage.hh.

References _value.

8.1.3.6 bool sofi::fmc::Damage::rsDecodeError () const [inline]

Definition at line 79 of file Damage.hh.

References _value.

8.1.3.7 unsigned sofi::fmc::Damage::rsErrorCount () const [inline]

Definition at line 81 of file Damage.hh.

References _value.

8.1.3.8 bool sofi::fmc::Damage::error () const [inline]

Definition at line 83 of file Damage.hh.

References _value.

8.1.4 Member Data Documentation

8.1.4.1 unsigned sofi::fmc::Damage::word

Definition at line 48 of file Damage.hh.

8.1.4.2 unsigned sofi::fmc::Damage::dnc1

Definition at line 50 of file Damage.hh.

8.1.4.3 unsigned sofi::fmc::Damage::writeDataError

Definition at line 51 of file Damage.hh.

8.1.4.4 unsigned sofi::fmc::Damage::writeProtectionError

Definition at line 52 of file Damage.hh.

8.1.4.5 unsigned sofi::fmc::Damage::flashChipError

Definition at line 53 of file Damage.hh.

8.1.4.6 unsigned sofi::fmc::Damage::flashTimeout

Definition at line 54 of file Damage.hh.

8.1.4.7 unsigned sofi::fmc::Damage::rsDecodeFailure

Definition at line 55 of file Damage.hh.

8.1.4.8 unsigned sofi::fmc::Damage::rsDecodeError

Definition at line 56 of file Damage.hh.

8.1.4.9 unsigned sofi::fmc::Damage::rsErrorCount

Definition at line 57 of file Damage.hh.

8.1.4.10 unsigned sofi::fmc::Damage::dnc2

Definition at line 58 of file Damage.hh.

8.1.4.11 struct { ... } sofi::fmc::Damage::bits

8.1.4.12 union { ... } sofi::fmc::Damage::_value [private]

Referenced by Damage(), error(), flashChipError(), flashTimeout(), rsDecodeError(), rsDecodeFailure(), rsErrorCount(), writeDataError(), and writeProtectionError().

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

- Damage.hh

8.2 sofi::fmc::EraseBlockHeader Struct Reference

`#include <pgp_frames.hh>`

8.2.1 Detailed Description

Header for the Erase Block command.

Definition at line 173 of file pgp_frames.hh.

Public Member Functions

- EraseBlockHeader (unsigned tid, unsigned lane, unsigned fmc, const FlashAddr &addr)
  Parameters:
  ← ***addr*** *All fields except page are used.*

Public Attributes

- FmcHeader fmch

8.2.2 Constructor & Destructor Documentation

8.2.2.1 sofi::fmc::EraseBlockHeader::EraseBlockHeader (unsigned *tid*, unsigned *lane*, unsigned *fmc*, const FlashAddr & *addr*) `[inline]`

Parameters:

← ***addr*** All fields except page are used.

Definition at line 176 of file pgp_frames.hh.

8.2.3 Member Data Documentation

8.2.3.1 FmcHeader sofi::fmc::EraseBlockHeader::fmch

Definition at line 174 of file pgp_frames.hh.

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

- pgp_frames.hh

8.3 sofi::fmc::FlashAddr Class Reference

`#include <FlashAddr.hh>`

8.3.1 Detailed Description

Used to build the Address field used in the second word of FMC-type PGP frames.

Definition at line 48 of file FlashAddr.hh.

Public Member Functions

- FlashAddr (unsigned page=0, unsigned block=0, unsigned chip=0)
- unsigned page () const
- unsigned block () const
- unsigned chip () const
- unsigned fcsAddr () const

  Return all the address fields as a single word a.k.a. a Flash Core Set address.

Private Attributes

- union {
  unsigned word
  struct {
  unsigned dnc:11
  unsigned chip:2
  unsigned block:13
  unsigned page:6
  } bits
  } _addr

8.3.2 Constructor & Destructor Documentation

8.3.2.1 sofi::fmc::FlashAddr::FlashAddr (unsigned *page* = 0, unsigned *block* = 0, unsigned *chip* = 0) [inline, explicit]

Definition at line 61 of file FlashAddr.hh.

References _addr, block(), chip(), and page().

8.3.3 Member Function Documentation

8.3.3.1 unsigned sofi::fmc::FlashAddr::page () const [inline]

Definition at line 68 of file FlashAddr.hh.

References _addr.

Referenced by FlashAddr().

8.3.3.2 unsigned sofi::fmc::FlashAddr::block () const [inline]

Definition at line 70 of file FlashAddr.hh.

References _addr.

Referenced by FlashAddr().

8.3.3.3 unsigned sofi::fmc::FlashAddr::chip () const [inline]

Definition at line 72 of file FlashAddr.hh.

References _addr.

Referenced by FlashAddr().

8.3.3.4 unsigned sofi::fmc::FlashAddr::fcsAddr () const [inline]

Return all the address fields as a single word a.k.a. a Flash Core Set address.

Definition at line 77 of file FlashAddr.hh.

References _addr.

8.3.4 Member Data Documentation

8.3.4.1 unsigned sofi::fmc::FlashAddr::word

Definition at line 51 of file FlashAddr.hh.

8.3.4.2 unsigned sofi::fmc::FlashAddr::dnc

Definition at line 53 of file FlashAddr.hh.

8.3.4.3 unsigned sofi::fmc::FlashAddr::chip

Definition at line 54 of file FlashAddr.hh.

8.3.4.4 unsigned sofi::fmc::FlashAddr::block

Definition at line 55 of file FlashAddr.hh.

8.3.4.5 unsigned sofi::fmc::FlashAddr::page

Definition at line 56 of file FlashAddr.hh.

8.3.4.6 struct { ... } sofi::fmc::FlashAddr::bits

8.3.4.7 union { ... } sofi::fmc::FlashAddr::_addr [private]

Referenced by block(), chip(), fcsAddr(), FlashAddr(), and page().

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

- FlashAddr.hh

8.4 sofi::fmc::FmcHeader Struct Reference

`#include <pgp_frames.hh>`

8.4.1 Detailed Description

The first three words of all frames for Flash Core operations and Loopback, import and export.

The FMC number within the lane is the same as the PGP VC number, so the VC number appears in two places. The destination filed must have zero in the lowest order bit.

Definition at line 85 of file pgp_frames.hh.

Public Member Functions

- FmcHeader (unsigned tid, Cmd::Code command, unsigned lane, unsigned fmc, const FlashAddr &addr=FlashAddr(), bool reed_solomon=false, unsigned length=0, unsigned offset=0)

Public Attributes

- PgpHeader pgph
- unsigned command:3

  Command code.

- unsigned _dnc1:2
- unsigned address:23

  Flash address (Flash Core Set).

- unsigned lane:2

  Lane number.

- unsigned fmc:2

  FMC number (same value as the VC field in PgpHeader).

- unsigned we:1

  Enable Reed-Solomon calculation on write.

- unsigned ws:1

  Client pagelet on write has R-S bytes.

- unsigned re:1

  Enable Reed-Solomon checking on read.

- unsigned rs:1

  Return R-S bytes in each pagelet to client.

- unsigned _dnc2:14
- unsigned length:6
- unsigned _dnc3:1
- unsigned od:1

Disable use of offset field.

- unsigned offset:6
- unsigned data

  The 32-bit data transferred (a don't-care for exports).

8.4.2 Constructor & Destructor Documentation

8.4.2.1 sofi::fmc::FmcHeader::FmcHeader (unsigned *tid*, Cmd::Code *command*, unsigned *lane*, unsigned *fmc*, const FlashAddr & *addr* = FlashAddr(), bool *reed_solomon* = false, unsigned *length* = 0, unsigned *offset* = 0) [inline]

Parameters:

← ***tid*** The transaction ID.

← ***command*** The command code.

← ***lane*** The lane number.

← ***fmc*** The fmc number (within the lane).

← ***reed_solomon*** Will the firmware do Reed-Solomon error checking? Set the WE, WS, RE and RS bits appropriately.

← ***length*** The length field value.

← ***offset*** Zero sets the OD bit; otherwise OD is zero and Offset become this value less one.

Definition at line 116 of file pgp_frames.hh.

8.4.3 Member Data Documentation

8.4.3.1 PgpHeader sofi::fmc::FmcHeader::pgph

Definition at line 86 of file pgp_frames.hh.

8.4.3.2 unsigned sofi::fmc::FmcHeader::command

Command code.

Definition at line 87 of file pgp_frames.hh.

8.4.3.3 unsigned sofi::fmc::FmcHeader::_dnc1

Definition at line 88 of file pgp_frames.hh.

8.4.3.4 unsigned sofi::fmc::FmcHeader::address

Flash address (Flash Core Set).

Definition at line 89 of file pgp_frames.hh.

8.4.3.5 unsigned sofi::fmc::FmcHeader::lane

Lane number.

Definition at line 90 of file pgp_frames.hh.

8.4.3.6 unsigned sofi::fmc::FmcHeader::fmc

FMC number (same value as the VC field in PgpHeader).

Definition at line 91 of file pgp_frames.hh.

8.4.3.7 unsigned sofi::fmc::FmcHeader::we

Enable Reed-Solomon calculation on write.

Definition at line 92 of file pgp_frames.hh.

8.4.3.8 unsigned sofi::fmc::FmcHeader::ws

Client pagelet on write has R-S bytes.

Definition at line 93 of file pgp_frames.hh.

8.4.3.9 unsigned sofi::fmc::FmcHeader::re

Enable Reed-Solomon checking on read.

Definition at line 94 of file pgp_frames.hh.

8.4.3.10 unsigned sofi::fmc::FmcHeader::rs

Return R-S bytes in each pagelet to client.

Definition at line 95 of file pgp_frames.hh.

8.4.3.11 unsigned sofi::fmc::FmcHeader::_dnc2

Definition at line 96 of file pgp_frames.hh.

8.4.3.12 unsigned sofi::fmc::FmcHeader::length

One less than the number of pagelets to transfer (or that were transferred) from the FMC core, starting at the beginning of the page.

Definition at line 97 of file pgp_frames.hh.

8.4.3.13 unsigned sofi::fmc::FmcHeader::_dnc3

Definition at line 100 of file pgp_frames.hh.

8.4.3.14 unsigned sofi::fmc::FmcHeader::od

Disable use of offset field.

Definition at line 101 of file pgp_frames.hh.

8.4.3.15 unsigned sofi::fmc::FmcHeader::offset

if od is not set, one less than the number of pagelets to drop (or that were dropped) from the series transferred from the FMC core.

Definition at line 102 of file pgp_frames.hh.

8.4.3.16 unsigned sofi::fmc::FmcHeader::data

The 32-bit data transferred (a don't-care for exports).

Definition at line 105 of file pgp_frames.hh.

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

- pgp_frames.hh

8.5 sofi::fmc::GetFlashAttributesHeader Struct Reference

`#include <pgp_frames.hh>`

8.5.1 Detailed Description

Header for the GetFlashAttributes command.

Definition at line 145 of file pgp_frames.hh.

Public Member Functions

- GetFlashAttributesHeader (unsigned tid, unsigned lane, unsigned fmc)

Public Attributes

- FmcHeader fmch

8.5.2 Constructor & Destructor Documentation

8.5.2.1 sofi::fmc::GetFlashAttributesHeader::GetFlashAttributesHeader (unsigned *tid*, unsigned *lane*, unsigned *fmc*) [inline]

Definition at line 147 of file pgp_frames.hh.

8.5.3 Member Data Documentation

8.5.3.1 FmcHeader sofi::fmc::GetFlashAttributesHeader::fmch

Definition at line 146 of file pgp_frames.hh.

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

- pgp_frames.hh

8.6 sofi::fmc::GetFmcAttributesHeader Struct Reference

`#include <pgp_frames.hh>`

8.6.1 Detailed Description

Header for the GetFmcAttributes command.

Definition at line 154 of file pgp_frames.hh.

Public Member Functions

- GetFmcAttributesHeader (unsigned tid, unsigned lane, unsigned fmc)

Public Attributes

- FmcHeader fmch

8.6.2 Constructor & Destructor Documentation

8.6.2.1 sofi::fmc::GetFmcAttributesHeader::GetFmcAttributesHeader (unsigned *tid*, unsigned *lane*, unsigned *fmc*) [inline]

Definition at line 156 of file pgp_frames.hh.

8.6.3 Member Data Documentation

8.6.3.1 FmcHeader sofi::fmc::GetFmcAttributesHeader::fmch

Definition at line 155 of file pgp_frames.hh.

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

- pgp_frames.hh

8.7 sofi::fmc::GetFmcCounterHeader Struct Reference

`#include <pgp_frames.hh>`

8.7.1 Detailed Description

Header for the GetCounter command.

Definition at line 163 of file pgp_frames.hh.

Public Member Functions

- GetFmcCounterHeader (unsigned tid, unsigned lane, unsigned fmc, Ctr::Counter counter)

Public Attributes

- FmcHeader fmch

8.7.2 Constructor & Destructor Documentation

8.7.2.1 sofi::fmc::GetFmcCounterHeader::GetFmcCounterHeader (unsigned *tid*, unsigned *lane*, unsigned *fmc*, Ctr::Counter *counter*) [inline]

Definition at line 166 of file pgp_frames.hh.

8.7.3 Member Data Documentation

8.7.3.1 FmcHeader sofi::fmc::GetFmcCounterHeader::fmch

Definition at line 164 of file pgp_frames.hh.

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

- pgp_frames.hh

8.8 sofi::fmc::GetPageletsHeader Struct Reference

`#include <pgp_frames.hh>`

8.8.1 Detailed Description

Header for the Get Blocks (Get Pagelets) command.

Definition at line 183 of file pgp_frames.hh.

Public Member Functions

- GetPageletsHeader (unsigned tid, unsigned lane, unsigned fmc, const FlashAddr &addr, bool reed_solomon, unsigned firstPagelet, unsigned lastPagelet)

Public Attributes

- FmcHeader fmch

8.8.2 Constructor & Destructor Documentation

8.8.2.1 sofi::fmc::GetPageletsHeader::GetPageletsHeader (unsigned *tid*, unsigned *lane*, unsigned *fmc*, const FlashAddr & *addr*, bool *reed_solomon*, unsigned *firstPagelet*, unsigned *lastPagelet*) [inline]

Parameters:

← ***addr*** All fields are used.

← ***reed_solomon*** If true then the "extra" bytes in each pagelet are used as check bytes and are not returned to the client. If false then each pagelet has the extra bytes added to its end.

← ***firstPagelet*** The number of the first pagelet to retrieve (counting from zero).

← ***lastPagelet*** The number of the last pagelet to retrieve (counting from zero).

Definition at line 192 of file pgp_frames.hh.

8.8.3 Member Data Documentation

8.8.3.1 FmcHeader sofi::fmc::GetPageletsHeader::fmch

Definition at line 184 of file pgp_frames.hh.

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

- pgp_frames.hh

8.9 sofi::fmc::LaneManager Class Reference

`#include <LaneManager.hh>`

8.9.1 Detailed Description

Construct and transmit (export) FMC commands, receive (import) replies. Non-copyable.

A Petacache RCE is constructed so that the lane number is the same as number of the PGP port used to communicate with the lane's FMCs (some FMC commands require an explicit lane number). Each instance of this class uses one PGP driver hence only one PGP port; therefore each instance deals with only a single lane.

Client code uses the operation member functions of this class to communicate with FMCs. They know how to manage buffers, manage transaction descriptors and assemble FMC commands within buffers. Since this class inherits from RcePgp::Handler it also defines the methods for handling completion events and the reception of FMC responses. All commands are sent with the "do not complete" bit set so the only completion events we'll get will be in case of PGP transmission errors. Command completion events only tell you when commands have been successfully transmitted; code using this class will have to synchronize its operations by waiting for replies using class OpDescr.

An instance of OpDescr is passed to each command member function so that the client can use OpDescr::waitForCompletion() to block until the operation is done. The OpDescr holds the transaction descriptors and buffers for both the export and import sides of an operation. The export and import sides are freed in the OpDescr destructor. In either case the freeing is actually performed by member functions in this class which are called from the OpDescr.

The Flash Core Set operations Get Flash Attributes, Get FMC Attributes, Get Counter, Erase Block, Get Blocks (Pagelets), and Set Page require an explicit lane number in the FMC-type PGP frames they use. The lane number is supplied from the _lane member of this class. The other operations implicitly use the lane atached to the port used by the PGP driver.

The Loopback operation is exceptional in that it uses FMC frames but involves no FMC; the lane and FMC numbers are ignored (and set to zero).

The other FMC operations are carried out using Register Access protocol PGP frames which have a value of one in the DEST field of the PGP header and a VC number of zero. The lane number is implicitly the same as the PGP port number. Some operations affect the entire lane and so don't need any address. Others affect only one FMC (for example each has its own CSR) and take an FMC number argument.

Each FMC operation is assigned a transaction ID which is placed in the field assigned for this purpose in the PGP header of the export frame. The import frame received in response contains a copy of the exported TID. Each instance of this class contains a table which maps TID to the OpDescr of the pending operation. The TID is basically a simple counter incremented before each operation but shifted left two bits so that the lane/port number can be OR-ed in.

Definition at line 124 of file LaneManager.hh.

Public Member Functions

- LaneManager (RcePgp::Driver &pgpd, const RcePic::Params &importParams, const RcePic::Params &exportParams)
- void initializeLane ()

  Prepare a lane and its FMCs for the real work to come. Must be called only after PGP driver handlers have been set and before any flash operations are performed.

- void read (unsigned fmc, const FlashAddr &addr, unsigned firstPagelet, unsigned lastPagelet, OpDescr &od)

  Read data using a Reed-Solomon code for error detection and correction.

- void readNoRs (unsigned fmc, const FlashAddr &addr, unsigned firstPagelet, unsigned lastPagelet, OpDescr &od)

  Read data without any Reed-Solomon checking.

- void write (unsigned fmc, const FlashAddr &addr, const void *pageData, OpDescr &od)

  Write data with the lane firmware adding Reed-Solomon check bytes.

- void erase (unsigned fmc, const FlashAddr &addr, OpDescr &od)

  Erase a block of flash memory.

- void getFmcAttr (unsigned fmc, OpDescr &od)

  Get the attribute word for a given FMC.

- void getFlashAttr (unsigned fmc, OpDescr &od)

  Get the attribute data for all the devices connected to a given FMC.

- void getFmcCounter (unsigned fmc, Ctr::Counter counter, OpDescr &od)

  Get the value of one of the FMC performance counters.

- void loopback (void *payloadData, unsigned payloadSize, unsigned firstPagelet, unsigned lastPagelet, OpDescr &od)

  Send some data to the lane which sends it right back as if it were the result of an FMC flash read with RS; RS code bytes are actually generated and checked (but not sent to the client).

- void disableLaneWrites (OpDescr &od)

  Disable writes to flash for the lane.

- void enableLaneWrites (OpDescr &od)

  Enable writes to flash for the lane.

- void resetFmc (unsigned fmc, OpDescr &od)

  Reset an FMC.

- void enableFmc (unsigned fmc, OpDescr &od)

  Enable operation of an FMC.

- void readLaneRegister (Reg::Addr addr, OpDescr &od)

  Read one of the lane registers.

- void writeLaneRegister (Reg::Addr addr, unsigned data, OpDescr &od)

  Write one of the lane registers.

- void forward (const void *pgpFrame, unsigned frameSize, OpDescr &od)

  Forward to our PGP port a frame constructed elsewhere.

- RcePgp::Driver & pgpd () const

Protected Member Functions

- virtual void completed (RcePic::Tds *tds, RcePgp::Driver *)

  Handle an export completion event.

- virtual void received (RcePic::Tds *tds, RcePgp::Driver *)

  Handle an import event.

Private Member Functions

- void _setFmcPageSize (unsigned fmc, OpDescr &od)

  Set the FMC flash-page-size bit to the correct value.

- void _allocExportSide (OpDescr &od)

  Reserve the descriptors and buffers needed for an export and give them to the OpDescr.

- void _export (OpDescr &od, unsigned payloadsize=0)

  Perform an export based on the information in the OpDescr.

- void _freeExportSide (OpDescr &od)

  Release the export-related resources held by the OpDescr.

- void _freeImportSide (OpDescr &od)

  Release the import-related resources help by the OpDescr.

- LaneManager (const LaneManager &)
- LaneManager & operator= (const LaneManager &)

Private Attributes

- unsigned _tidCounter

  Transaction counter used with lane no. to make TIDs.

- RceSvc::Semaphore _exportWait

  Given when _exportPool goes fron empty to nonempty.

- RcePic::Pool _exportPool

  Pool of resources used for export.

- std::map< unsigned, OpDescr * > _pendingOds

  Mapping TID -> pending OpDescrs.

- RcePgp::Driver & _pgp_driver

  The PGP driver used for our lane.

- const RcePic::Params & _importParams

  Describes the set of import buffers.

- const RcePic::Params & _exportParams

  Describes the set of export buffers.

- const unsigned _lane

  The lane number.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 sofi::fmc::LaneManager::LaneManager (RcePgp::Driver & *pgpd*, const RcePic::Params & *importParams*, const RcePic::Params & *exportParams*)

Parameters:

↔ ***pgpd*** A reference to the driver for the PGP port connected to the lane's FMCs. Must be properly initialized with a set of buffers and transaction descriptors for import. The lane number for this instance will be the PGP port number of this driver. used to track a preallocated set of buffers and transaction descriptors for export.

← ***importParams*** Describes the set of buffers allocated for import.

← ***exportParams*** Describes the set of buffers allocated for export.

Definition at line 64 of file LaneManager.cc.

8.9.2.2 sofi::fmc::LaneManager::LaneManager (const LaneManager &) [private]

8.9.3 Member Function Documentation

8.9.3.1 void sofi::fmc::LaneManager::initializeLane ()

Prepare a lane and its FMCs for the real work to come. Must be called only after PGP driver handlers have been set and before any flash operations are performed.

Definition at line 78 of file LaneManager.cc.

References _pgp_driver, _setFmcPageSize(), enableFmc(), enableLaneWrites(), sofi::fmc::Params::FmcsPerLane, resetFmc(), and sofi::fmc::OpDescr::waitForCompletion().

8.9.3.2 void sofi::fmc::LaneManager::read (unsigned *fmc*, const FlashAddr & *addr*, unsigned *firstPagelet*, unsigned *lastPagelet*, OpDescr & *od*)

Read data using a Reed-Solomon code for error detection and correction.

The "extra" parts of the pagelets containing the RS code bytes are not returned to the client. A Get Blocks operation with the RE bit set and the RS bit clear.

Parameters:

← ***fmc*** The number of the FMC to use.

← ***addr*** Specifies the page to be read.

← ***firstPagelet*** The first pagelet (counting from 0) to transfer from the page.

← ***lastPagelet*** The last pagelet (counting from 0) to transfer from the page.

↔ ***od*** Operation synchronization and resource control.

Definition at line 110 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.3 void sofi::fmc::LaneManager::readNoRs (unsigned *fmc*, const FlashAddr & *addr*, unsigned *firstPagelet*, unsigned *lastPagelet*, OpDescr & *od*)

Read data without any Reed-Solomon checking.

The "extra" part of each pagelet containing the RS code bytes is returned to the client at the end of the pagelet. A Get Blocks operation with the RE bit clear and the RS bit set.

Parameters:

← ***fmc*** The number of the FMC to use.

← ***addr*** Specifies the page to be read.

← ***firstPagelet*** The first pagelet (counting from 0) to transfer from the page.

← ***lastPagelet*** The last pagelet (counting from 0) to transfer from the page.

↔ ***od*** Operation synchronization and resource control.

Definition at line 127 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.4 void sofi::fmc::LaneManager::write (unsigned *fmc*, const FlashAddr & *addr*, const void * *pageData*, OpDescr & *od*)

Write data with the lane firmware adding Reed-Solomon check bytes.

The "extra" part of each pagelet containing the RS code must not be supplied by the client; the lane firmware manufactures it.

Parameters:

← ***fmc*** The number of the FMC to use.

← ***addr*** Specifies the page to be written. A Set Page operation with the WE bit set and the WS bit clear.

← ***pageData*** Either null or a pointer to at least a page's worth of data (only one page is copied). If null then whatever random stuff happens to be in the export buffer is used.

↔ ***od*** Operation synchronization and resource control.

Definition at line 144 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), sofi::fmc::Params::FmcPageSize, and sofi::fmc::OpDescr::tid().

8.9.3.5 void sofi::fmc::LaneManager::erase (unsigned *fmc*, const FlashAddr & *addr*, OpDescr & *od*)

Erase a block of flash memory.

Parameters:

← ***fmc*** The number of the FMC to use.

← ***addr*** Specifies the block to be erased (the page number is ignored).

↔ ***od*** Operation synchronization and resource control.

Definition at line 151 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.6 void sofi::fmc::LaneManager::getFmcAttr (unsigned *fmc*, OpDescr & *od*)

Get the attribute word for a given FMC.

Definition at line 157 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.7 void sofi::fmc::LaneManager::getFlashAttr (unsigned *fmc*, OpDescr & *od*)

Get the attribute data for all the devices connected to a given FMC.

Parameters:

↔ ***od*** Operation synchronization and resource control.

Definition at line 163 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.8 void sofi::fmc::LaneManager::getFmcCounter (unsigned *fmc*, Ctr::Counter *counter*, OpDescr & *od*)

Get the value of one of the FMC performance counters.

Parameters:

← ***counter*** Specifies the counter to read.

↔ ***od*** Operation synchronization and resource control.

Definition at line 169 of file LaneManager.cc.

References _allocExportSide(), _export(), _lane, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.9 void sofi::fmc::LaneManager::loopback (void * *payloadData*, unsigned *payloadSize*, unsigned *firstPagelet*, unsigned *lastPagelet*, OpDescr & *od*)

Send some data to the lane which sends it right back as if it were the result of an FMC flash read with RS; RS code bytes are actually generated and checked (but not sent to the client).

Parameters:

← ***payloadData*** A pointer to the data to be sent and read back.

← ***payloadSize*** The number of bytes being sent (at most a pageful).

← ***firstPagelet*** The first pagelet (counting from 0) to transfer back.

← ***lastPagelet*** The last pagelet (counting from 0) to transfer back.

↔ ***od*** Operation synchronization and resource control.

Definition at line 175 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.10 void sofi::fmc::LaneManager::disableLaneWrites (OpDescr & *od*)

Disable writes to flash for the lane.

Parameters:

↔ ***od*** Operation synchronization and resource control.

Definition at line 196 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::Reg::Clear, sofi::fmc::Reg::EnableFlashWrite, sofi::fmc::OpDescr::exportTds(), sofi::fmc::Reg::Status, and sofi::fmc::OpDescr::tid().

8.9.3.11 void sofi::fmc::LaneManager::enableLaneWrites (OpDescr & *od*)

Enable writes to flash for the lane.

Parameters:

↔ ***od*** Operation synchronization and resource control.

Definition at line 187 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::Reg::EnableFlashWrite, sofi::fmc::OpDescr::exportTds(), sofi::fmc::Reg::Set, sofi::fmc::Reg::Status, and sofi::fmc::OpDescr::tid().

Referenced by initializeLane().

8.9.3.12 void sofi::fmc::LaneManager::resetFmc (unsigned *fmc*, OpDescr & *od*)

Reset an FMC.

Parameters:

← ***fmc*** The number of the FMC within the lane.

↔ ***od*** Operation synchronization and resource control.

Definition at line 205 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::OpDescr::exportTds(), sofi::fmc::Reg::Fmc0Csr, sofi::fmc::Reg::Reset, sofi::fmc::Reg::Set, and sofi::fmc::OpDescr::tid().

Referenced by initializeLane().

8.9.3.13 void sofi::fmc::LaneManager::_setFmcPageSize (unsigned *fmc*, OpDescr & *od*) [private]

Set the FMC flash-page-size bit to the correct value.

Parameters:

← ***fmc*** The number of the FMC within the lane.

↔ ***od*** Operation synchronization and resource control.

All Petacache RCEs use only the "new" flash with the larger page size.

Definition at line 214 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::OpDescr::exportTds(), sofi::fmc::Reg::Fmc0Csr, sofi::fmc::Reg::PageSize, sofi::fmc::Reg::Set, and sofi::fmc::OpDescr::tid().

Referenced by initializeLane().

8.9.3.14 void sofi::fmc::LaneManager::enableFmc (unsigned *fmc*, OpDescr & *od*)

Enable operation of an FMC.

Parameters:

← ***fmc*** The number of the FMC within the lane.

↔ ***od*** Operation synchronization and resource control.

Definition at line 223 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::Reg::Enable, sofi::fmc::OpDescr::exportTds(), sofi::fmc::Reg::Fmc0Csr, sofi::fmc::Reg::Set, and sofi::fmc::OpDescr::tid().

Referenced by initializeLane().

8.9.3.15 void sofi::fmc::LaneManager::readLaneRegister (Reg::Addr *addr*, OpDescr & *od*)

Read one of the lane registers.

Parameters:

← ***addr*** The address of the register.

↔ ***od*** Operation synchronization and resource control.

Definition at line 232 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::OpDescr::exportTds(), sofi::fmc::Reg::Read, and sofi::fmc::OpDescr::tid().

8.9.3.16 void sofi::fmc::LaneManager::writeLaneRegister (Reg::Addr *addr*, unsigned *data*, OpDescr & *od*)

Write one of the lane registers.

Parameters:

← ***addr*** The address of the register.

↔ ***od*** Operation synchronization and resource control.

Definition at line 240 of file LaneManager.cc.

References _allocExportSide(), _export(), sofi::fmc::OpDescr::exportTds(), sofi::fmc::OpDescr::tid(), and sofi::fmc::Reg::Write.

8.9.3.17 void sofi::fmc::LaneManager::forward (const void * *pgpFrame*, unsigned *frameSize*, OpDescr & *od*)

Forward to our PGP port a frame constructed elsewhere.

Parameters:

↔ ***od*** Operation synchronization and resource control.

Normally the LaneManager constructs the PGP frames it sends, using member function arguments to fill in most of the fields. This function instead sends a (contiguous) PGP frame already constructed in memory. Only the transaction ID is changed; a new TID is allocated just as if we had constructed the packet ourselves.

Definition at line 249 of file LaneManager.cc.

References _allocExportSide(), _export(), _exportParams, sofi::fmc::OpDescr::exportTds(), and sofi::fmc::OpDescr::tid().

8.9.3.18 RcePgp::Driver& sofi::fmc::LaneManager::pgpd () const [inline]

Return a reference to the PGP driver used by this instance.

Definition at line 298 of file LaneManager.hh.

References _pgp_driver.

Referenced by _export().

8.9.3.19 void sofi::fmc::LaneManager::completed (RcePic::Tds * *tds*, RcePgp::Driver * *pgpd*) [protected, virtual]

Handle an export completion event.

Parameters:

← ***tds*** A pointer to the transaction descriptor.

Definition at line 274 of file LaneManager.cc.

8.9.3.20 void sofi::fmc::LaneManager::received (RcePic::Tds * *tds*, RcePgp::Driver *) [protected, virtual]

Handle an import event.

Parameters:

← ***tds*** A pointer to the transaction descriptor.

Definition at line 279 of file LaneManager.cc.

References _freeImportSide(), _lane, _pendingOds, sofi::fmc::OpDescr::dumpExported(), sofi::fmc::OpDescr::dumpImported(), sofi::fmc::OpDescr::setImportSide(), and sofi::fmc::PgpHeader::tid.

8.9.3.21 void sofi::fmc::LaneManager::_allocExportSide (OpDescr & *od*) [private]

Reserve the descriptors and buffers needed for an export and give them to the OpDescr.

Definition at line 296 of file LaneManager.cc.

References _exportPool, _exportWait, _freeExportSide(), _lane, _pendingOds, _tidCounter, sofi::fmc::Params::LanesPerRce, and sofi::fmc::OpDescr::setExportSide().

Referenced by _setFmcPageSize(), disableLaneWrites(), enableFmc(), enableLaneWrites(), erase(), forward(), getFlashAttr(), getFmcAttr(), getFmcCounter(), loopback(), read(), readLaneRegister(), readNoRs(), resetFmc(), write(), and writeLaneRegister().

8.9.3.22 void sofi::fmc::LaneManager::_export (OpDescr & *od*, unsigned *payloadsize* = 0) [private]

Perform an export based on the information in the OpDescr.

Definition at line 309 of file LaneManager.cc.

References sofi::fmc::OpDescr::exportTds(), and pgpd().

Referenced by _setFmcPageSize(), disableLaneWrites(), enableFmc(), enableLaneWrites(), erase(), forward(), getFlashAttr(), getFmcAttr(), getFmcCounter(), loopback(), read(), readLaneRegister(), readNoRs(), resetFmc(), write(), and writeLaneRegister().

8.9.3.23 void sofi::fmc::LaneManager::_freeExportSide (OpDescr & *od*) [private]

Release the export-related resources held by the OpDescr.

Definition at line 323 of file LaneManager.cc.

References _exportPool, _exportWait, and sofi::fmc::OpDescr::exportTds().

Referenced by _allocExportSide().

8.9.3.24 void sofi::fmc::LaneManager::_freeImportSide (OpDescr & *od*) [private]

Release the import-related resources help by the OpDescr.

Definition at line 331 of file LaneManager.cc.

References _pgp_driver, and sofi::fmc::OpDescr::importTds().

Referenced by received().

8.9.3.25 LaneManager& sofi::fmc::LaneManager::operator= (const LaneManager &) [private]

8.9.4 Member Data Documentation

8.9.4.1 unsigned sofi::fmc::LaneManager::_tidCounter [private]

Transaction counter used with lane no. to make TIDs.

Definition at line 340 of file LaneManager.hh.

Referenced by _allocExportSide().

8.9.4.2 RceSvc::Semaphore sofi::fmc::LaneManager::_exportWait [private]

Given when _exportPool goes fron empty to nonempty.

Definition at line 341 of file LaneManager.hh.

Referenced by _allocExportSide(), and _freeExportSide().

8.9.4.3 RcePic::Pool sofi::fmc::LaneManager::_exportPool [private]

Pool of resources used for export.

Definition at line 342 of file LaneManager.hh.

Referenced by _allocExportSide(), and _freeExportSide().

8.9.4.4 std::map<unsigned, OpDescr*> sofi::fmc::LaneManager::_pendingOds [private]

Mapping TID -> pending OpDescrs.

Definition at line 343 of file LaneManager.hh.

Referenced by _allocExportSide(), and received().

8.9.4.5 RcePgp::Driver& sofi::fmc::LaneManager::_pgp_driver [private]

The PGP driver used for our lane.

Definition at line 344 of file LaneManager.hh.

Referenced by _freeImportSide(), initializeLane(), and pgpd().

8.9.4.6 const RcePic::Params& sofi::fmc::LaneManager::_importParams [private]

Describes the set of import buffers.

Definition at line 345 of file LaneManager.hh.

8.9.4.7 const RcePic::Params& sofi::fmc::LaneManager::_exportParams [private]

Describes the set of export buffers.

Definition at line 346 of file LaneManager.hh.

Referenced by forward().

8.9.4.8 const unsigned sofi::fmc::LaneManager::_lane [private]

The lane number.

Definition at line 347 of file LaneManager.hh.

Referenced by _allocExportSide(), erase(), getFlashAttr(), getFmcAttr(), getFmcCounter(), read(), readNoRs(), received(), and write().

The documentation for this class was generated from the following files:

- LaneManager.hh
- LaneManager.cc

8.10 sofi::fmc::LaneRegisterHeader Struct Reference

`#include <pgp_frames.hh>`

8.10.1 Detailed Description

The PGP frame used for register access (import and export).

These frames all have one in the LSB of the DEST field and a VC of zero.

Definition at line 249 of file pgp_frames.hh.

Public Member Functions

- LaneRegisterHeader (unsigned tid, Reg::Opcode op, Reg::Addr addr, unsigned contents=0)

Public Attributes

- PgpHeader pgph
- unsigned opcode:2

  See the Opcode enum.

- unsigned _dnc1:6
- unsigned address:24

  See the Register enum.

- unsigned contents

  Register contents (Read, Write) or a bitmask (Set, Clear).

- unsigned _dnc2:14
- unsigned timeout:1

  Import only. Register access timed out.

- unsigned failure:1

  Import only. Register access failed.

- unsigned _dnc3:16

8.10.2 Constructor & Destructor Documentation

8.10.2.1 sofi::fmc::LaneRegisterHeader::LaneRegisterHeader (unsigned *tid*, Reg::Opcode *op*, Reg::Addr *addr*, unsigned *contents* = 0) [inline]

Definition at line 261 of file pgp_frames.hh.

8.10.3 Member Data Documentation

8.10.3.1 PgpHeader sofi::fmc::LaneRegisterHeader::pgph

Definition at line 251 of file pgp_frames.hh.

8.10.3.2 unsigned sofi::fmc::LaneRegisterHeader::opcode

See the Opcode enum.

Definition at line 252 of file pgp_frames.hh.

8.10.3.3 unsigned sofi::fmc::LaneRegisterHeader::_dnc1

Definition at line 253 of file pgp_frames.hh.

8.10.3.4 unsigned sofi::fmc::LaneRegisterHeader::address

See the Register enum.

Definition at line 254 of file pgp_frames.hh.

8.10.3.5 unsigned sofi::fmc::LaneRegisterHeader::contents

Register contents (Read, Write) or a bitmask (Set, Clear).

Definition at line 255 of file pgp_frames.hh.

8.10.3.6 unsigned sofi::fmc::LaneRegisterHeader::_dnc2

Definition at line 256 of file pgp_frames.hh.

8.10.3.7 unsigned sofi::fmc::LaneRegisterHeader::timeout

Import only. Register access timed out.

Definition at line 257 of file pgp_frames.hh.

8.10.3.8 unsigned sofi::fmc::LaneRegisterHeader::failure

Import only. Register access failed.

Definition at line 258 of file pgp_frames.hh.

8.10.3.9 unsigned sofi::fmc::LaneRegisterHeader::_dnc3

Definition at line 259 of file pgp_frames.hh.

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

- pgp_frames.hh

8.11 sofi::fmc::LoopbackHeader Struct Reference

`#include <pgp_frames.hh>`

8.11.1 Detailed Description

Header for the Loopback command.

This header looks almost exactly like that of the Get Pagelets command. The difference between the two operations is that for Loopback the client supplies the "page data" in the payload section of the PGP frame; flash is not actually read. The looping back is done downstream of the Reed-Solomon firmware so the client must have the pagelet size correct and "extra" data (if any) in each pagelet must be the R-S codes that the firmware expects for the data (unless your aim is to provoke an R-S check error).

Definition at line 229 of file pgp_frames.hh.

Public Member Functions

- LoopbackHeader (unsigned tid, bool reed_solomon, unsigned firstPagelet, unsigned lastPagelet)

Public Attributes

- FmcHeader fmch

8.11.2 Constructor & Destructor Documentation

8.11.2.1 sofi::fmc::LoopbackHeader::LoopbackHeader (unsigned *tid*, bool *reed_solomon*, unsigned *firstPagelet*, unsigned *lastPagelet*) [inline]

Parameters:

← ***reed_solomon*** If true then the client will provide R-S check bytes as "extra" pagelet data which will not be returned but will be used to perform error checking of the data on the way back. If false then the client must not supply "exra" pagelet bytes and the data will be returned unchecked.

← ***firstPagelet*** The number of the first pagelet to retrieve (counting from zero).

← ***lastPagelet*** The number of the last pagelet to retrieve (counting from zero).

Definition at line 238 of file pgp_frames.hh.

8.11.3 Member Data Documentation

8.11.3.1 FmcHeader sofi::fmc::LoopbackHeader::fmch

Definition at line 230 of file pgp_frames.hh.

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

- pgp_frames.hh

8.12 sofi::fmc::OpBarrier Class Reference

`#include <OpBarrier.hh>`

8.12.1 Detailed Description

Waits for completion to be posted for a set of OpDescr instances. Non-copyable.

An instance of this class will be the creator and owner of multiple OpDescr instances used in a high-level operation composed of multiple primitive operations. It doesn't matter whether the primitives execute concurrently or not. At the end of the high-level operation, make sure that the OpBarrier instance goes out of scope so that all its OpDescr instances are freed along with the resources they own.

Definition at line 58 of file OpBarrier.hh.

Public Member Functions

- OpBarrier (int n)

  Create n OpDescr objects.

- int numOps () const

  The number n given to the constructor.

- OpDescr & get (int i) const

  Get a reference to the i'th OpDescr.

- void waitForCompletion () const

  Wait for all operations to complete.

Private Member Functions

- OpBarrier (const OpBarrier &)
- OpBarrier & operator= (const OpBarrier &)

Private Attributes

- std::vector< std::tr1::shared_ptr< OpDescr > > opvec

8.12.2 Constructor & Destructor Documentation

8.12.2.1 sofi::fmc::OpBarrier::OpBarrier (int *n*) [inline]

Create n OpDescr objects.

Definition at line 62 of file OpBarrier.hh.

References opvec.

8.12.2.2 sofi::fmc::OpBarrier::OpBarrier (const OpBarrier &) [private]

8.12.3 Member Function Documentation

8.12.3.1 int sofi::fmc::OpBarrier::numOps () const [inline]

The number n given to the constructor.

Definition at line 72 of file OpBarrier.hh.

References opvec.

Referenced by waitForCompletion().

8.12.3.2 OpDescr& sofi::fmc::OpBarrier::get (int *i*) const [inline]

Get a reference to the i'th OpDescr.

Definition at line 76 of file OpBarrier.hh.

References opvec.

Referenced by checkBadBlockBytes(), and startAllFmcs().

8.12.3.3 void sofi::fmc::OpBarrier::waitForCompletion () const [inline]

Wait for all operations to complete.

Definition at line 80 of file OpBarrier.hh.

References numOps().

Referenced by findbad().

8.12.3.4 OpBarrier& sofi::fmc::OpBarrier::operator= (const OpBarrier &) [private]

8.12.4 Member Data Documentation

8.12.4.1 std::vector<std::tr1::shared_ptr<OpDescr> > sofi::fmc::OpBarrier::opvec [private]

Definition at line 91 of file OpBarrier.hh.

Referenced by get(), numOps(), and OpBarrier().

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

- OpBarrier.hh

8.13 sofi::fmc::OpDescr Class Reference

`#include <OpDescr.hh>`

8.13.1 Detailed Description

For one FMC operation hold the transaction descriptors and buffers for both the import and export transactions as well as the transaction ID. Non-copyable.

The typical use will look like this:

```
{ OpDescr od;
  Allocate export resources;
  od.setExportSide(...);
  Start the operation;
  od.waitForCompletion();
  Examine the imported results;
} # Resources are freed by ~OpDescr().
```

Definition at line 63 of file OpDescr.hh.

Public Types

- typedef std::tr1::function< void(OpDescr &)> ResourceFreeingFunc

  A type of function object used to free operation resources.

Public Member Functions

- OpDescr ()
- ~OpDescr ()

  Free the resources reserved for the operation.

- void setExportSide (unsigned tid, RcePic::Tds *expTds, const ResourceFreeingFunc &freeExp)

  Store export resources and a function object for freeing them.

- void waitForCompletion ()

  Wait until a reply has been imported.

- void setImportSide (RcePic::Tds *impTds, const ResourceFreeingFunc &freeImp)

  Store the import resources and signal that the operation is complete.

- unsigned tid () const

  Return the transaction ID of the operation.

- RcePic::Tds * exportTds () const

  Return the pointer to the export transaction descriptor.

- RcePic::Tds * importTds () const

  Return the pointer to the import transaction descriptor.

- unsigned damage () const

  Extract the "damage" word (the last word of the import payload) or zero if the transaction is not yet complete.

- void dumpExported () const

  Produce a formatted partial dump of the export frame (if any is available).

- void dumpImported () const

  Produce a formatted partial dump of the import frame (if any is available).

- unsigned bytesExported () const

  Return the transfer count for the export (0 if no export has been done).

- unsigned bytesImported () const

  Return the transfer count for the import (0 if no import has been done).

- unsigned headerBytesImported () const

  Return the true header size for the import.

- unsigned payloadBytesImported () const

  Return the true payload size for an import.

Private Member Functions

- OpDescr (const OpDescr &)
- OpDescr & operator= (const OpDescr &)
- void _dummyFree (OpDescr &)

  Used to make freeing non-existent resources harmless.

Private Attributes

- unsigned _tid

  The transaction ID.

- RcePic::Tds * _exportTds

  The pointer to the export descriptor (or null).

- ResourceFreeingFunc _freeExport

  Called in order to free export resources.

- RceSvc::Semaphore _syncsem

  Given when import is finished.

- RcePic::Tds * _importTds

  The pointer to the import descriptor (or null).

- ResourceFreeingFunc _freeImport

  Called in order to free import resources.

8.13.2 Member Typedef Documentation

8.13.2.1 typedef std::tr1::function<void (OpDescr &)> sofi::fmc::OpDescr::ResourceFreeingFunc

A type of function object used to free operation resources.

Definition at line 68 of file OpDescr.hh.

8.13.3 Constructor & Destructor Documentation

8.13.3.1 sofi::fmc::OpDescr::OpDescr () [inline]

Definition at line 158 of file OpDescr.hh.

8.13.3.2 sofi::fmc::OpDescr::~OpDescr () [inline]

Free the resources reserved for the operation.

Definition at line 167 of file OpDescr.hh.

References _freeExport, and _freeImport.

8.13.3.3 sofi::fmc::OpDescr::OpDescr (const OpDescr &) [private]

8.13.4 Member Function Documentation

8.13.4.1 void sofi::fmc::OpDescr::setExportSide (unsigned *tid*, RcePic::Tds * *expTds*, const ResourceFreeingFunc & *freeExp*) [inline]

Store export resources and a function object for freeing them.

Parameters:

← ***tid*** The 24-bit transaction ID that goes in the PGP header.

← ***expTds*** A pointer to the export transaction descriptor.

← ***freeExp*** A genealized function object used to free the export resources.

Definition at line 172 of file OpDescr.hh.

References _exportTds, _freeExport, and _tid.

Referenced by sofi::fmc::LaneManager::_allocExportSide().

8.13.4.2 void sofi::fmc::OpDescr::waitForCompletion () [inline]

Wait until a reply has been imported.

Definition at line 181 of file OpDescr.hh.

References _syncsem.

Referenced by sofi::fmc::LaneManager::initializeLane().

8.13.4.3 void sofi::fmc::OpDescr::setImportSide (RcePic::Tds * *impTds*, const ResourceFreeingFunc & *freeImp*) [inline]

Store the import resources and signal that the operation is complete.

Definition at line 185 of file OpDescr.hh.

References _freeImport, _importTds, and _syncsem.

Referenced by sofi::fmc::LaneManager::received().

8.13.4.4 unsigned sofi::fmc::OpDescr::tid () const [inline]

Return the transaction ID of the operation.

Definition at line 191 of file OpDescr.hh.

References _tid.

Referenced by sofi::fmc::LaneManager::_setFmcPageSize(), sofi::fmc::LaneManager::disableLaneWrites(), dumpExported(), dumpImported(), sofi::fmc::LaneManager::enableFmc(), sofi::fmc::LaneManager::enableLaneWrites(), sofi::fmc::LaneManager::erase(), sofi::fmc::LaneManager::forward(), sofi::fmc::LaneManager::getFlashAttr(), sofi::fmc::LaneManager::getFmcAttr(), sofi::fmc::LaneManager::getFmcCounter(), sofi::fmc::LaneManager::loopback(), sofi::fmc::LaneManager::read(), sofi::fmc::LaneManager::readLaneRegister(), sofi::fmc::LaneManager::readNoRs(), sofi::fmc::LaneManager::resetFmc(), sofi::fmc::LaneManager::write(), and sofi::fmc::LaneManager::writeLaneRegister().

8.13.4.5 RcePic::Tds * sofi::fmc::OpDescr::exportTds () const [inline]

Return the pointer to the export transaction descriptor.

Definition at line 193 of file OpDescr.hh.

References _exportTds.

Referenced by sofi::fmc::LaneManager::_export(), sofi::fmc::LaneManager::_freeExportSide(), sofi::fmc::LaneManager::_setFmcPageSize(), bytesExported(), sofi::fmc::LaneManager::disableLaneWrites(), dumpExported(), sofi::fmc::LaneManager::enableFmc(), sofi::fmc::LaneManager::enableLaneWrites(), sofi::fmc::LaneManager::erase(), sofi::fmc::LaneManager::forward(), sofi::fmc::LaneManager::getFlashAttr(), sofi::fmc::LaneManager::getFmcAttr(), sofi::fmc::LaneManager::getFmcCounter(), sofi::fmc::LaneManager::loopback(), sofi::fmc::LaneManager::read(), sofi::fmc::LaneManager::readLaneRegister(), sofi::fmc::LaneManager::readNoRs(), sofi::fmc::LaneManager::resetFmc(), sofi::fmc::LaneManager::write(), and sofi::fmc::LaneManager::writeLaneRegister().

8.13.4.6 RcePic::Tds * sofi::fmc::OpDescr::importTds () const [inline]

Return the pointer to the import transaction descriptor.

Definition at line 195 of file OpDescr.hh.

References _importTds.

Referenced by sofi::fmc::LaneManager::_freeImportSide(), bytesImported(), checkBadBlockBytes(), damage(), dumpImported(), headerBytesImported(), and payloadBytesImported().

8.13.4.7 unsigned sofi::fmc::OpDescr::damage () const

Extract the "damage" word (the last word of the import payload) or zero if the transaction is not yet complete.

Definition at line 72 of file OpDescr.cc.

References headerBytesImported(), importTds(), and payloadBytesImported().

8.13.4.8 void sofi::fmc::OpDescr::dumpExported () const

Produce a formatted partial dump of the export frame (if any is available).

Definition at line 162 of file OpDescr.cc.

References bytesExported(), anonymous_namespace{OpDescr.cc}::dumpBytes(), exportTds(), and tid().

Referenced by sofi::fmc::LaneManager::received().

8.13.4.9 void sofi::fmc::OpDescr::dumpImported () const

Produce a formatted partial dump of the import frame (if any is available).

Definition at line 113 of file OpDescr.cc.

References bytesImported(), anonymous_namespace{OpDescr.cc}::dumpBytes(), headerBytesImported(), importTds(), payloadBytesImported(), and tid().

Referenced by sofi::fmc::LaneManager::received().

8.13.4.10 unsigned sofi::fmc::OpDescr::bytesExported () const

Return the transfer count for the export (0 if no export has been done).

Definition at line 91 of file OpDescr.cc.

References exportTds().

Referenced by dumpExported().

8.13.4.11 unsigned sofi::fmc::OpDescr::bytesImported () const

Return the transfer count for the import (0 if no import has been done).

Definition at line 89 of file OpDescr.cc.

References importTds().

Referenced by dumpImported(), headerBytesImported(), and payloadBytesImported().

8.13.4.12 unsigned sofi::fmc::OpDescr::headerBytesImported () const

Return the true header size for the import.

For imports the size methods of the transfer descriptor (Tds) return the maximum values set when the import buffers were allocated. This is required so that the amount of space available for incoming data is known. This methods examines the transfer count in order to determine how big a header was actually transferred.

Definition at line 93 of file OpDescr.cc.

References bytesImported(), and importTds().

Referenced by damage(), dumpImported(), and payloadBytesImported().

8.13.4.13 unsigned sofi::fmc::OpDescr::payloadBytesImported () const

Return the true payload size for an import.

See also:

headerBytesImported()

Definition at line 103 of file OpDescr.cc.

References bytesImported(), headerBytesImported(), and importTds().

Referenced by damage(), and dumpImported().

8.13.4.14 OpDescr& sofi::fmc::OpDescr::operator= (const OpDescr &) [private]

8.13.4.15 void sofi::fmc::OpDescr::_dummyFree (OpDescr &) [inline, private]

Used to make freeing non-existent resources harmless.

Definition at line 144 of file OpDescr.hh.

8.13.5 Member Data Documentation

8.13.5.1 unsigned sofi::fmc::OpDescr::_tid [private]

The transaction ID.

Definition at line 146 of file OpDescr.hh.

Referenced by setExportSide(), and tid().

8.13.5.2 RcePic::Tds* sofi::fmc::OpDescr::_exportTds [private]

The pointer to the export descriptor (or null).

Definition at line 147 of file OpDescr.hh.

Referenced by exportTds(), and setExportSide().

8.13.5.3 ResourceFreeingFunc sofi::fmc::OpDescr::_freeExport [private]

Called in order to free export resources.

Definition at line 148 of file OpDescr.hh.

Referenced by setExportSide(), and ~OpDescr().

8.13.5.4 RceSvc::Semaphore sofi::fmc::OpDescr::_syncsem [private]

Given when import is finished.

Definition at line 150 of file OpDescr.hh.

Referenced by setImportSide(), and waitForCompletion().

8.13.5.5 RcePic::Tds* sofi::fmc::OpDescr::_importTds [private]

The pointer to the import descriptor (or null).

Definition at line 152 of file OpDescr.hh.

Referenced by importTds(), and setImportSide().

8.13.5.6 ResourceFreeingFunc sofi::fmc::OpDescr::_freeImport [private]

Called in order to free import resources.

Definition at line 153 of file OpDescr.hh.

Referenced by setImportSide(), and ~OpDescr().

The documentation for this class was generated from the following files:

- OpDescr.hh
- OpDescr.cc

8.14 sofi::fmc::Params Struct Reference

`#include <Params.hh>`

8.14.1 Detailed Description

Publicly accessible constants mostly describing the structure of flash memory.

Definition at line 45 of file Params.hh.

Public Types

- enum { LanesPerRce = 4 }
- enum { FmcsPerLane = 4 }
- enum { DimmsPerLane = 2 }
- enum { DevicesPerFmc = 4 }
- enum { ChipsPerDevice = 4 }
- enum { BlocksPerChip = 8192 }
- enum { PagesPerBlock = 64 }
- enum { PageletsPerPage = 64 }
- enum { DevicePageSize = 4096 }
- enum { DevicePageletSize = DevicePageSize / PageletsPerPage }
- enum { DevicePageExtraSize = 128 }
- enum { DevicePageSizeWithExtra = DevicePageSize + DevicePageExtraSize }
- enum { DevicePageletSizeWithExtra = DevicePageSizeWithExtra / PageletsPerPage }
- enum { FmcPageSize = DevicesPerFmc ∗ DevicePageSize }
- enum { FmcPageletSize = DevicesPerFmc ∗ DevicePageletSize }
- enum { FmcPageExtraSize = DevicesPerFmc ∗ DevicePageExtraSize }
- enum { FmcPageSizeWithExtra = DevicesPerFmc ∗ DevicePageSizeWithExtra }
- enum { FmcPageletSizeWithExtra = DevicesPerFmc ∗ DevicePageletSizeWithExtra }
- enum { ErasedByteValue = 0xff }
- enum { DeviceBadBlockMarkerOffset = DevicePageSize }
- enum { FmcBadBlockMarkerOffset = DevicesPerFmc ∗ DeviceBadBlockMarkerOffset }

8.14.2 Member Enumeration Documentation

8.14.2.1 anonymous enum

Enumerator:

LanesPerRce

Definition at line 54 of file Params.hh.

8.14.2.2 anonymous enum

Enumerator:

FmcsPerLane

Definition at line 55 of file Params.hh.

8.14.2.3 anonymous enum

Enumerator:

DimmsPerLane

Definition at line 56 of file Params.hh.

8.14.2.4 anonymous enum

Enumerator:

DevicesPerFmc

Definition at line 57 of file Params.hh.

8.14.2.5 anonymous enum

Enumerator:

ChipsPerDevice

Definition at line 58 of file Params.hh.

8.14.2.6 anonymous enum

Enumerator:

BlocksPerChip

Definition at line 59 of file Params.hh.

8.14.2.7 anonymous enum

Enumerator:

PagesPerBlock

Definition at line 60 of file Params.hh.

8.14.2.8 anonymous enum

Enumerator:

PageletsPerPage

Definition at line 61 of file Params.hh.

8.14.2.9 anonymous enum

Enumerator:

DevicePageSize

Definition at line 78 of file Params.hh.

8.14.2.10 anonymous enum

Enumerator:

DevicePageletSize

Definition at line 79 of file Params.hh.

8.14.2.11 anonymous enum

Enumerator:

DevicePageExtraSize

Definition at line 80 of file Params.hh.

8.14.2.12 anonymous enum

Enumerator:

DevicePageSizeWithExtra

Definition at line 81 of file Params.hh.

8.14.2.13 anonymous enum

Enumerator:

DevicePageletSizeWithExtra

Definition at line 82 of file Params.hh.

8.14.2.14 anonymous enum

Enumerator:

FmcPageSize

Definition at line 89 of file Params.hh.

8.14.2.15 anonymous enum

Enumerator:

FmcPageletSize

Definition at line 90 of file Params.hh.

8.14.2.16 anonymous enum

Enumerator:

FmcPageExtraSize

Definition at line 91 of file Params.hh.

8.14.2.17 anonymous enum

Enumerator:

FmcPageSizeWithExtra

Definition at line 92 of file Params.hh.

8.14.2.18 anonymous enum

Enumerator:

FmcPageletSizeWithExtra

Definition at line 93 of file Params.hh.

8.14.2.19 anonymous enum

Enumerator:

ErasedByteValue

Definition at line 97 of file Params.hh.

8.14.2.20 anonymous enum

Enumerator:

DeviceBadBlockMarkerOffset

Definition at line 107 of file Params.hh.

8.14.2.21 anonymous enum

Enumerator:

FmcBadBlockMarkerOffset

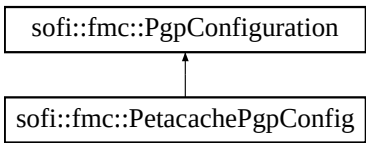
Definition at line 108 of file Params.hh.

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

- Params.hh

8.15 sofi::fmc::PetacachePgpConfig Class Reference

Inheritance diagram for sofi::fmc::PetacachePgpConfig::



8.15.1 Detailed Description

Describes the PGP configuration used for Petacache RCE boards.

The superclass describes the kind of information needed to initialize PGP. This calls actually provides that information, some defined here and some derived from pgp_frames.hh.

Definition at line 252 of file PgpDriverList.cc.

Public Member Functions

- PetacachePgpConfig ()

  Set the highest port used and the highest port possible to Port_3.

- virtual ~PetacachePgpConfig ()
- virtual const Blocks firmwareBlocks (port_number_t port) const

  Use the firmware block assignments described in Summary of PGP initialization.

- virtual const RcePic::Params picParams () const

  Just relay the information provided by pgp_frames.hh.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 sofi::fmc::PetacachePgpConfig::PetacachePgpConfig () [inline]

Set the highest port used and the highest port possible to Port_3.

Definition at line 257 of file PgpDriverList.cc.

8.15.2.2 virtual sofi::fmc::PetacachePgpConfig::~PetacachePgpConfig () [inline, virtual]

Definition at line 261 of file PgpDriverList.cc.

8.15.3 Member Function Documentation

8.15.3.1 virtual const Blocks sofi::fmc::PetacachePgpConfig::firmwareBlocks (port_number_t *port*) const [inline, virtual]

Use the firmware block assignments described in Summary of PGP initialization.

Implements sofi::fmc::PgpConfiguration.

Definition at line 265 of file PgpDriverList.cc.

8.15.3.2 virtual const RcePic::Params sofi::fmc::PetacachePgpConfig::picParams () const [inline, virtual]

Just relay the information provided by pgp_frames.hh.

Implements sofi::fmc::PgpConfiguration.

Definition at line 272 of file PgpDriverList.cc.

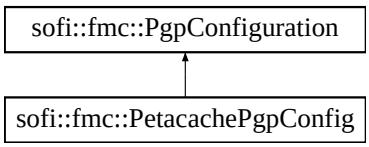
References sofi::fmc::importBufferInfo().

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

- PgpDriverList.cc

8.16 sofi::fmc::PgpConfiguration Class Reference

Inheritance diagram for sofi::fmc::PgpConfiguration::



8.16.1 Detailed Description

Abstract base class for a description of the way PGP is set up.

To initialize the PGP drivers we need to know:

- The allocation strategy (contiguous or not) for import buffers.
- The frame header size.
- The maximum frame payload size.
- The number of frame buffers used for import.
- The block numbers for the PEB, ECB, FLB and PIB used by a PGP port number.
- The highest port number in use.

Contiguous allocation means that the buffer header and payload are both put into a single allocated chunk of storage, header first. Otherwise header and payload are allocated independently.

Definition at line 167 of file PgpDriverList.cc.

Public Member Functions

- const RcePgp::Params driverParameters (port_number_t port) const

  Return the right RcePic::Params value for the given port.

- bool portUsed (port_number_t p) const

  Is the given port going to be available after PGP init?

- port_number_t highestPortUsed () const

  The maximum of the set of port numbers of all ports used.

- port_number_t maxPort () const

  The maximum port number that could be used on the host hardware.

- void report () const

  Log a report on this configuration at severity INFO.

Protected Member Functions

- PgpConfiguration (port_number_t highestUsed, port_number_t max)

  Store info about PGP ports.

- virtual ~PgpConfiguration ()
- virtual const Blocks firmwareBlocks (port_number_t) const =0

  Return the set of firmware blocks used for the given port.

- virtual const RcePic::Params picParams () const =0

  Return the description of frame buffer allocation.

Private Attributes

- bool _portUsed [(Port_3-Port_0)+1]

  Which ports are used?

- port_number_t _highestPortUsed

  The largest port number used.

- port_number_t _maxPort

  The largest port number possible for this hardware.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 sofi::fmc::PgpConfiguration::PgpConfiguration (port_number_t *highestUsed*, port_number_t *max*) [inline, protected]

Store info about PGP ports.

Parameters:

← ***highestUsed*** The highest port to be used in this initialization.

← ***max*** The maximum PGP port number available on this hardware.

Definition at line 194 of file PgpDriverList.cc.

References _portUsed.

8.16.2.2 virtual sofi::fmc::PgpConfiguration::~PgpConfiguration () [inline, protected, virtual]

Definition at line 203 of file PgpDriverList.cc.

8.16.3 Member Function Documentation

8.16.3.1 const RcePgp::Params sofi::fmc::PgpConfiguration::driverParameters (port_number_t *port*) const [inline]

Return the right RcePic::Params value for the given port.

Definition at line 171 of file PgpDriverList.cc.

References firmwareBlocks(), and picParams().

Referenced by sofi::fmc::PgpDriverList::PgpDriverList(), and report().

8.16.3.2 bool sofi::fmc::PgpConfiguration::portUsed (port_number_t *p*) const [inline]

Is the given port going to be available after PGP init?

Definition at line 176 of file PgpDriverList.cc.

References _portUsed.

Referenced by sofi::fmc::PgpDriverList::PgpDriverList(), and report().

8.16.3.3 port_number_t sofi::fmc::PgpConfiguration::highestPortUsed () const [inline]

The maximum of the set of port numbers of all ports used.

Definition at line 180 of file PgpDriverList.cc.

References _highestPortUsed.

Referenced by sofi::fmc::PgpDriverList::PgpDriverList().

8.16.3.4 port_number_t sofi::fmc::PgpConfiguration::maxPort () const [inline]

The maximum port number that could be used on the host hardware.

Definition at line 184 of file PgpDriverList.cc.

References _maxPort.

Referenced by report().

8.16.3.5 void sofi::fmc::PgpConfiguration::report () const

Log a report on this configuration at severity INFO.

Definition at line 220 of file PgpDriverList.cc.

References driverParameters(), maxPort(), and portUsed().

Referenced by sofi::fmc::PgpDriverList::create().

8.16.3.6 virtual const Blocks sofi::fmc::PgpConfiguration::firmwareBlocks (port_number_t) const [protected, pure virtual]

Return the set of firmware blocks used for the given port.

Implemented in sofi::fmc::PetacachePgpConfig.

Referenced by driverParameters().

8.16.3.7 virtual const RcePic::Params sofi::fmc::PgpConfiguration::picParams () const [protected, pure virtual]

Return the description of frame buffer allocation.

Implemented in sofi::fmc::PetacachePgpConfig.

Referenced by driverParameters().

8.16.4 Member Data Documentation

8.16.4.1 bool sofi::fmc::PgpConfiguration::_portUsed[(Port_3-Port_0)+1] [private]

Which ports are used?

Definition at line 214 of file PgpDriverList.cc.

Referenced by PgpConfiguration(), and portUsed().

8.16.4.2 port_number_t sofi::fmc::PgpConfiguration::_highestPortUsed [private]

The largest port number used.

Definition at line 215 of file PgpDriverList.cc.

Referenced by highestPortUsed().

8.16.4.3 port_number_t sofi::fmc::PgpConfiguration::_maxPort [private]

The largest port number possible for this hardware.

Definition at line 216 of file PgpDriverList.cc.

Referenced by maxPort().

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

- PgpDriverList.cc

8.17 sofi::fmc::PgpDriverList Class Reference

`#include <PgpDriverList.hh>`

8.17.1 Detailed Description

Manage a set of active PGP drivers. A Singleton class.

We need to override RcePgp::DriverList.

See also:

init_pgp()

Definition at line 68 of file PgpDriverList.hh.

Public Member Functions

- RcePgp::Driver * lookup (RcePgp::port_number_t)
  Return a pointer to the driver associated for a particular PGP port. If there is none return a null pointer.
- RcePgp::Driver * handler (RcePgp::port_number_t, RcePgp::Handler *)
  Set the handler for the driver (if any) associated with a particular PGP port.

Static Public Member Functions

- static PgpDriverList * create (const PgpConfiguration &, bool verbose=false)
  Set up PGP in a specific configuration.
- static PgpDriverList * instance ()
  Return a pointer (possibly null) to the PGP driver list.

Private Member Functions

- PgpDriverList (const PgpConfiguration &)
  Used by create() to do all the dirty work.
- ~PgpDriverList ()
  Releases no resources since a PGP initialization is is assumed to remain in effect until the next reboot (although there seems to be no hardware reason to forbid re-initialization).

8.17.2 Constructor & Destructor Documentation

8.17.2.1 sofi::fmc::PgpDriverList::PgpDriverList (const PgpConfiguration & *config*) [private]

Used by create() to do all the dirty work.

Definition at line 300 of file PgpDriverList.cc.

References sofi::fmc::PgpConfiguration::driverParameters(), sofi::fmc::PgpConfiguration::highestPortUsed(), and sofi::fmc::PgpConfiguration::portUsed().

Referenced by create().

8.17.2.2 sofi::fmc::PgpDriverList::~PgpDriverList () [private]

Releases no resources since a PGP initialization is is assumed to remain in effect until the next reboot (although there seems to be no hardware reason to forbid re-initialization).

Definition at line 319 of file PgpDriverList.cc.

8.17.3 Member Function Documentation

8.17.3.1 PgpDriverList * sofi::fmc::PgpDriverList::create (const PgpConfiguration & *config*, bool *verbose* = false) [static]

Set up PGP in a specific configuration.

Definition at line 287 of file PgpDriverList.cc.

References sofi::fmc::_pgp_driver_list, PgpDriverList(), and sofi::fmc::PgpConfiguration::report().

8.17.3.2 PgpDriverList * sofi::fmc::PgpDriverList::instance () [static]

Return a pointer (possibly null) to the PGP driver list.

Definition at line 283 of file PgpDriverList.cc.

References sofi::fmc::_pgp_driver_list.

8.17.3.3 RcePgp::Driver* sofi::fmc::PgpDriverList::lookup (RcePgp::port_number_t)

Return a pointer to the driver associated for a particular PGP port. If there is none return a null pointer.

8.17.3.4 RcePgp::Driver* sofi::fmc::PgpDriverList::handler (RcePgp::port_number_t, RcePgp::Handler *)

Set the handler for the driver (if any) associated with a particular PGP port.

Returns:

A pointer to the driver or a null pointer.

Setting a handler is simply a matter of resetting a pointer; no attempt is made to delete any existing handler.

The documentation for this class was generated from the following files:

- PgpDriverList.hh
- PgpDriverList.cc

8.18 sofi::fmc::PgpHeader Struct Reference

`#include <pgp_frames.hh>`

8.18.1 Detailed Description

The first word of all PGP frames, both import and export.

Definition at line 64 of file pgp_frames.hh.

Public Member Functions

- PgpHeader (unsigned tid, unsigned dest, unsigned vc)

Public Attributes

- unsigned tid:24

  Transaction ID arbitrarily set by the user.

- unsigned dest:6

  Destination.

- unsigned vc:2

  Virtual channel.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 sofi::fmc::PgpHeader::PgpHeader (unsigned *tid*, unsigned *dest*, unsigned *vc*) [inline]

Parameters:

← ***tid*** Transaction ID.

← ***dest*** destination code.

← ***vc*** PGP virtual channel.

Definition at line 72 of file pgp_frames.hh.

8.18.3 Member Data Documentation

8.18.3.1 unsigned sofi::fmc::PgpHeader::tid

Transaction ID arbitrarily set by the user.

Definition at line 65 of file pgp_frames.hh.

Referenced by sofi::fmc::LaneManager::received().

8.18.3.2 unsigned sofi::fmc::PgpHeader::dest

Destination.

Definition at line 66 of file pgp_frames.hh.

8.18.3.3 unsigned sofi::fmc::PgpHeader::vc

Virtual channel.

Definition at line 67 of file pgp_frames.hh.

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

- pgp_frames.hh

8.19 sofi::fmc::SetPageHeader Struct Reference

`#include <pgp_frames.hh>`

8.19.1 Detailed Description

Header for the Set Page command.

The data to be transferred is not supplied in this header but in the payload part of the PGP frame and must be exactly one page long. As usual the size of a pagelet is determined by the setting of the Reed-Solomon flag.

Definition at line 209 of file pgp_frames.hh.

Public Member Functions

- SetPageHeader (unsigned tid, unsigned lane, unsigned fmc, const FlashAddr &addr, bool reed_solomon)

  Parameters:

  ← ***addr*** *All fields are used.*

Public Attributes

- FmcHeader fmch

8.19.2 Constructor & Destructor Documentation

8.19.2.1 sofi::fmc::SetPageHeader::SetPageHeader (unsigned *tid*, unsigned *lane*, unsigned *fmc*, const FlashAddr & *addr*, bool *reed_solomon*) `[inline]`

Parameters:

← ***addr*** All fields are used.

Definition at line 213 of file pgp_frames.hh.

8.19.3 Member Data Documentation

8.19.3.1 FmcHeader sofi::fmc::SetPageHeader::fmch

Definition at line 210 of file pgp_frames.hh.

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

- pgp_frames.hh

Chapter 9

SOFI server File Documentation

9.1 Damage.hh File Reference

9.1.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares class Damage.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-29 15:19:05 -0700 (Mon, 29 Mar 2010) $ by $Author: tether $.

Revision number:

$Revision: 1120 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/Damage.hh $

Credits:

SLAC

Definition in file Damage.hh.

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::Damage

  The structure of the "damage" word at the end of FMC-type import frames.

Defines

- #define SOFI_FMC_DAMAGE_HH

9.1.2 Define Documentation

9.1.2.1 #define SOFI_FMC_DAMAGE_HH

Definition at line 37 of file Damage.hh.

9.2 doc.hh File Reference

9.2.1 Detailed Description

Facility:

RCE Petacache

Abstract:

Contains doxygen-format notes on the development of FMC code.

Author:

Stephen Tether<tether@slac.stanford.edu>

Date created:

2010/03/18

Last commit:

$Date: 2010-03-29 15:17:54 -0700 (Mon, 29 Mar 2010) $ by $Author: tether $.

Revision number:

$Revision: 1119 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/doc.hh $

Credits:

SLAC

Definition in file doc.hh.

9.3 docmain.hh File Reference

9.3.1 Detailed Description

Facility:

RCE Petacache

Abstract:

Define for project "sofi" the doxygen main page contents, namespace docs and any other documentation that has no other good place to sit.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2010/03/29

Last commit:

$Date: 2010-03-29 15:17:54 -0700 (Mon, 29 Mar 2010) $ by $Author: tether $.

Revision number:

$Revision: 1119 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/docmain.hh $

Credits:

SLAC

Definition in file docmain.hh.

Namespaces

- namespace sofi
- namespace sofi::fmc

9.4 findbadblocks.cc File Reference

9.4.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Module that scans for bad flash blocks.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2010/01/27

Last revision:

$Date: 2010-03-18 13:38:21 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1106 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/findbadblocks.cc $

Credits:

SLAC

Definition in file findbadblocks.cc.

```
#include <cstdio>
#include <exception>
#include <vector>
#include <tr1/memory>
#include "rce/pgp/Driver.hh"
#include "rce/pic/Params.hh"
#include "rce/pic/Pool.hh"
#include "rce/pic/Tds.hh"
#include "rce/service/Exception.hh"
```

```
#include "quarks/service/Logger.hh"
#include "quarks/service/PtyLogger.hh"
#include "sofi/fmc/FlashAddr.hh"
#include "sofi/fmc/LaneManager.hh"
#include "sofi/fmc/OpBarrier.hh"
#include "sofi/fmc/OpDescr.hh"
#include "sofi/fmc/Params.hh"
#include "sofi/fmc/PgpDriverList.hh"
#include "sofi/fmc/src/pgp_frames.hh"
```

Namespaces

- namespace anonymous_namespace{findbadblocks.cc}

Typedefs

- typedef vector< shared_ptr< LaneManager > > LaneMgrVec

Functions

- unsigned opDescIndex (unsigned lanenum, unsigned fmcnum)
- void findbad ()
- void createLaneManagers (LaneMgrVec &)
- void startAllFmcs (unsigned blocknum, unsigned chipnum, unsigned pagenum, LaneMgrVec &lanemgr, OpBarrier &readBarrier)
- void checkBadBlockBytes (unsigned blocknum, unsigned chipnum, OpBarrier &readBarrier)
- void dumpPage (const unsigned char *data, unsigned nbytes)
- void rce_appmain (void *)

Variables

- unsigned anonymous_namespace{findbadblocks.cc}::LANES_PER_RCE = Params::LanesPerRce
- unsigned anonymous_namespace{findbadblocks.cc}::FMCS_PER_LANE = Params::FmcsPerLane
- unsigned anonymous_namespace{findbadblocks.cc}::CHIPS_PER_DEVICE = Params::ChipsPerDevice
- unsigned anonymous_namespace{findbadblocks.cc}::BLOCKS_PER_CHIP = Params::BlocksPerChip

9.4.2 Typedef Documentation

9.4.2.1 typedef vector<shared_ptr<LaneManager> > LaneMgrVec

Definition at line 85 of file findbadblocks.cc.

9.4.3 Function Documentation

9.4.3.1 void checkBadBlockBytes (unsigned *blocknum*, unsigned *chipnum*, OpBarrier & *readBarrier*)

Definition at line 232 of file findbadblocks.cc.

References dumpPage(), sofi::fmc::OpBarrier::get(), sofi::fmc::OpDescr::importTds(), and opDescIndex().

Referenced by findbad().

9.4.3.2 void createLaneManagers (LaneMgrVec & *lanemgr*)

Definition at line 176 of file findbadblocks.cc.

References sofi::fmc::exportBufferInfo(), sofi::fmc::importBufferInfo(), and anonymous_namespace{findbadblocks.cc}::LANES_PER_RCE.

Referenced by findbad().

9.4.3.3 void dumpPage (const unsigned char * *data*, unsigned *nbytes*)

Definition at line 281 of file findbadblocks.cc.

Referenced by checkBadBlockBytes().

9.4.3.4 void findbad ()

Definition at line 140 of file findbadblocks.cc.

References anonymous_namespace{findbadblocks.cc}::BLOCKS_PER_CHIP, checkBadBlockBytes(), anonymous_namespace{findbadblocks.cc}::CHIPS_PER_DEVICE, createLaneManagers(), sofi::fmc::init_pgp(), startAllFmcs(), and sofi::fmc::OpBarrier::waitForCompletion().

Referenced by rce_appmain().

9.4.3.5 unsigned opDescIndex (unsigned *lanenum*, unsigned *fmcnum*) [inline]

Definition at line 87 of file findbadblocks.cc.

Referenced by checkBadBlockBytes(), and startAllFmcs().

9.4.3.6 void rce_appmain (void *)

Definition at line 108 of file findbadblocks.cc.

References findbad().

9.4.3.7 void startAllFmcs (unsigned *blocknum*, unsigned *chipnum*, unsigned *pagenum*, LaneMgrVec & *lanemgr*, OpBarrier & *readBarrier*)

Definition at line 202 of file findbadblocks.cc.

References sofi::fmc::OpBarrier::get(), and opDescIndex().

Referenced by findbad().

9.5 FlashAddr.hh File Reference

9.5.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares class FlashAddr.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-02-16 15:49:02 -0800 (Tue, 16 Feb 2010) $ by $Author: tether $

Revision number:

$Revision: 1076 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/FlashAddr.hh $

Credits:

SLAC

Definition in file FlashAddr.hh.

```
#include "sofi/fmc/Params.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::FlashAddr

  Used to build the Address field used in the second word of FMC-type PGP frames.

Defines

- #define SOFI_FMC_FLASHADDR_HH

9.5.2 Define Documentation

9.5.2.1 #define SOFI_FMC_FLASHADDR_HH

Definition at line 37 of file FlashAddr.hh.

9.6 fmc_const.hh File Reference

9.6.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Defines FMC command codes and performance counter numbers.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-02-16 15:49:02 -0800 (Tue, 16 Feb 2010) $ by $Author: tether $

Revision number:

$Revision: 1076 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/fmc_const.hh $

Credits:

SLAC

Definition in file fmc_const.hh.

Namespaces

- namespace sofi
- namespace sofi::fmc
- namespace sofi::fmc::Cmd
- namespace sofi::fmc::Ctr

Defines

- #define SOFI_FMC_FMCCONST_HH

Enumerations

- enum sofi::fmc::Cmd::Code {

  sofi::fmc::Cmd::GetFlashAttributes = 0, sofi::fmc::Cmd::GetFmcAttributes = 1, sofi::fmc::Cmd::GetCounter = 2, sofi::fmc::Cmd::EraseBlock = 3,

  sofi::fmc::Cmd::GetBlocks = 4, sofi::fmc::Cmd::SetPage = 5, sofi::fmc::Cmd::Loopback = 7 }

  The legal FMC-frame command code values.

- enum sofi::fmc::Ctr::Counter {

  sofi::fmc::Ctr::Reads, sofi::fmc::Ctr::Writes, sofi::fmc::Ctr::Erasures, sofi::fmc::Ctr::DeviceErrors,

  sofi::fmc::Ctr::ArbTime, sofi::fmc::Ctr::BusyTime, sofi::fmc::Ctr::ArbTimeouts, sofi::fmc::Ctr::CommandCongestion }

  The set of FMC counters.

9.6.2 Define Documentation

9.6.2.1 #define SOFI_FMC_FMCCONST_HH

Definition at line 37 of file fmc_const.hh.

9.7 lane_const.hh File Reference

9.7.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Defines names, opcodes and bit positions for lane registers.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-18 13:39:56 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1107 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/lane_const.hh $

Credits:

SLAC

Definition in file lane_const.hh.

Namespaces

- namespace sofi
- namespace sofi::fmc
- namespace sofi::fmc::Reg

Defines

- #define SOFI_FMC_LANECONST_HH

Enumerations

- enum sofi::fmc::Reg::Addr {

  sofi::fmc::Reg::Version = 0, sofi::fmc::Reg::Status = 1, sofi::fmc::Reg::IdValid = 2, sofi::fmc::Reg::Fmc01IdLow = 4,

  sofi::fmc::Reg::Fmc01IdHigh = 5, sofi::fmc::Reg::Fmc23IdLow = 6, sofi::fmc::Reg::Fmc23IdHigh = 7, sofi::fmc::Reg::Fmc0Csr = 16,

  sofi::fmc::Reg::Fmc1Csr = 17, sofi::fmc::Reg::Fmc2Csr = 18, sofi::fmc::Reg::Fmc3Csr = 19 }

  The set of known register addresses for each lane.

- enum sofi::fmc::Reg::Opcode { sofi::fmc::Reg::Read, sofi::fmc::Reg::Write, sofi::fmc::Reg::Set, sofi::fmc::Reg::Clear }

  Lane register opcodes.

- enum sofi::fmc::Reg::CsrBits { sofi::fmc::Reg::Reset = 0x1, sofi::fmc::Reg::Enable = 0x8, sofi::fmc::Reg::PageSize = 0x01000000 }

  Bits in each FMC CSR.

- enum sofi::fmc::Reg::StatusBits { sofi::fmc::Reg::RotateDisplay = 0x40000000, sofi::fmc::Reg::EnableFlashWrite = 0x80000000 }

  Bits in the lane status register.

Functions

- Addr sofi::fmc::Reg::operator++ (Addr &a)

9.7.2 Define Documentation

9.7.2.1 #define SOFI_FMC_LANECONST_HH

Definition at line 37 of file lane_const.hh.

9.8 LaneManager.cc File Reference

9.8.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Implements class LaneManager.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-22 12:09:42 -0700 (Mon, 22 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1110 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/LaneManager.cc $

Credits:

SLAC

Definition in file LaneManager.cc.

```
#include <tr1/functional>
#include <cstring>
#include <map>
#include "rce/service/Cache.hh"
#include "sofi/fmc/LaneManager.hh"
#include "sofi/fmc/OpDescr.hh"
#include "sofi/fmc/Params.hh"
#include "sofi/fmc/src/pgp_frames.hh"
#include "sofi/fmc/src/fmc_const.hh"
#include "sofi/fmc/src/lane_const.hh"
#include "quarks/service/Logger.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

9.9 LaneManager.hh File Reference

9.9.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares class LaneManager.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-18 13:11:08 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1100 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/LaneManager.hh $

Credits:

SLAC

Definition in file LaneManager.hh.

```
#include <map>
#include "rce/pgp/Driver.hh"
#include "rce/pgp/Handler.hh"
#include "rce/service/Semaphore.hh"
#include "sofi/fmc/FlashAddr.hh"
#include "sofi/fmc/src/fmc_const.hh"
#include "sofi/fmc/src/lane_const.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::LaneManager

  Construct and transmit (export) FMC commands, receive (import) replies. Non-copyable.

Defines

- #define SOFI_FMC_LANEMANAGER_HH

9.9.2 Define Documentation

9.9.2.1 #define SOFI_FMC_LANEMANAGER_HH

Definition at line 37 of file LaneManager.hh.

9.10 OpBarrier.hh File Reference

9.10.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares class OpBarrier.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2010/01/22

Last revision:

$Date: 2010-03-18 13:12:53 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1101 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/OpBarrier.hh $

Credits:

SLAC

Definition in file OpBarrier.hh.

```
#include <vector>
#include <tr1/memory>
#include "sofi/fmc/OpDescr.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::OpBarrier

  Waits for completion to be posted for a set of OpDescr instances. Non-copyable.

Defines

- #define SOFI_FMC_OPBARRIER_HH

9.10.2 Define Documentation

9.10.2.1 #define SOFI_FMC_OPBARRIER_HH

Definition at line 37 of file OpBarrier.hh.

9.11 OpDescr.cc File Reference

9.11.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Implements class OpDescr.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-29 15:25:10 -0700 (Mon, 29 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1121 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/OpDescr.cc $

Credits:

SLAC

Definition in file OpDescr.cc.

```
#include <algorithm>
#include <cstdio>
#include "quarks/service/Logger.hh"
#include "sofi/fmc/Damage.hh"
#include "sofi/fmc/OpDescr.hh"
```

Namespaces

- namespace anonymous_namespace{OpDescr.cc}
- namespace sofi
- namespace sofi::fmc

Functions

- void anonymous_namespace{OpDescr.cc}::dumpBytes (const unsigned char *data, unsigned nbytes)

9.12 OpDescr.hh File Reference

9.12.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares the OpDescr class.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-29 15:25:10 -0700 (Mon, 29 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1121 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/OpDescr.hh $

Credits:

SLAC

Definition in file OpDescr.hh.

```
#include <tr1/functional>
#include "rce/pic/Tds.hh"
#include "rce/service/Semaphore.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::OpDescr

  For one FMC operation hold the transaction descriptors and buffers for both the import and export transactions as well as the transaction ID. Non-copyable.

Defines

- #define SOFI_FMC_OPDESCR_HH

9.12.2 Define Documentation

9.12.2.1 #define SOFI_FMC_OPDESCR_HH

Definition at line 37 of file OpDescr.hh.

9.13 Params.hh File Reference

9.13.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares the Params struct.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-02-16 15:49:02 -0800 (Tue, 16 Feb 2010) $ by $Author: tether $

Revision number:

$Revision: 1076 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/Params.hh $

Credits:

SLAC

Definition in file Params.hh.

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- struct sofi::fmc::Params

  Publicly accessible constants mostly describing the structure of flash memory.

Defines

- #define SOFI_FMC_PARAMS_HH

9.13.2 Define Documentation

9.13.2.1 #define SOFI_FMC_PARAMS_HH

Definition at line 37 of file Params.hh.

9.14 pgp_frames.hh File Reference

9.14.1 Detailed Description

Facility:

RCE SOFI

Abstract:

PGP frame layout structs, max PGP header sizes, etc.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-18 13:43:39 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1108 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/pgp_frames.hh $

Credits:

SLAC

Warning:

When you add a new type of header make sure to update maxPgpHeaderSize() as well.
All the header structs must be Plain Old Data, no virtual anything.

Definition in file pgp_frames.hh.

```
#include <algorithm>
#include "rce/pic/Params.hh"
#include "sofi/fmc/Params.hh"
#include "sofi/fmc/FlashAddr.hh"
#include "sofi/fmc/src/fmc_const.hh"
#include "sofi/fmc/src/lane_const.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- struct sofi::fmc::PgpHeader

  The first word of all PGP frames, both import and export.

- struct sofi::fmc::FmcHeader

  The first three words of all frames for Flash Core operations and Loopback, import and export.

- struct sofi::fmc::GetFlashAttributesHeader

  Header for the GetFlashAttributes command.

- struct sofi::fmc::GetFmcAttributesHeader

  Header for the GetFmcAttributes command.

- struct sofi::fmc::GetFmcCounterHeader

  Header for the GetCounter command.

- struct sofi::fmc::EraseBlockHeader

  Header for the Erase Block command.

- struct sofi::fmc::GetPageletsHeader

  Header for the Get Blocks (Get Pagelets) command.

- struct sofi::fmc::SetPageHeader

  Header for the Set Page command.

- struct sofi::fmc::LoopbackHeader

  Header for the Loopback command.

- struct sofi::fmc::LaneRegisterHeader

  The PGP frame used for register access (import and export).

Defines

- #define SOFI_FMC_PGPFRAMES_HH

Functions

- size_t sofi::fmc::maxPgpHeaderSize ()

  The maximum number of bytes in any of the PGP headers defined in this file.

- unsigned sofi::fmc::numExportBuffers ()

  Return the number of export frame buffers used.

- unsigned sofi::fmc::numImportBuffers ()

  Return the number of export frame buffers used.

- unsigned sofi::fmc::maxExportPayloadSize ()

  Return the maximum payload size to be used for any PGP frame used for export.

- unsigned sofi::fmc::maxImportPayloadSize ()

  Return the maximum payload size to be used for any PGP frame used for import.

- const RcePic::Params & sofi::fmc::importBufferInfo ()

  Return a description of the frame buffer allocation for imports.

- const RcePic::Params & sofi::fmc::exportBufferInfo ()

  Return a description of the frame buffer allocation for exports.

9.14.2 Define Documentation

9.14.2.1 #define SOFI_FMC_PGPFRAMES_HH

Definition at line 42 of file pgp_frames.hh.

9.15 PgpDriverList.cc File Reference

9.15.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Implements class PgpDriverList.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-29 15:26:30 -0700 (Mon, 29 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1122 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/PgpDriverList.cc $

Credits:

SLAC

Definition in file PgpDriverList.cc.

```
#include "sofi/fmc/PgpDriverList.hh"
#include "sofi/fmc/src/pgp_frames.hh"
#include "rce/pgp/Driver.hh"
#include "rce/pgp/Handler.hh"
#include "rce/pgp/Params.hh"
#include "rce/pgp/Stats.hh"
#include "rce/pic/Params.hh"
#include "quarks/service/Logger.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::PgpConfiguration

  Abstract base class for a description of the way PGP is set up.

- class sofi::fmc::PetacachePgpConfig

  Describes the PGP configuration used for Petacache RCE boards.

Functions

- port_number_t sofi::fmc::operator++ (port_number_t &p)
- void sofi::fmc::init_pgp (bool verbose=false)

  Initialize all PGP drivers for the Petacache. Installs the single instance of PgpDriverList.

Variables

- PgpDriverList * sofi::fmc::_pgp_driver_list = 0

9.16 PgpDriverList.hh File Reference

9.16.1 Detailed Description

Facility:

RCE SOFI

Abstract:

Declares class PgpDriverList.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-18 13:15:04 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1102 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/PgpDriverList.hh $

Credits:

SLAC

Definition in file PgpDriverList.hh.

```
#include "rce/pgp/Driver.hh"
#include "rce/service/List.hh"
```

Namespaces

- namespace sofi
- namespace sofi::fmc

Classes

- class sofi::fmc::PgpDriverList

  Manage a set of active PGP drivers. A Singleton class.

Defines

- #define SOFI_FMC_PGPDRIVERLIST_HH

Functions

- void sofi::fmc::init_pgp (bool verbose=false)

  Initialize all PGP drivers for the Petacache. Installs the single instance of PgpDriverList.

9.16.2 Define Documentation

9.16.2.1 #define SOFI_FMC_PGPDRIVERLIST_HH

Definition at line 37 of file PgpDriverList.hh.

9.17 testfmcs.cc File Reference

9.17.1 Detailed Description

Facility:

RCE SOFI

Abstract:

A module to exercise all the FMCs on an RCE.

Author:

Stephen Tether <tether@slac.stanford.edu>

Date created:

2009/12/19

Last revision:

$Date: 2010-03-18 13:45:50 -0700 (Thu, 18 Mar 2010) $ by $Author: tether $

Revision number:

$Revision: 1109 $

Location in repository:

$HeadURL: file:///reg/g/npa/svnrepo/sofi/trunk/fmc/src/testfmcs.cc $

Credits:

SLAC

Definition in file testfmcs.cc.

```
#include "rtems.h"
#include <exception>
#include <vector>
#include "rce/service/Exception.hh"
#include "rce/service/Semaphore.hh"
#include "quarks/service/Logger.hh"
#include "quarks/service/PtyLogger.hh"
```

Functions

- void myterminate ()
- void rce_appmain (void *)

9.17.2 Function Documentation

9.17.2.1 void myterminate ()

Definition at line 57 of file testfmcs.cc.

Referenced by rce_appmain().

9.17.2.2 void rce_appmain (void *)

Definition at line 62 of file testfmcs.cc.

References myterminate().

Chapter 10

SOFI server Page Documentation

10.1 The FMC package

This package contains the code that constructs, imports and exports the various PGP frames used to communicate with the Flash Memory Controller firmware and the "lane" (or "integration") firmware that binds four FMCs into a single lane.

Directory structure:

- ./doc.hh Extra doxygen input.
- ./∗.hh Public declarations.
- ./src/∗.hh Private declarations.
- ./src/∗.cc Implementation.

Generally each header and each implementation file deals with only one class except for the case of several small classes that are closely related.

10.2 Miscellaneous notes on the FMC library code

10.2.1 RCE hardware

The flash devices currently used for the Petacache have an undocumented change in their page size: it's 4K + 128 instead of the 2K + 64 we find in online spec sheets for Samsung K9XXG08UXM devices. There's a page-size bit you have to set in the FMC's CSR register in order to accomodate the larger page.

The PowerPC 405 data cache is set to operate in write-back mode instead of the less efficient (if safer) write-through mode. As a result new values for memory locations will hang around in the cache for an unpredictable length of time before being written to main memory. Furthermore, the cache is not visible to external devices such as the PGP ports used to communicate with the FMCs; they transfer data directly to and from main memory. If you use cached memory for I/O buffers then before asking a device to read a buffer you have to force any new values for buffer locations out of the cache and into main memory (flushing). Before asking a device to write to a buffer you have to throw away any cached values for buffer locations (invalidation); otherwise your attempts to read from the buffer may return what's in the cache instead of what was placed in main memory by the device.

The current implementation of the PGP drivers allocates the frame buffers in cached memory, so there are plenty of calls to RceSvc:Cache::flush()/invalidate() throughout the PGP driver and related classes.

10.2.2 FMC application modules

The bad block finding and FMC test modules run with notification of export completions turned off (by calling RcePgp::Driver:post() with the "complete" argument set to Driver::DoNotComplete). Instead they wait for an import frame with the same transaction ID as the export frame, thus ensuring that the requested operation is finished rather than just queued.

10.2.3 C++ standard library

Beware of std::tr1::unordered_map<>. It uses a floating point hash function. The Virtex PowerPCs have no hardware floating point so using this kind of map generates references to a lot of floating point emulation routines that are not in the RCE core. Besides, software floating point is inefficient. We should use hash tables that use integer arithmetic only.

10.2.4 PGP

The maximum payload size for FMC PGP frames is 16,900. That's just big enough for one full page of flash memory (with the "extra" bytes) plus a four-byte damage word. Try using a larger payload size and you'll get PIC errors and possibly suspended PGP service tasks.

The Transaction DeScriptor described in the RCE manual is implemented using class RcePic::Tds. There is an associated class confusingly named RcePic::Descr an instance of which holds a pointer to a corresponding Tds instance (and the Tds holds a reciprocal pointer to the Descr). The memory pool object RcePic::Pool holds instances of Descr but RcePgp::Driver passes Tds pointers to handlers. A Descr instance holds a pointer to a Driver instance as well as a flag which determines whether Tds::flush() should flush cache entries for the payload buffer as well as for the Tds object proper.

Class RcePgp::DriverList determines whether the PGP subsystem has been initialized by checking a static variable containing a pointer to a DriverList (or null). I've had to replace DriverList by sofi::fmc::PgpDriverList so that approach won't work in a module run from the shell using runTask; any such variable in the module will have a null value each time the module is (re)loaded. At the moment that

means rebooting between runs to avoid double-initing PGP. If we move PgpDriverList to the core then the problem will go away. Failing that we could test for the existence of one of the PGP service threads.

The PGP frame document at http://www.slac.stanford.edu/exp/npa/design/frames.pdf doesn't say which words belong to the header and which to the payload in each case. For most of the Petacache frames it doesn't matter since they're header-only and the import frame that comes back has the same size header as the export frame. This isn't true for the FMC commands Get Flash Attributes (0), Get Blocks (4) and Loopback (7). For them the import frame header is padded with zeros to 32 bytes, which if the payload is allocated right after the header forces the payload to a cache line boundary. Since all PGP frames for a given port have to have the same header size it means you have to use a header size of 32 instead of 16 when initializing PGP.

You can't rely on Tds::headersize() and Tds::payloadsize() when trying to figure out the true sizes for an newly received import frame. These functions return the values last set for them and, unless you set them yourself, for import buffers they were last set when the buffers were allocated, using the maximum sizes given to the PGP init code. You have to use the transfer count obtained from Tds::transferred(). If the transfer count T is less than or equal to the maximum header size H then there is no payload and the header size is T; otherwise the header size is H and the payload size is T - H. If you like you can use payloadsize(unsigned) and headersize(unsigned) to set the values in the Tds once you've calculated them, but you'll have to set them back to the maxima before you release the Tds since the PGP driver uses these member functions to determine how much space was actually allocated. Note that Tds::size() returns the sum of headersize() and payloadsize().

You're expected to set the true payload size with Tds::payloadsize(unsigned) before posting an export. Just one wart: if the payload size is zero then you have to call RcePgp::Driver::post() with the "contiguous" argument set to RcePgp::Driver::Contiguous; otherwise use the value the reflects the true state of affairs. This is the result of a requirement in the protocol plugin.

10.3 Summary of PGP initialization

A given PGP port may communicate with a Rear Transition Module (RTM) or with a lane of flash memory. At present an RCE can have only one of those two types of PGP ports; a Petacache RCE has four lanes and an RTM RCE has three RTM ports.

The two types of PGP ports are functionally identical; the reason for the distinction lies in the extremely limited number of firmware blocks available for I/O ports in Virtex-4 RCEs. These blocks are used both by ethernet ports and by PGP ports.

Block types:

- PEB - Pending Export Block (manages exported frames)
- ECB - Export Control Block (posts exported frame completion/error status)
- FLB - Free List Block (manages buffers for imported frames)
- PIB - Pending Import Block (reads into FLB buffers, posts reception notices)

Each I/O port uses up one PEB and one PIB and there are only five of each. Therefore a Petacache RCE, which needs one PGP port for each of the four flash lanes, has room for only one ethernet port. The other kind of RCE, used for the Linear Coherent Light Source DAQ, uses only three PGP ports for the Rear Transition Module and so may have two ethernet ports. In a Petacache RCE PEBs no. 1-4 and PIBs nos. 1-4 are used for PGP; other RCEs use PEBs 2-4 and PIBs 2-4. PGP port zero uses the lowest numbered PEB and PIB, port one the next highest numbered, etc.

All the PGP ports share a single ECB and a single FLB. On a Petacache RCE ECB no. 1 and FLB no. 1 are used. Other RCEs use ECB 2 and FLB 2.

The constructor for RcePgp::Driver needs to know if the new driver is for the port with the highest number (boolean argument "last"). For that driver the constructor will create a so-called completion task which handles completion and error notices from the common ECB. Only one such thread should be created because only one ECB is used. For reasons unknown things work only if the driver associated with the completion thread is the one with the highest port number.

For each initialized port two RTEMS tasks named pxrcv and pxfull are created where x stands for the port number. The pxrcv task reads reception notices from the port's PIB and passes the addresses of the transaction descriptors to the port driver's designated "handler". The pxfull task waits for the port's PEB to signal an AlmostFull event (which refers to the state of the PEB's transaction FIFO). When that happens the task takes a semaphore which will block further export posts to the PEB. The task then waits for the PEB to signal an AlmostEmpty event at which time the task will release the semaphore and go back to waiting for AlmostFull.

The highest-numbered port initialized will have a third task associated with it named pxcmplt. This is the export-completion task described above.

Here's a task listing created on a Petacache RCE after PGP initialization. Note that RTEMS retains only the first four characters of a task's name.

```
  ID       NAME           PRI  STATE MODES   EVENTS    WAITID  WAITARG  NOTES
------------------------------------------------------------------------------
0a010003   ehc0            80 READY  P:T:nA    NONE   f7bff7bf 0x1ef548
0a010004   ehr0            80 READY  P:T:nA    NONE   1a010013 0x1ef548
0a010005   ehf0            70 Wevnt  P:T:nA    NONE   08400840 0x1ef548
0a010006   ntwk            80 READY  P:T:nA    NONE   1a010013 0x1ef548
0a010008   TNTD            50 Wevnt  P:T:nA    NONE   1a010013 0x1ef548
0a010009   pty0            50 READY  P:T:nA    NONE   1a010013 0x1ef548
0a01000a   RPCd            80 Wevnt  P:T:nA    NONE   f7bff7bf 0x1ef548
0a01000c   p0rc            90 Wevnt  P:T:nA    NONE   1a010013 0x1ef548
```

```
0a01000d   p0fu            70 Wevnt  P:T:nA    NONE   b7bdf3bf 0x1ef548
0a01000e   p1rc            90 Wevnt  P:T:nA    NONE   08400840 0x1ef548
0a01000f   p1fu            70 Wevnt  P:T:nA    NONE   08400840 0x1ef548
0a010010   p2rc            90 Wevnt  P:T:nA    NONE   f7bff7bf 0x1ef548
0a010011   p2fu            70 Wevnt  P:T:nA    NONE   08400840 0x1ef548
0a010012   p3cm            91 Wevnt  P:T:nA    NONE   08400840 0x1ef548
0a010013   p3rc            90 Wevnt  P:T:nA    NONE   e7bff7bf 0x1ef548
0a010014   p3fu            70 Wevnt  P:T:nA    NONE   08400840 0x1ef548
```

Index