LIBIOPMP

Generated by Doxygen 1.14.0

Chapter 1

libiopmp - A Library to Program RISC-V IOPMP

The `libiopmp` is intended to be driver of RISC-V IOPMP which:

- Complies with IOPMP specification **v0.8**
- Operates one or multiple IOPMPs
- Supports several IOPMP models and configurations
- Extensible for adding vendor-customized IOPMP driver
- Supports IOPMP with Multi-Faults Record (MFR) extension
- Supports IOPMP with Secondary Permission Setting (SPS) extension
- Supports IOPMP with Message-Signaled Interrupts (MSI) extension

1.1 Adjust `config.mk`

The `libiopmp` has a `config.mk` configuration file which let you modularize your `libiopmp` to reduce the code size.

- `DEBUG`: Turn on this option to build `libiopmp` without compiler optimization and assert() macro will be enabled
- `CFG_IOPMP_REF_MODEL`: Turn on this option to enable compiling of register read/write interface as weak functions. This is useful if the IOPMP you operate is simulated by the reference model. If you want to control real IOPMP you just turn off this option.
- `CFG_IOPMP_DRV_FULL`: Turn on this option to enable compiling of driver for full model
- `CFG_IOPMP_DRV_RAPID_K`: Turn on this option to enable compiling of driver for rapid-k model
- `CFG_IOPMP_DRV_DYNAMIC_K`: Turn on this option to enable compiling of driver for dynamic-k model
- `CFG_IOPMP_DRV_ISOLATION`: Turn on this option to enable compiling of driver for isolation model
- `CFG_IOPMP_DRV_COMPACT_K`: Turn on this option to enable compiling of driver for compact-k model

- CFG_IOPMP_DRV_SRCMD_FMT_1_MDCFG_FMT_2: Turn on this option to enable compiling of driver for SRCMD_FMT=1 & MDCFG_FMT=2
- CFG_IOPMP_DRV_SRCMD_FMT_2_MDCFG_FMT_0: Turn on this option to enable compiling of driver for SRCMD_FMT=2 & MDCFG_FMT=0
- CFG_IOPMP_DRV_SRCMD_FMT_2_MDCFG_FMT_1: Turn on this option to enable compiling of driver for SRCMD_FMT=2 & MDCFG_FMT=1
- CFG_IOPMP_DRV_SRCMD_FMT_2_MDCFG_FMT_2: Turn on this option to enable compiling of driver for SRCMD_FMT=2 & MDCFG_FMT=2
- CFG_IOPMP_DRV_SPS_EXTENSION: Turn on this option to enable compiling of driver for Secondary Permission Setting (SPS) extension

1.2 Compilation

libiopmp can be built by host compiler or RISC-V toolchain. The former one is useful when testing libiopmp using the reference model, while the later one is necessary if you want to use libiopmp on RISC-V platforms.

1.2.1 Compiled by Host GCC

```
~/libiopmp$ make
 CC        libiopmp.o
 CC        iopmp_drv_common.o
 CARRAY    iopmp_drivers.carray.c
 CC        iopmp_drivers.carray.o
 CC        iopmp_drv_full.o
 CC        iopmp_drv_rapid_k.o
 CC        iopmp_drv_dynamic_k.o
 CC        iopmp_drv_isolation.o
 CC        iopmp_drv_compact_k.o
 CC        iopmp_drv_srcmd_fmt_1_mdcfg_fmt_2.o
 CC        iopmp_drv_srcmd_fmt_2_mdcfg_fmt_0.o
 CC        iopmp_drv_srcmd_fmt_2_mdcfg_fmt_1.o
 CC        iopmp_drv_srcmd_fmt_2_mdcfg_fmt_2.o
 AR        lib/libiopmp.a
```

1.2.2 Compiled by RISC-V toolchain

To compiled libiopmp by RISC-V toolchain, you need to add the "path to your RISC-V toolchain" into $PATH environment variable, and input the following command:

For RV32 target:

```
~/libiopmp$ export CROSS_COMPILE=riscv32-unknown-elf-
~/libiopmp$ make
```

For RV64 target:

```
~/libiopmp$ export CROSS_COMPILE=riscv64-unknown-elf-
~/libiopmp$ make
```

1.3 Usage

Assume the directory path to libiopmp is $(LIBIOPMP_DIR), the output library archive will be $(LIBIOPMP↩_DIR)/build/lib/libiopmp.a. All the data structures and the APIs are declared in $(LIBIOPMP_↩DIR)/include/libiopmp.h header file.

Add the path to the library and header file into your build system. Assume CFLAGS represents the compiler flags and LDFLAGS represents the linker flags, please add the path to libiopmp.h into CFLAGS and libiopmp.a into LDFLAGS accordingly:

```
CFLAGS  += -I$(LIBIOPMP_DIR)/include
LDFLAGS += $(LIBIOPMP_DIR)/build/lib/libiopmp.a
```

Then, including the libiopmp.h into your program and using the APIs to operate your IOPMP:

```
#include "libiopmp.h"
```

1.4 Documentation

Please check the `libiopmp.pdf` under `docs` folder.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 iopmp_entry Struct Reference

`#include <libiopmp.h>`

Data Fields

- union {
  struct {
  uint32_t addrl
  uint32_t addrh
  }
  uint64_t addr
  };

- union {
  struct {
  uint32_t r: 1
  uint32_t w: 1
  uint32_t x: 1
  uint32_t a: 2
  uint32_t sire: 1
  uint32_t siwe: 1
  uint32_t sixe: 1
  uint32_t sere: 1
  uint32_t sewe: 1
  uint32_t sexe: 1
  uint32_t rsv: 21
  }
  uint32_t cfg
  };

- enum iopmp_prient_flags prient_flag
- uint64_t private_data

4.1.1 Detailed Description

Structure to represent an IOPMP entry, including the physical address of protected memory region, permission attributes, per-entry suppression settings, priority flags, private data, etc

4.1.2 Field Documentation

4.1.2.1 addrl

```
uint32_t addrl
```

The physical address[33:2] of memory region

4.1.2.2 addrh

```
uint32_t addrh
```

The physical address[65:34] of memory region

4.1.2.3 addr

```
uint64_t addr
```

The physical address[65:2] of protected memory region

4.1.2.4 [union]

```
union { ... }
```

Values of ENTRY_ADDR and ENTRY_ADDRH

4.1.2.5 r

```
uint32_t r
```

ENTRY_CFG.r

4.1.2.6 w

```
uint32_t w
```

ENTRY_CFG.w

4.1.2.7 x

```
uint32_t x
```

ENTRY_CFG.x

4.1.2.8 a

```
uint32_t a
```

ENTRY_CFG.a

4.1.2.9 sire

```
uint32_t sire
```

ENTRY_CFG.sire

4.1.2.10 siwe

```
uint32_t siwe
```

ENTRY_CFG.siwe

4.1.2.11 sixe

```
uint32_t sixe
```

ENTRY_CFG.sixe

4.1.2.12 sere

```
uint32_t sere
```

ENTRY_CFG.sere

4.1.2.13 sewe

```
uint32_t sewe
```

ENTRY_CFG.sewe

4.1.2.14 sexe

```
uint32_t sexe
```

ENTRY_CFG.sexe

4.1.2.15 rsv

```
uint32_t rsv
```

ENTRY_CFG.rsv

4.1.2.16 cfg

```
uint32_t cfg
```

ENTRY_CFG

4.1.2.17 [union]

```
union { ... }
```

Value of ENTRY_CFG

4.1.2.18 prient_flag

```
enum iopmp_prient_flags prient_flag
```

Flag to indicate this is priority or non-priority entry

4.1.2.19 private_data

```
uint64_t private_data
```

Additional 64-bit data that can be used in specific model.

For example, it can be used as SRCMD_PERM(H) in SRCMD_FMT=2, MDCFG_FMT=1 and HWCFG0.md_entry↩
_num=0 (K=1). In this configuration, each MD has exactly single entry. User can set SRCMD_PERM(H) and entry in single entry API call.

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

- libiopmp.h

4.2 iopmp_err_report Struct Reference

```
#include <libiopmp.h>
```

Data Fields

- uint64_t addr
- uint32_t rrid
- uint32_t eid
- enum iopmp_errinfo_ttype ttype
- enum iopmp_errinfo_etype etype
- bool msi_werr
- bool svc

4.2.1 Detailed Description

Structure represents an IOPMP error report

4.2.2 Field Documentation

4.2.2.1 addr

```
uint64_t addr
```

Errored address[65:2]

4.2.2.2 rrid

```
uint32_t rrid
```

Errored RRID

4.2.2.3 eid

```
uint32_t eid
```

Indicates the index pointing to the entry that catches the violation

4.2.2.4 ttype

```
enum iopmp_errinfo_ttype ttype
```

Indicated the transaction type of the first captured violation

4.2.2.5 etype

```
enum iopmp_errinfo_etype etype
```

Indicated the type of violation

4.2.2.6 msi_werr

```
bool msi_werr
```

Indicate the write access to trigger an IOPMP originated MSI failed

4.2.2.7 svc

```
bool svc
```

Indicate there is a subsequent violation caught in ERR_MFR

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

- libiopmp.h

4.3 iopmp_instance Struct Reference

```
#include <libiopmp.h>
```

Data Fields

- uintptr_t addr
- uint32_t granularity
- uint64_t entry_addr_bits
- struct iopmp_operations_generic * ops_generic
- struct iopmp_operations_specific * ops_specific
- struct iopmp_operations_sps * ops_sps
- uintptr_t addr_entry_array
- uint32_t vendor
- uint32_t impid
- uint16_t rrid_num
- uint16_t entry_num
- uint16_t prio_entry_num
- uint16_t rrid_transl
- uint8_t specver
- uint8_t md_num
- uint8_t md_entry_num
- uint8_t mdlck_lock
- uint64_t mdlck_md
- uint8_t mdcfglck_lock
- uint8_t mdcfglck_f
- uint8_t entrylck_lock
- uint16_t entrylck_f
- uint64_t msiaddr64
- uint16_t msidata
- struct {
  unsigned int init: 1
  unsigned int mdcfg_fmt: 2
  unsigned int srcmd_fmt: 2
  unsigned int tor_en: 1
  unsigned int sps_en: 1
  unsigned int user_cfg_en: 1
  unsigned int prient_prog: 1
  unsigned int rrid_transl_en: 1
  unsigned int rrid_transl_prog: 1
  unsigned int chk_x: 1
  unsigned int no_x: 1

```
    unsigned int no_w: 1
    unsigned int stall_en: 1
    unsigned int peis: 1
    unsigned int pees: 1
    unsigned int mfr_en: 1
    unsigned int addrh_en: 1
    unsigned int enable: 1
    unsigned int err_cfg_lock: 1
    unsigned int intr_enable: 1
    unsigned int err_resp_suppress: 1
    unsigned int msi_en: 1
    unsigned int stall_violation_en: 1
    unsigned int support_stall_by_rrid: 1
    unsigned int support_stall_by_md: 1
    unsigned int is_stalling: 1
};
```

4.3.1 Detailed Description

Structure for an IOPMP instance, including base address, operations, configurations, etc

4.3.2 Field Documentation

4.3.2.1 addr

```
uintptr_t addr
```

Base MMIO physical address of IOPMP

4.3.2.2 granularity

```
uint32_t granularity
```

PMP granularity

4.3.2.3 entry_addr_bits

```
uint64_t entry_addr_bits
```

Implemented bits of ENTRY_ADDR(H)

4.3.2.4 ops_generic

```
struct iopmp_operations_generic* ops_generic
```

Generic operations for all models

4.3.2.5 ops_specific

```
struct iopmp_operations_specific* ops_specific
```

Operations for specific model

4.3.2.6 ops_sps

```
struct iopmp_operations_sps* ops_sps
```

Operations for model supports SPS extension

4.3.2.7 addr_entry_array

```
uintptr_t addr_entry_array
```

Base MMIO physical address of IOPMP entries

4.3.2.8 vendor

```
uint32_t vendor
```

The JEDEC manufacturer ID

4.3.2.9 impid

```
uint32_t impid
```

The user-defined implementation ID

4.3.2.10 rrid_num

```
uint16_t rrid_num
```

Indicate the supported number of RRID in the instance

4.3.2.11 entry_num

```
uint16_t entry_num
```

Indicate the supported number of entries in the instance

4.3.2.12 prio_entry_num

```
uint16_t prio_entry_num
```

Indicate the number of entries matched with priority

4.3.2.13 rrid_transl

```
uint16_t rrid_transl
```

The RRID tagged to outgoing transactions

4.3.2.14 specver

```
uint8_t specver
```

The specification version

4.3.2.15 md_num

```
uint8_t md_num
```

Indicate the supported number of MD in the instance

4.3.2.16 md_entry_num

```
uint8_t md_entry_num
```

When mdcfg_fmt={1,2}, indicate each MD has (md_entry_num+1) entries

4.3.2.17 mdlck_lock

```
uint8_t mdlck_lock
```

Cache of MDLCK.l

4.3.2.18 mdlck_md

```
uint64_t mdlck_md
```

Cache of MDLCK.md

4.3.2.19 mdcfglck_lock

```
uint8_t mdcfglck_lock
```

Cache of MDCFGLCK.l

4.3.2.20 mdcfglck_f

```
uint8_t mdcfglck_f
```

Cache of MDCFGLCK.f

4.3.2.21 entrylck_lock

```
uint8_t entrylck_lock
```

Cache of ENTRYLCK.l

4.3.2.22 entrylck_f

```
uint16_t entrylck_f
```

Cache of ENTRYLCK.f

4.3.2.23 msiaddr64

```
uint64_t msiaddr64
```

Cache of {ERR_MSIADDRH, ERR_MSIADDR}. If HWCFG0.addrh_en=0, this member contains bits 33 to 2 of the MSI address. If HWCFG0.addrh_en=1, this member contains bits 63 to 0 of the MSI address

4.3.2.24 msidata

```
uint16_t msidata
```

Cache of ERR_CFG.msidata

4.3.2.25 init

```
unsigned int init
```

Flag to indicate the IOPMP instance has been initialized

4.3.2.26 mdcfg_fmt

```
unsigned int mdcfg_fmt
```

Flag to indicate the MDCFG format

4.3.2.27 srcmd_fmt

```
unsigned int srcmd_fmt
```

Flag to indicate the SRCMD format

4.3.2.28 tor_en

```
unsigned int tor_en
```

Flag to indicate if TOR is supported

4.3.2.29 sps_en

```
unsigned int sps_en
```

Flag to indicate SPS(secondary permission settings) is supported

4.3.2.30 user_cfg_en

```
unsigned int user_cfg_en
```

Flag to indicate if user customized attributes is supported

4.3.2.31 prient_prog

```
unsigned int prient_prog
```

Flag to indicates if HWCFG2.prio_entry is programmable

4.3.2.32 rrid_transl_en

```
unsigned int rrid_transl_en
```

Flag to indicate the if tagging a new RRID on the initiator port is supported

4.3.2.33 rrid_transl_prog

```
unsigned int rrid_transl_prog
```

Flag to indicate if the field HWCFG2.rrid_transl is programmable

4.3.2.34 chk_x

```
unsigned int chk_x
```

Flag to indicate if the IOPMP implements the check of an instruction fetch

4.3.2.35 no_x

```
unsigned int no_x
```

Flag to indicate for chk_x=1, the IOPMP with no_x=1 always fails on an instruction fetch

4.3.2.36 no_w

```
unsigned int no_w
```

Flag to indicate if the IOPMP always fails write accesses considered as no rule matched

4.3.2.37 stall_en

```
unsigned int stall_en
```

Flag to indicate if the IOPMP implements stall-related features

4.3.2.38 peis

```
unsigned int peis
```

Flag to indicate if the IOPMP implements interrupt suppression per entry

4.3.2.39 pees

```
unsigned int pees
```

Flag to indicate if the IOPMP implements error suppression per entry

4.3.2.40 mfr_en

```
unsigned int mfr_en
```

Flag to indicate if the IOPMP implements MFR(Multi-Faults Record) extension

4.3.2.41 addrh_en

```
unsigned int addrh_en
```

Flag to indicate if registers ENTRY_ADDRH(i) and ERR_MSIADDRH (if ERR_CFG.msi_en = 1) are available

4.3.2.42 enable

```
unsigned int enable
```

Indicate if the IOPMP checks transactions

4.3.2.43 err_cfg_lock

```
unsigned int err_cfg_lock
```

Lock fields to ERR_CFG register

4.3.2.44 intr_enable

```
unsigned int intr_enable
```

Enable the global interrupt of the IOPMP

4.3.2.45 err_resp_suppress

```
unsigned int err_resp_suppress
```

Suppress the global error responses of the IOPMP

4.3.2.46 msi_en

```
unsigned int msi_en
```

Flag to indicate whether the IOPMP triggers MSI

4.3.2.47 stall_violation_en

```
unsigned int stall_violation_en
```

Flag to indicate whether the IOPMP faults stalled transactions

4.3.2.48 support_stall_by_rrid

```
unsigned int support_stall_by_rrid
```

Flag to indicate if stall by RRID is supported

4.3.2.49 support_stall_by_md

```
unsigned int support_stall_by_md
```

Flag to indicate if stall by MD is supported

4.3.2.50 is_stalling

```
unsigned int is_stalling
```

Flag to indicate if IOPMP is stalling some transactions

4.3.2.51 [struct]

```
struct { ... }
```

Flags

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

- libiopmp.h

4.4 iopmp_srcmd_perm_config Struct Reference

Configuration used in srcmd_fmt=2 to set SRCMD_PERM(H)

```
#include <libiopmp.h>
```

Data Fields

- uint64_t srcmd_perm_mask
- uint64_t srcmd_perm_val

4.4.1 Detailed Description

Configuration used in srcmd_fmt=2 to set SRCMD_PERM(H)

Note

User should call the following macros or APIs to update this structure:

- iopmp_set_srcmd_perm_cfg() to update single RRID
- iopmp_set_srcmd_perm_cfg_nocheck() to update single RRID
- IOPMP_SRCMD_PERM_CFG_SET_DIRECT() to directly set multiple RRIDs

4.4.2 Field Documentation

4.4.2.1 srcmd_perm_mask

```
uint64_t srcmd_perm_mask
```

Bit mask to indicate which RRIDs' permission bits should be configured. For example, if we are going to configure RRID(0)'s bits, the bit 0 and bit 1 of this member will be set to 1

4.4.2.2 srcmd_perm_val

```
uint64_t srcmd_perm_val
```

Bit mask to indicate the desired permissions for configured RRIDs. For example, if we are going to configure RRID(0)'s bits, the bit 0 indicates whether RRID(0) has read permission on this MD, while the bit 1 indicates whether RRID(0) has write permission on this MD

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

- libiopmp.h

Chapter 5

File Documentation

5.1 libiopmp.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
```

Data Structures

- struct iopmp_instance
- struct iopmp_entry
- struct iopmp_err_report
- struct iopmp_srcmd_perm_config

  Configuration used in srcmd_fmt=2 to set SRCMD_PERM(H)

Macros

- #define IOPMP_MAX_RRID_SRCMD_FMT_2 32
- #define IOPMP_SRCMD_PERM_R (1 << 0)
- #define IOPMP_SRCMD_PERM_W (1 << 1)
- #define IOPMP_SRCMD_PERM_MASK (IOPMP_SRCMD_PERM_W | IOPMP_SRCMD_PERM_R)
- #define IOPMP_SRCMD_PERM_CFG_SET_DIRECT(cfg, mask, val)

  Macro used to directly set members in struct iopmp_srcmd_perm_config.
- #define LIBIOPMP_VERSION_MAJOR 0
- #define LIBIOPMP_VERSION_MINOR 1
- #define LIBIOPMP_VERSION_EXTRA 0
- #define LIBIOPMP_VERSION_MAJOR_SHIFT 16
- #define LIBIOPMP_VERSION_MAJOR_MASK 0xffff
- #define LIBIOPMP_VERSION_MINOR_SHIFT 8
- #define LIBIOPMP_VERSION_MINOR_MASK 0xff
- #define LIBIOPMP_VERSION_EXTRA_SHIFT 0
- #define LIBIOPMP_VERSION_EXTRA_MASK 0xff
- #define LIBIOPMP_VERSION(__major, __minor, __extra)

  The macro to construct the IOPMP version number.

Typedefs

- typedef struct iopmp_instance IOPMP_t
- typedef struct iopmp_entry IOPMP_Entry_t
- typedef struct iopmp_err_report IOPMP_ERR_REPORT_t
- typedef struct iopmp_srcmd_perm_config IOPMP_SRCMD_PERM_CFG_t

Enumerations

- enum iopmp_prient_flags
- enum iopmp_errinfo_ttype
- enum iopmp_errinfo_etype
- enum iopmp_impid
- enum iopmp_srcmd_fmt
- enum iopmp_mdcfg_fmt
- enum iopmp_model
- enum iopmp_rridscp_op
- enum iopmp_rridscp_stat
- enum iopmp_entry_flags
- enum iopmp_error

Functions

- int libiopmp_major_version (void)

  Get major version of libiopmp.
- int libiopmp_minor_version (void)

  Get minor version of libiopmp.
- int libiopmp_extra_version (void)

  Get extra version of libiopmp.
- bool libiopmp_check_version (int major, int minor, int extra)

  Check given version with libiopmp.
- static bool iopmp_is_initialized (IOPMP_t *iopmp)

  Check if the IOPMP has been initialized by libiopmp.
- static uintptr_t iopmp_get_base_addr (IOPMP_t *iopmp)

  Get the base physical address of the IOPMP.
- static uintptr_t iopmp_get_base_addr_entry_array (IOPMP_t *iopmp)

  Get the base physical address of the IOPMP entry array.
- static uint32_t iopmp_get_granularity (IOPMP_t *iopmp)

  Get the granularity of the IOPMP.
- static enum iopmp_mdcfg_fmt iopmp_get_mdcfg_fmt (IOPMP_t *iopmp)

  Get HWCFG0.mdcfg_fmt of the IOPMP.
- static enum iopmp_srcmd_fmt iopmp_get_srcmd_fmt (IOPMP_t *iopmp)

  Get HWCFG0.srcmd_fmt of the IOPMP.
- static bool iopmp_get_support_tor (IOPMP_t *iopmp)

  Get HWCFG0.tor_en of the IOPMP.
- static bool iopmp_get_support_sps (IOPMP_t *iopmp)

  Check if the IOPMP supports SPS extension.
- static bool iopmp_get_support_user_entry_cfg (IOPMP_t *iopmp)

  Check if the IOPMP supports user customized attributes.
- static bool iopmp_get_support_programmable_prio_entry (IOPMP_t *iopmp)

  Check if HWCFG2.prio_entry is programmable.

- static bool iopmp_get_support_rrid_transl (IOPMP_t *iopmp)
  Check if tagging a new RRID on the initiator port is supported.
- static bool iopmp_get_rrid_transl_prog (IOPMP_t *iopmp)
  Check if HWCFG2.rrid_transl is programmable.
- static uint16_t iopmp_get_rrid_transl (IOPMP_t *iopmp)
  Get the value of HWCFG2.rrid_transl.
- static bool iopmp_get_support_chk_x (IOPMP_t *iopmp)
  Check if the IOPMP implements the check of an instruction fetch.
- static bool iopmp_get_no_x (IOPMP_t *iopmp)
  Check if the IOPMP always fails on an instruction fetch.
- static bool iopmp_get_no_w (IOPMP_t *iopmp)
  Check if the IOPMP always fails on write accesses considered as as no rule matched.
- static bool iopmp_get_support_stall (IOPMP_t *iopmp)
  Check if the IOPMP implements stall-related features.
- static bool iopmp_get_support_peis (IOPMP_t *iopmp)
  Check if the IOPMP implements interrupt suppression per entry.
- static bool iopmp_get_support_pees (IOPMP_t *iopmp)
  Check if the IOPMP implements the error suppression per entry.
- static bool iopmp_get_support_mfr (IOPMP_t *iopmp)
  Check if the IOPMP implements the Multi-Faults Record Extension.
- static uint32_t iopmp_get_md_num (IOPMP_t *iopmp)
  Get the supported number of MD in the IOPMP instance.
- static uint32_t iopmp_get_addrh_en (IOPMP_t *iopmp)
  Check if ENTRY_ADDRH(i) and ERR_MSIADDRH (if ERR_CFG.msi_en = 1) are available.
- static bool iopmp_get_enable (IOPMP_t *iopmp)
  Check if the IOPMP checks transactions.
- static uint32_t iopmp_get_rrid_num (IOPMP_t *iopmp)
  Get the supported number of RRID in the IOPMP instance.
- static uint32_t iopmp_get_entry_num (IOPMP_t *iopmp)
  Get the supported number of entries in the IOPMP instance.
- static uint16_t iopmp_get_prio_entry_num (IOPMP_t *iopmp)
  Get the number of entries matched with priority.
- static bool iopmp_get_support_stall_by_md (IOPMP_t *iopmp)
  Check if the IOPMP implements stall-related features of MDSTALL(H)
- static bool iopmp_get_support_stall_by_rrid (IOPMP_t *iopmp)
  Check if the IOPMP implements stall-related features of RRIDSCP.
- static bool iopmp_is_err_cfg_locked (IOPMP_t *iopmp)
  Check if the ERR_CFG register has been locked.
- static bool iopmp_get_global_intr (IOPMP_t *iopmp)
  Check if the interrupt of the IOPMP rule violation has been enabled.
- static bool iopmp_get_global_err_resp (IOPMP_t *iopmp)
  Check if the IOPMP suppresses error response on a rule violation.
- static bool iopmp_get_stall_violation_en (IOPMP_t *iopmp)
  Check if the IOPMP faults stalled transactions.
- static bool iopmp_get_msi_en (IOPMP_t *iopmp)
  Check if the IOPMP triggers interrupt by MSI.
- static bool iopmp_is_mdlck_locked (IOPMP_t *iopmp)
  Check if MDLCK register has been locked.
- static bool iopmp_is_entrylck_locked (IOPMP_t *iopmp)
  Check if ENTRYLCK register has been locked.
- static uint32_t iopmp_get_locked_entry_num (IOPMP_t *iopmp)

Get the number of locked IOPMP entries.

- static uint64_t iopmp_err_report_get_addr (IOPMP_ERR_REPORT_t *err_report)

  Get the errored address from the error report.
- static uint32_t iopmp_err_report_get_rrid (IOPMP_ERR_REPORT_t *err_report)

  Get the errored RRID from the error report.
- static uint32_t iopmp_err_report_get_eid (IOPMP_ERR_REPORT_t *err_report)

  Get the index pointing to the entry that catches the violation from the error report.
- static bool iopmp_err_report_is_no_hit (IOPMP_ERR_REPORT_t *err_report)

  Check if the type of violation is "not hit any rule" in the error report.
- static bool iopmp_err_report_is_part_hit (IOPMP_ERR_REPORT_t *err_report)

  Check if the type of violation is "partial hit on a priority rule" in the error report.
- static enum iopmp_errinfo_ttype iopmp_err_report_get_ttype (IOPMP_ERR_REPORT_t *err_report)

  Get the transaction type from the error report.
- static bool iopmp_err_report_get_msi_werr (IOPMP_ERR_REPORT_t *err_report)

  Check if the write access to trigger an IOPMP originated MSI has failed in the error report.
- static enum iopmp_errinfo_etype iopmp_err_report_get_etype (IOPMP_ERR_REPORT_t *err_report)

  Get the type of violation from the error report.
- static bool iopmp_err_report_get_svc (IOPMP_ERR_REPORT_t *err_report)

  Get ERR_INFO.svc from the error report.
- static uint64_t iopmp_entry_get_addr (IOPMP_Entry_t *entry)

  Get the physical address[65:2] of protected memory region from the IOPMP entry structure.
- static uint32_t iopmp_entry_get_cfg (IOPMP_Entry_t *entry)

  Get the permissions and attributes of protected memory region from the IOPMP entry structure.
- enum iopmp_error iopmp_init (IOPMP_t *iopmp, uintptr_t addr, uint8_t srcmd_fmt, uint8_t mdcfg_fmt, uint32_t impid)

  Initialize the IOPMP instance. Read the intial states and prepare the IOPMP driver operations.
- enum iopmp_error iopmp_get_vendor_id (IOPMP_t *iopmp, uint32_t *vendor)

  Get the vendor ID of the IOPMP.
- enum iopmp_error iopmp_get_specver (IOPMP_t *iopmp, uint32_t *specver)

  Get the specification version of the IOPMP.
- enum iopmp_error iopmp_get_impid (IOPMP_t *iopmp, uint32_t *impid)

  Get the implementation ID of the IOPMP.
- enum iopmp_error iopmp_lock_prio_entry_num (IOPMP_t *iopmp)

  Lock number of priority entry if the IOPMP HWCFG0.prient_prog=1.
- enum iopmp_error iopmp_lock_rrid_transl (IOPMP_t *iopmp)

  Lock the RRID tagged to outgoing transactions if the IOPMP HWCFG0.rrid_transl_prog=1.
- enum iopmp_error iopmp_set_enable (IOPMP_t *iopmp)

  Enable the IOPMP checker.
- enum iopmp_error iopmp_set_prio_entry_num (IOPMP_t *iopmp, uint16_t *num_entry)

  Set the number of entries matched with priority.
- enum iopmp_error iopmp_set_rrid_transl (IOPMP_t *iopmp, uint16_t *rrid_transl)

  Set the RRID tagged to outgoing transactions.
- enum iopmp_error iopmp_stall_transactions_by_mds (IOPMP_t *iopmp, uint64_t *mds, bool exempt, bool polling)

  Stall the transactions related to given MD bitmap and poll the stall status until stalling takes effect if necessary.
- enum iopmp_error iopmp_resume_transactions (IOPMP_t *iopmp, bool polling)

  Resume the stalled transactions previously stalled, and poll the resume status until resuming takes effect if necessary.
- enum iopmp_error iopmp_transactions_are_stalled (IOPMP_t *iopmp, bool polling)

  Check if the requested stall transactions takes effect.
- enum iopmp_error iopmp_transactions_are_resumed (IOPMP_t *iopmp, bool polling)

  Check if the requested resume transactions takes effect.

- enum iopmp_error iopmp_stall_cherry_pick_rrid (IOPMP_t *iopmp, uint32_t *rrid, bool select, enum iopmp_rridscp_stat *stat)

  Select or deselect the transactions with specific RRIDs to stall.
- enum iopmp_error iopmp_query_stall_stat_by_rrid (IOPMP_t *iopmp, uint32_t *rrid, enum iopmp_rridscp_stat *stat)

  Query the stall status of given RRID.
- enum iopmp_error iopmp_get_locked_md (IOPMP_t *iopmp, uint64_t *mds, bool *mdlck_lock)

  Get locked MDs and MDLCK.l.
- enum iopmp_error iopmp_lock_md (IOPMP_t *iopmp, uint64_t *mds, bool mdlck_lock)

  Lock MDs.
- enum iopmp_error iopmp_lock_mdcfg (IOPMP_t *iopmp, uint32_t *md_num, bool lock)

  Lock MDCFG(0) ∼ MDCFG(md_num - 1)
- enum iopmp_error iopmp_is_mdcfglck_locked (IOPMP_t *iopmp, bool *locked)

  Check if MDCFGLCK was locked.
- enum iopmp_error iopmp_get_locked_mdcfg_num (IOPMP_t *iopmp, uint32_t *md_num)

  Get number of MDs whose MDCFG were locked by MDCFGLCK.
- enum iopmp_error iopmp_lock_entries (IOPMP_t *iopmp, uint32_t *entry_num, bool lock)

  Lock ENTRY_ADDR[0 ∼ (entry_num-1)], ENTRY_ADDRH[0 ∼ (entry_num-1)], ENTRY_CFG[0 ∼ (entry_num-1)], and ENTRY_USER_CFG[0 ∼ (entry_num-1)].
- enum iopmp_error iopmp_lock_err_cfg (IOPMP_t *iopmp)

  Lock fields of ERR_CFG register.
- enum iopmp_error iopmp_set_global_intr (IOPMP_t *iopmp, bool enable)

  Enable/Disable global interrupt.
- enum iopmp_error iopmp_set_global_err_resp (IOPMP_t *iopmp, bool *suppress)

  Suppress/express global error responses.
- enum iopmp_error iopmp_set_msi_en (IOPMP_t *iopmp, bool *enable)

  Enable/disable IOPMP trigger message-signaled interrupts on errors.
- enum iopmp_error iopmp_get_msi_addr (IOPMP_t *iopmp, uint64_t *msiaddr64)

  Get the address to trigger message-signaled interrupts.
- enum iopmp_error iopmp_get_msi_data (IOPMP_t *iopmp, uint16_t *msidata)

  Get the data to trigger message-signaled interrupts.
- enum iopmp_error iopmp_set_msi_info (IOPMP_t *iopmp, uint64_t *msiaddr64, uint16_t *msidata)

  Set address and data of message-signaled interrupts.
- enum iopmp_error iopmp_get_and_clear_msi_werr (IOPMP_t *iopmp, bool *msi_werr)

  Check if there is an MSI write error and clear the flag.
- enum iopmp_error iopmp_set_stall_violation_en (IOPMP_t *iopmp, bool *enable)

  Enable or disable the IOPMP faults stalled transactions.
- enum iopmp_error iopmp_invalidate_error (IOPMP_t *iopmp)

  Invalidate the error record by clearing ERR_INFO.v bit.
- enum iopmp_error iopmp_capture_error (IOPMP_t *iopmp, IOPMP_ERR_REPORT_t *err_report, bool invalidate)

  Capture an IOPMP error information.
- enum iopmp_error iopmp_mfr_get_sv_window (IOPMP_t *iopmp, uint16_t *svi, uint16_t *svw)

  Get subsequent violation window, if IOPMP supports MFR extension.
- enum iopmp_error iopmp_lock_srcmd_table_fmt_0 (IOPMP_t *iopmp, uint32_t rrid)

  Lock SRCMD_EN(rrid), SRCMD_ENH(rrid), SRCMD_R(rrid), SRCMD_RH(rrid), SRCMD_W(rrid), and SRCMD_↩ WH(rrid) if any.
- enum iopmp_error iopmp_is_srcmd_table_fmt_0_locked (IOPMP_t *iopmp, uint32_t rrid, bool *locked)

  Check if SRCMD_EN(rrid), SRCMD_ENH(rrid), SRCMD_R(rrid), SRCMD_RH(rrid), SRCMD_W(rrid), and SRCMD↩ _WH(rrid) if any, have been locked.
- enum iopmp_error iopmp_lock_srcmd_table_fmt_2 (IOPMP_t *iopmp, uint32_t mdidx)

  Lock SRCMD_PERM(mdidx) and SRCMD_PERMH(mdidx)

- enum iopmp_error iopmp_is_srcmd_table_fmt_2_locked (IOPMP_t *iopmp, uint32_t mdidx, bool *locked)

  Check if SRCMD_PERM(mdidx) and SRCMD_PERMH(mdidx) have been locked.
- enum iopmp_error iopmp_get_rrid_md_association (IOPMP_t *iopmp, uint32_t rrid, uint64_t *mds, bool *lock)

  Get the associated MD bitmap and lock bit of given RRID.
- enum iopmp_error iopmp_set_rrid_md_association (IOPMP_t *iopmp, uint32_t rrid, uint64_t mds_set, uint64_t mds_clr, uint64_t *mds, bool lock)

  Associate/Disassociate the given RRID with the given MD bitmap.
- enum iopmp_error iopmp_set_md_permission (IOPMP_t *iopmp, uint32_t rrid, uint32_t mdidx, bool *r, bool *w)

  (srcmd_fmt=2 only) Set single RRID's r/w permissions to MD
- enum iopmp_error iopmp_set_md_permission_multi (IOPMP_t *iopmp, uint32_t mdidx, IOPMP_SRCMD_PERM_CFG_t *cfg)

  (srcmd_fmt=2 only) Set multiple RRID's r/w permissions to MD
- enum iopmp_error iopmp_set_srcmd_perm_cfg (IOPMP_SRCMD_PERM_CFG_t *cfg, uint32_t rrid, bool r, bool w)

  Helper function used to set struct iopmp_srcmd_perm_config.
- void iopmp_set_srcmd_perm_cfg_nocheck (IOPMP_SRCMD_PERM_CFG_t *cfg, uint32_t rrid, bool r, bool w)

  Helper function used to set struct iopmp_srcmd_perm_config. This is similar to iopmp_set_srcmd_perm_cfg() but there are no checks on cfg and RRID.
- enum iopmp_error iopmp_sps_set_rrid_md_read (IOPMP_t *iopmp, uint32_t rrid, uint64_t mds_set, uint64_t mds_clr, uint64_t *mds)

  (SPS only) Set RRID's read permission to MDs
- enum iopmp_error iopmp_sps_get_rrid_md_read (IOPMP_t *iopmp, uint32_t rrid, uint64_t *mds)

  (SPS only) Get RRID's read permission to MDs
- enum iopmp_error iopmp_sps_set_rrid_md_write (IOPMP_t *iopmp, uint32_t rrid, uint64_t mds_set, uint64_t mds_clr, uint64_t *mds)

  (SPS only) Set RRID's write permission to MDs
- enum iopmp_error iopmp_sps_get_rrid_md_write (IOPMP_t *iopmp, uint32_t rrid, uint64_t *mds)

  (SPS only) Get RRID's write permission to MDs
- enum iopmp_error iopmp_sps_set_rrid_md_rw (IOPMP_t *iopmp, uint32_t rrid, uint64_t mds_set_r, uint64_t mds_clr_r, uint64_t mds_set_w, uint64_t mds_clr_w, uint64_t *mds_r, uint64_t *mds_w)

  (SPS only) Set RRID's read and write permission to MDs
- enum iopmp_error iopmp_sps_get_rrid_md_rw (IOPMP_t *iopmp, uint32_t rrid, uint64_t *mds_r, uint64_t *mds_w)

  (SPS only) Get RRID's read and write permission to multiple MDs
- enum iopmp_error iopmp_get_md_entry_association (IOPMP_t *iopmp, uint32_t mdidx, uint32_t *entry_idx_start, uint32_t *num_entry)

  Get start index and number of the entries belong to MD[mdidx].
- enum iopmp_error iopmp_set_md_entry_association_multi (IOPMP_t *iopmp, uint32_t mdidx_start, uint32_t *num_entries, uint32_t md_num)

  Associate given entries with given multiple MDs.
- static enum iopmp_error iopmp_set_md_entry_association (IOPMP_t *iopmp, uint32_t mdidx, uint32_t *num_entry)

  Associate given entries with given MD(mdidx)
- enum iopmp_error iopmp_get_md_entry_num (IOPMP_t *iopmp, uint32_t *md_entry_num)

  Get value of HWCFG0.md_entry_num if IOPMP model is xxx-K.
- enum iopmp_error iopmp_set_md_entry_num (IOPMP_t *iopmp, uint32_t *md_entry_num)

  Program value of HWCFG0.md_entry_num.
- enum iopmp_error iopmp_encode_entry (IOPMP_t *iopmp, struct iopmp_entry *entries, uint32_t num_entry, uint64_t addr, uint64_t size, enum iopmp_entry_flags flags, uint64_t private_data)

  Encode IOPMP entry from given memory region and flags.

- enum iopmp_error iopmp_set_entries_to_md (IOPMP_t *iopmp, uint32_t mdidx, const struct iopmp_entry *entry_array, uint32_t idx_start, uint32_t num_entry)

  Set the entries belong to given MD to IOPMP.
- static enum iopmp_error iopmp_set_entry_to_md (IOPMP_t *iopmp, uint32_t mdidx, const struct iopmp_entry *entry, uint32_t idx)

  Set single entry belong to given MD to IOPMP.
- enum iopmp_error iopmp_get_entries_from_md (IOPMP_t *iopmp, uint32_t mdidx, struct iopmp_entry *entry_array, uint32_t idx_start, uint32_t num_entry)

  Get the entries belong to given MD from IOPMP.
- static enum iopmp_error iopmp_get_entry_from_md (IOPMP_t *iopmp, uint32_t mdidx, struct iopmp_entry *entry, uint32_t idx)

  Get single entry belong to given MD from IOPMP.
- enum iopmp_error iopmp_get_entries (IOPMP_t *iopmp, struct iopmp_entry *entry_array, uint32_t idx_start, uint32_t num_entry)

  Get the global entries from IOPMP.
- static enum iopmp_error iopmp_get_entry (IOPMP_t *iopmp, struct iopmp_entry *entry, uint32_t idx)

  Get single global entry from IOPMP.
- enum iopmp_error iopmp_set_entries (IOPMP_t *iopmp, const struct iopmp_entry *entry_array, uint32_t idx_start, uint32_t num_entry)

  Set the global entries into IOPMP.
- static enum iopmp_error iopmp_set_entry (IOPMP_t *iopmp, const struct iopmp_entry *entry, uint32_t idx)

  Set single global entry into IOPMP.
- enum iopmp_error iopmp_clear_entries_in_md (IOPMP_t *iopmp, uint32_t mdidx)

  Clear IOPMP entries in MD.
- enum iopmp_error iopmp_clear_entries (IOPMP_t *iopmp, uint32_t idx_start, uint32_t num_entry)

  Clear IOPMP entries.
- static enum iopmp_error iopmp_clear_entry (IOPMP_t *iopmp, uint32_t idx)

  Clear single global entry.
- enum iopmp_error iopmp_entries_get_belong_md (IOPMP_t *iopmp, uint32_t idx_start, uint32_t num_entry, uint64_t *mds)

  Get the MD bitmap that given index range of IOPMP entries belong to.

5.1.1 Macro Definition Documentation

5.1.1.1 IOPMP_MAX_RRID_SRCMD_FMT_2

```
#define IOPMP_MAX_RRID_SRCMD_FMT_2 32
```

Maximum supported RRID when srcmd_fmt=2

5.1.1.2 IOPMP_SRCMD_PERM_R

```
#define IOPMP_SRCMD_PERM_R (1 << 0)
```

Bit position of SRCMD_PERM.r for each RRID

5.1.1.3 IOPMP_SRCMD_PERM_W

```
#define IOPMP_SRCMD_PERM_W (1 << 1)
```

Bit position of SRCMD_PERM.w for each RRID

5.1.1.4 IOPMP_SRCMD_PERM_MASK

```
#define IOPMP_SRCMD_PERM_MASK (IOPMP_SRCMD_PERM_W | IOPMP_SRCMD_PERM_R)
```

Bit mask of SRCMD_PERM for each RRID

5.1.1.5 IOPMP_SRCMD_PERM_CFG_SET_DIRECT

```
#define IOPMP_SRCMD_PERM_CFG_SET_DIRECT(
            cfg,
            mask,
            val)
```

Value:

```
do {                                                \
    IOPMP_SRCMD_PERM_CFG_t *__cfg = (cfg);          \
    __cfg->srcmd_perm_mask = mask;                  \
    __cfg->srcmd_perm_val  = val;                   \
} while (0);
```

Macro used to directly set members in struct iopmp_srcmd_perm_config.

Parameters

in	*cfg*	pointer to struct iopmp_srcmd_perm_config
in	*mask*	Desired value of srcmd_perm_mask
in	*val*	Desired value of srcmd_perm_val

5.1.1.6 LIBIOPMP_VERSION_MAJOR

```
#define LIBIOPMP_VERSION_MAJOR 0
```

Major version of libiopmp release version

5.1.1.7 LIBIOPMP_VERSION_MINOR

```
#define LIBIOPMP_VERSION_MINOR 1
```

Minor version of libiopmp release version

5.1.1.8 LIBIOPMP_VERSION_EXTRA

```
#define LIBIOPMP_VERSION_EXTRA 0
```

Extra version of libiopmp release version

5.1.1.9 LIBIOPMP_VERSION_MAJOR_SHIFT

```
#define LIBIOPMP_VERSION_MAJOR_SHIFT 16
```

The bit position of the major version encoded in the IOPMP version number

5.1.1.10 LIBIOPMP_VERSION_MAJOR_MASK

```
#define LIBIOPMP_VERSION_MAJOR_MASK 0xffff
```

The bit mask of the major version encoded in the IOPMP version number

5.1.1.11 LIBIOPMP_VERSION_MINOR_SHIFT

```
#define LIBIOPMP_VERSION_MINOR_SHIFT 8
```

The bit position of the minor version encoded in the IOPMP version number

5.1.1.12 LIBIOPMP_VERSION_MINOR_MASK

```
#define LIBIOPMP_VERSION_MINOR_MASK 0xff
```

The bit mask of the minor version encoded in the IOPMP version number

5.1.1.13 LIBIOPMP_VERSION_EXTRA_SHIFT

```
#define LIBIOPMP_VERSION_EXTRA_SHIFT 0
```

The bit position of the extra version encoded in the IOPMP version number

5.1.1.14 LIBIOPMP_VERSION_EXTRA_MASK

```
#define LIBIOPMP_VERSION_EXTRA_MASK 0xff
```

The bit mask of the extra version encoded in the IOPMP version number

5.1.1.15 LIBIOPMP_VERSION

```
#define LIBIOPMP_VERSION(
              __major,
              __minor,
              __extra)
```

Value:

```
((((__major) & LIBIOPMP_VERSION_MAJOR_MASK) « LIBIOPMP_VERSION_MAJOR_SHIFT) | \
 (((__minor) & LIBIOPMP_VERSION_MINOR_MASK) « LIBIOPMP_VERSION_MINOR_SHIFT) | \
 (((__extra) & LIBIOPMP_VERSION_EXTRA_MASK) « LIBIOPMP_VERSION_EXTRA_SHIFT))
```

The macro to construct the IOPMP version number.

Parameters

in	*__major*	The major version
in	*__minor*	The minor version
in	*__extra*	The extra version

Returns

IOPMP version number

5.1.2 Typedef Documentation

5.1.2.1 IOPMP_t

```
typedef struct iopmp_instance IOPMP_t
```

5.1.2.2 IOPMP_Entry_t

```
typedef struct iopmp_entry IOPMP_Entry_t
```

5.1.2.3 IOPMP_ERR_REPORT_t

```
typedef struct iopmp_err_report IOPMP_ERR_REPORT_t
```

5.1.2.4 IOPMP_SRCMD_PERM_CFG_t

```
typedef struct iopmp_srcmd_perm_config IOPMP_SRCMD_PERM_CFG_t
```

5.1.3 Enumeration Type Documentation

5.1.3.1 iopmp_prient_flags

```
enum iopmp_prient_flags
```

Flags to indicate an entry must be priority entry or not. Some APIs which writing the entries into IOPMP will check this flag

- 0b00: ignore check
- 0b01: must be priority entry
- 0b10: must be non-priority entry

Enumerator

IOPMP_PRIENT_ANY	0	User sets this flag to indicate the entry's priority doesn't matter
IOPMP_PRIENT_PRIORITY	(1 << 0)	User sets this flag to indicate the entry must be priority entry
IOPMP_PRIENT_NON_PRIORITY	(1 << 1)	User sets this flag to indicate the entry must be non-priority entry

5.1.3.2 iopmp_errinfo_ttype

`enum iopmp_errinfo_ttype`

Indicated the transaction type of the first captured violation

Enumerator

IOPMP_ERRINFO_TTYPE_RSVD	0x00	Reserved
IOPMP_ERRINFO_TTYPE_READ	0x01	Read access
IOPMP_ERRINFO_TTYPE_WRITE	0x02	Write access/AMO
IOPMP_ERRINFO_TTYPE_INST_FETCH	0x03	Instruction fetch

5.1.3.3 iopmp_errinfo_etype

`enum iopmp_errinfo_etype`

Indicated the type of violation

Enumerator

IOPMP_ERRINFO_ETYPE_NONE	0x00	No error
IOPMP_ERRINFO_ETYPE_READ	0x01	Illegal read access
IOPMP_ERRINFO_ETYPE_WRITE	0x02	Illegal write access/AMO
IOPMP_ERRINFO_ETYPE_INST_FETCH	0x03	Illegal instruction fetch
IOPMP_ERRINFO_ETYPE_PART_HIT	0x04	Partial hit on a priority rule
IOPMP_ERRINFO_ETYPE_NOT_HIT	0x05	Not hit any rule
IOPMP_ERRINFO_ETYPE_UNKNOWN_RRID	0x06	Unknown RRID
IOPMP_ERRINFO_ETYPE_STALL	0x07	Error due to a stalled transaction
IOPMP_ERRINFO_ETYPE_RESERVED_0	0x08	N/A, reserved for future
IOPMP_ERRINFO_ETYPE_RESERVED_1	0x09	N/A, reserved for future
IOPMP_ERRINFO_ETYPE_RESERVED_2	0x0A	N/A, reserved for future
IOPMP_ERRINFO_ETYPE_RESERVED_3	0x0B	N/A, reserved for future
IOPMP_ERRINFO_ETYPE_RESERVED_4	0x0C	N/A, reserved for future
IOPMP_ERRINFO_ETYPE_RESERVED_5	0x0D	N/A, reserved for future
IOPMP_ERRINFO_ETYPE_USER_DEF_0	0x0E	User-defined error
IOPMP_ERRINFO_ETYPE_USER_DEF_1	0x0F	User-defined error

5.1.3.4 iopmp_impid

`enum iopmp_impid`

Enumerate implementation ID of IOPMP

Enumerator

IOPMP_IMPID_NOT_SPECIFIED	0xFFFFFFFF	The implementation ID of IOPMP is not specified

5.1.3.5 iopmp_srcmd_fmt

`enum iopmp_srcmd_fmt`

Enumerate the SRCMD table format

Enumerator

IOPMP_SRCMD_FMT_0	Format 0. SRCMD_EN(s) and SRCMD_ENH(s) are available
IOPMP_SRCMD_FMT_1	Format 1. No SRCMD Table
IOPMP_SRCMD_FMT_2	Format 2. SRCMD_PERM(m) and SRCMD_PERMH(m) are available
IOPMP_SRCMD_FMT_RESERVED	Reserved
IOPMP_SRCMD_FMT_MAX	Maximum number of SRCMD table formats

5.1.3.6 iopmp_mdcfg_fmt

`enum iopmp_mdcfg_fmt`

Enumerate the MDCFG table format

Enumerator

IOPMP_MDCFG_FMT_0	Format 0. MDCFG Table is implemented
IOPMP_MDCFG_FMT_1	Format 1. No MDCFG Table. HWCFG.md_entry_num is fixed
IOPMP_MDCFG_FMT_2	Format 2. No MDCFG Table. HWCFG.md_entry_num is programmable
IOPMP_MDCFG_FMT_RESERVED	Reserved
IOPMP_MDCFG_FMT_MAX	Maximum number of MDCFG table formats

5.1.3.7 iopmp_model

`enum iopmp_model`

Enumerate well-defined IOPMP models

Enumerator

IOPMP_MODEL_FULL	0	srcmd_fmt = 0 and mdcfg_fmt = 0
IOPMP_MODEL_RAPID_K	1	srcmd_fmt = 0 and mdcfg_fmt = 1
IOPMP_MODEL_DYNAMIC_K	2	srcmd_fmt = 0 and mdcfg_fmt = 2
IOPMP_MODEL_RESERVED_3	3	srcmd_fmt = 0 and mdcfg_fmt = 3 (reserved)
IOPMP_MODEL_ISOLATION	4	srcmd_fmt = 1 and mdcfg_fmt = 0
IOPMP_MODEL_COMPACT_K	5	srcmd_fmt = 1 and mdcfg_fmt = 1
IOPMP_MODEL_6	6	srcmd_fmt = 1 and mdcfg_fmt = 2
IOPMP_MODEL_RESERVED_7	7	srcmd_fmt = 1 and mdcfg_fmt = 3 (reserved)
IOPMP_MODEL_8	8	srcmd_fmt = 2 and mdcfg_fmt = 0
IOPMP_MODEL_9	9	srcmd_fmt = 2 and mdcfg_fmt = 1
IOPMP_MODEL_RESERVED_10	10	srcmd_fmt = 2 and mdcfg_fmt = 2
IOPMP_MODEL_RESERVED_11	11	srcmd_fmt = 2 and mdcfg_fmt = 3 (reserved)
IOPMP_MODEL_RESERVED_12	12	srcmd_fmt = 3 and mdcfg_fmt = 0 (reserved)
IOPMP_MODEL_RESERVED_13	13	srcmd_fmt = 3 and mdcfg_fmt = 1 (reserved)
IOPMP_MODEL_RESERVED_14	14	srcmd_fmt = 3 and mdcfg_fmt = 2 (reserved)
IOPMP_MODEL_RESERVED_15	15	srcmd_fmt = 3 and mdcfg_fmt = 3 (reserved)

5.1.3.8 iopmp_rridscp_op

`enum iopmp_rridscp_op`

The operations of RRIDSCP.op field

Enumerator

IOPMP_RRIDSCP_OP_QUERY	0	Query
IOPMP_RRIDSCP_OP_STALL	1	Stall transactions associated with selected RRID
IOPMP_RRIDSCP_OP_DONT_STALL	2	Don't stall transactions associated with selected RRID
IOPMP_RRIDSCP_OP_RESERVED	3	Reserved

5.1.3.9 iopmp_rridscp_stat

`enum iopmp_rridscp_stat`

The states of RRIDSCP.stat field

Enumerator

IOPMP_RRIDSCP_STAT_NOT_IMPL	0	RRIDSCP is not implemented
IOPMP_RRIDSCP_STAT_STALLED	1	Transactions associated with selected RRID are stalled
IOPMP_RRIDSCP_STAT_NOT_STALLED	2	Transactions associated with selected RRID are not stalled
IOPMP_RRIDSCP_STAT_ERR_RRID	3	Unimplemented or unselectable RRID

5.1.3.10 iopmp_entry_flags

`enum iopmp_entry_flags`

The flags used when calling iopmp_encode_entry()

Enumerator

IOPMP_ENTRY_R	(1UL << 0)
IOPMP_ENTRY_W	(1UL << 1)
IOPMP_ENTRY_X	(1UL << 2)
IOPMP_ENTRY_RW	(IOPMP_ENTRY_R | IOPMP_ENTRY_W)
IOPMP_ENTRY_RX	(IOPMP_ENTRY_R | IOPMP_ENTRY_X)
IOPMP_ENTRY_RWX	(IOPMP_ENTRY_R | IOPMP_ENTRY_W | IOPMP_ENTRY_X)
IOPMP_ENTRY_A_OFF	(0UL << 3)
IOPMP_ENTRY_A_TOR	(1UL << 3)
IOPMP_ENTRY_A_NA4	(2UL << 3)
IOPMP_ENTRY_A_NAPOT	(3UL << 3)
IOPMP_ENTRY_A_MASK	(3UL << 3)
IOPMP_ENTRY_SIRE	(1UL << 5)
IOPMP_ENTRY_SIWE	(1UL << 6)
IOPMP_ENTRY_SIXE	(1UL << 7)
IOPMP_ENTRY_SIE_MASK	(7UL << 5)
IOPMP_ENTRY_SERE	(1UL << 8)
IOPMP_ENTRY_SEWE	(1UL << 9)
IOPMP_ENTRY_SEXE	(1UL << 10)
IOPMP_ENTRY_SEE_MASK	(7UL << 8)

IOPMP_ENTRY_FORCE_OFF	(1UL << 27)
IOPMP_ENTRY_FIRST_TOR	(1UL << 28)
IOPMP_ENTRY_FORCE_TOR	(1UL << 29)
IOPMP_ENTRY_PRIO	(1UL << 30)
IOPMP_ENTRY_NON_PRIO	(1UL << 31)
IOPMP_ENTRY_SW_FLAGS_MASK	(IOPMP_ENTRY_FORCE_OFF | IOPMP_ENTRY_FIRST_TOR | IOPMP_ENTRY_FORCE

5.1.3.11 iopmp_error

```
enum iopmp_error
```

The libiopmp API error Code

Enumerator

IOPMP_OK	0	Success
IOPMP_ERR_NOT_SUPPORTED	-1	The operation is not supported by this IOPMP
IOPMP_ERR_OUT_OF_BOUNDS	-2	The given index is out-of-bounds
IOPMP_ERR_REG_IS_LOCKED	-3	The register is locked
IOPMP_ERR_NOT_ALLOWED	-4	The operation is not allowed
IOPMP_ERR_NOT_EXIST	-5	The result does not exist
IOPMP_ERR_NOT_AVAILABLE	-6	The resource is not available
IOPMP_ERR_INVALID_PARAMETER	-7	The given parameter is invalid
IOPMP_ERR_INVALID_PRIORITY	-8	The given priority is invalid
IOPMP_ERR_ILLEGAL_VALUE	-9	The desired value written into WARL field does not match actual value

5.1.4 Function Documentation

5.1.4.1 libiopmp_major_version()

```
int libiopmp_major_version (
            void )
```

Get major version of libiopmp.

Returns

The major version of libiopmp

5.1.4.2 libiopmp_minor_version()

```
int libiopmp_minor_version (
            void )
```

Get minor version of libiopmp.

Returns

The minor version of libiopmp

5.1.4.3 libiopmp_extra_version()

```
int libiopmp_extra_version (
            void )
```

Get extra version of libiopmp.

Returns

The extra version of libiopmp

5.1.4.4 libiopmp_check_version()

```
bool libiopmp_check_version (
            int major,
            int minor,
            int extra)
```

Check given version with libiopmp.

Parameters

in	*major*	The major version
in	*minor*	The minor version
in	*extra*	The extra version

Return values

1	if given version is greater than version of libiopmp
0	if given version is less than or equal to version of libiopmp

5.1.4.5 iopmp_is_initialized()

```
bool iopmp_is_initialized (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP has been initialized by libiopmp.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if the IOPMP has been initialized by libiopmp
0	if the IOPMP hasn't been initialized by libiopmp

5.1.4.6 iopmp_get_base_addr()

```
uintptr_t iopmp_get_base_addr (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the base physical address of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

The base physical address of the IOPMP

5.1.4.7 iopmp_get_base_addr_entry_array()

```
uintptr_t iopmp_get_base_addr_entry_array (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the base physical address of the IOPMP entry array.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

The base physical address of the IOPMP entry array

5.1.4.8 iopmp_get_granularity()

```
uint32_t iopmp_get_granularity (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the granularity of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

The granularity of the IOPMP

5.1.4.9 iopmp_get_mdcfg_fmt()

```
enum iopmp_mdcfg_fmt iopmp_get_mdcfg_fmt (
            IOPMP_t * iopmp)  [inline], [static]
```

Get HWCFG0.mdcfg_fmt of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG0.mdcfg_fmt of the IOPMP

5.1.4.10 iopmp_get_srcmd_fmt()

```
enum iopmp_srcmd_fmt iopmp_get_srcmd_fmt (
            IOPMP_t * iopmp)  [inline], [static]
```

Get HWCFG0.srcmd_fmt of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG0.srcmd_fmt of the IOPMP

5.1.4.11 iopmp_get_support_tor()

```
bool iopmp_get_support_tor (
            IOPMP_t * iopmp)  [inline], [static]
```

Get HWCFG0.tor_en of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got

Return values

1	if HWCFG0.tor_en = 1
0	if HWCFG0.tor_en = 0

5.1.4.12 iopmp_get_support_sps()

```
bool iopmp_get_support_sps (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP supports SPS extension.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.sps_en = 1 and the SPS operations are implemented
0	if HWCFG0.sps_en = 0

5.1.4.13 iopmp_get_support_user_entry_cfg()

```
bool iopmp_get_support_user_entry_cfg (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP supports user customized attributes.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.user_cfg_en = 1
0	if HWCFG0.user_cfg_en = 0

5.1.4.14 iopmp_get_support_programmable_prio_entry()

```
bool iopmp_get_support_programmable_prio_entry (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if HWCFG2.prio_entry is programmable.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.prient_prog = 1
0	if HWCFG0.prient_prog = 0

5.1.4.15 iopmp_get_support_rrid_transl()

```
bool iopmp_get_support_rrid_transl (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if tagging a new RRID on the initiator port is supported.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.rrid_transl_en = 1
0	if HWCFG0.rrid_transl_en = 0

5.1.4.16 iopmp_get_rrid_transl_prog()

```
bool iopmp_get_rrid_transl_prog (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if HWCFG2.rrid_transl is programmable.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.rrid_transl_prog = 1
0	if HWCFG0.rrid_transl_prog = 0

5.1.4.17 iopmp_get_rrid_transl()

```
uint16_t iopmp_get_rrid_transl (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the value of HWCFG2.rrid_transl.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG2.rrid_transl

5.1.4.18 iopmp_get_support_chk_x()

```
bool iopmp_get_support_chk_x (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements the check of an instruction fetch.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.rrid_transl_prog = 1
0	if HWCFG0.rrid_transl_prog = 0

5.1.4.19 iopmp_get_no_x()

```
bool iopmp_get_no_x (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP always fails on an instruction fetch.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.no_x = 1
0	if HWCFG0.no_x = 0

5.1.4.20 iopmp_get_no_w()

```
bool iopmp_get_no_w (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP always fails on write accesses considered as as no rule matched.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.no_w = 1
0	if HWCFG0.no_w = 0

5.1.4.21 iopmp_get_support_stall()

```
bool iopmp_get_support_stall (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements stall-related features.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.stall_en = 1
0	if HWCFG0.stall_en = 0

5.1.4.22 iopmp_get_support_peis()

```
bool iopmp_get_support_peis (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements interrupt suppression per entry.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.peis = 1
0	if HWCFG0.peis = 0

5.1.4.23 iopmp_get_support_pees()

```
bool iopmp_get_support_pees (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements the error suppression per entry.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.pees = 1
0	if HWCFG0.pees = 0

5.1.4.24 iopmp_get_support_mfr()

```
bool iopmp_get_support_mfr (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements the Multi-Faults Record Extension.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.mfr_en = 1
0	if HWCFG0.mfr_en = 0

5.1.4.25 iopmp_get_md_num()

```
uint32_t iopmp_get_md_num (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the supported number of MD in the IOPMP instance.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG0.md_num

5.1.4.26 iopmp_get_addrh_en()

```
uint32_t iopmp_get_addrh_en (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if ENTRY_ADDRH(i) and ERR_MSIADDRH (if ERR_CFG.msi_en = 1) are available.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.addrh_en = 1
0	if HWCFG0.addrh_en = 0

5.1.4.27 iopmp_get_enable()

```
bool iopmp_get_enable (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP checks transactions.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if HWCFG0.enable = 1
0	if HWCFG0.enable = 0

5.1.4.28 iopmp_get_rrid_num()

```
uint32_t iopmp_get_rrid_num (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the supported number of RRID in the IOPMP instance.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG2.rrid_num

5.1.4.29 iopmp_get_entry_num()

```
uint32_t iopmp_get_entry_num (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the supported number of entries in the IOPMP instance.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG2.entry_num

5.1.4.30 iopmp_get_prio_entry_num()

```
uint16_t iopmp_get_prio_entry_num (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the number of entries matched with priority.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

HWCFG2.prio_entry

5.1.4.31 iopmp_get_support_stall_by_md()

```
bool iopmp_get_support_stall_by_md (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements stall-related features of MDSTALL(H)

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if MDSTALL(H) are implemented
0	if MDSTALL(H) are not implemented

5.1.4.32 iopmp_get_support_stall_by_rrid()

```
bool iopmp_get_support_stall_by_rrid (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP implements stall-related features of RRIDSCP.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if RRIDSCP is implemented
0	if RRIDSCP is not implemented

5.1.4.33 iopmp_is_err_cfg_locked()

```
bool iopmp_is_err_cfg_locked (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the ERR_CFG register has been locked.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if ERR_CFG.l = 1
0	if ERR_CFG.l = 0

5.1.4.34 iopmp_get_global_intr()

```
bool iopmp_get_global_intr (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the interrupt of the IOPMP rule violation has been enabled.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if ERR_CFG.ie = 1
0	if ERR_CFG.ie = 0

5.1.4.35 iopmp_get_global_err_resp()

```
bool iopmp_get_global_err_resp (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP suppresses error response on a rule violation.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if ERR_CFG.rs = 1
0	if ERR_CFG.rs = 0

5.1.4.36 iopmp_get_stall_violation_en()

```
bool iopmp_get_stall_violation_en (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP faults stalled transactions.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if ERR_CFG.stall_violation_en = 1
0	if ERR_CFG.stall_violation_en = 0

5.1.4.37 iopmp_get_msi_en()

```
bool iopmp_get_msi_en (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if the IOPMP triggers interrupt by MSI.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if ERR_CFG.msi_en = 1
0	if ERR_CFG.msi_en = 0

5.1.4.38 iopmp_is_mdlck_locked()

```
bool iopmp_is_mdlck_locked (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if MDLCK register has been locked.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if MDLCK.l = 1
0	if MDLCK.l = 0

5.1.4.39 iopmp_is_entrylck_locked()

```
bool iopmp_is_entrylck_locked (
            IOPMP_t * iopmp)  [inline], [static]
```

Check if ENTRYLCK register has been locked.

Parameters

in	*iopmp*	The IOPMP instance to be checked

Return values

1	if ENTRYLCK.l = 1
0	if ENTRYLCK.l = 0

5.1.4.40 iopmp_get_locked_entry_num()

```
uint32_t iopmp_get_locked_entry_num (
            IOPMP_t * iopmp)  [inline], [static]
```

Get the number of locked IOPMP entries.

Parameters

in	*iopmp*	The IOPMP instance to be got

Returns

ENTRYLCK.f

5.1.4.41 iopmp_err_report_get_addr()

```
uint64_t iopmp_err_report_get_addr (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Get the errored address from the error report.

Parameters

in	*err_report*	The pointer to the error report

Returns

Errored address[65:2]

5.1.4.42 iopmp_err_report_get_rrid()

```
uint32_t iopmp_err_report_get_rrid (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Get the errored RRID from the error report.

Parameters

in	*err_report*	The pointer to the error report

Returns

Errored RRID

5.1.4.43 iopmp_err_report_get_eid()

```
uint32_t iopmp_err_report_get_eid (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Get the index pointing to the entry that catches the violation from the error report.

Parameters

in	*err_report*	The pointer to the error report

Returns

The index pointing to the entry that catches the violation

5.1.4.44 iopmp_err_report_is_no_hit()

```
bool iopmp_err_report_is_no_hit (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Check if the type of violation is "not hit any rule" in the error report.

Parameters

in	*err_report*	The pointer to the error report

Return values

1	if the type of violation is "not hit any rule"
0	if the type of violation is not "not hit any rule"

5.1.4.45 iopmp_err_report_is_part_hit()

```
bool iopmp_err_report_is_part_hit (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Check if the type of violation is "partial hit on a priority rule" in the error report.

Parameters

in	*err_report*	The pointer to the error report

Return values

1	if the type of violation is "partial hit on a priority rule"
0	if the type of violation is not "partial hit on a priority rule"

5.1.4.46 iopmp_err_report_get_ttype()

```
enum iopmp_errinfo_ttype iopmp_err_report_get_ttype (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Get the transaction type from the error report.

Parameters

in	*err_report*	The pointer to the error report

Returns

The transaction type

5.1.4.47 iopmp_err_report_get_msi_werr()

```
bool iopmp_err_report_get_msi_werr (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Check if the write access to trigger an IOPMP originated MSI has failed in the error report.

Parameters

in	*err_report*	The pointer to the error report

Return values

1	if the write access to trigger an IOPMP originated MSI has failed
0	if the write access to trigger an IOPMP originated MSI hasn't failed

5.1.4.48 iopmp_err_report_get_etype()

```
enum iopmp_errinfo_etype iopmp_err_report_get_etype (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Get the type of violation from the error report.

Parameters

in	*err_report*	The pointer to the error report

Returns

The type of violation

5.1.4.49 iopmp_err_report_get_svc()

```
bool iopmp_err_report_get_svc (
            IOPMP_ERR_REPORT_t * err_report)  [inline], [static]
```

Get ERR_INFO.svc from the error report.

Parameters

in	*err_report*	The pointer to the error report

Return values

1	if there is a subsequent violation caught in ERR_MFR
0	if there is no subsequent violation

5.1.4.50 iopmp_entry_get_addr()

```
uint64_t iopmp_entry_get_addr (
            IOPMP_Entry_t * entry)  [inline], [static]
```

Get the physical address[65:2] of protected memory region from the IOPMP entry structure.

Parameters

in	*entry*	The pointer to the IOPMP entry structure

Returns

The physical address[65:2] of protected memory region

5.1.4.51 iopmp_entry_get_cfg()

```
uint32_t iopmp_entry_get_cfg (
            IOPMP_Entry_t * entry)  [inline], [static]
```

Get the permissions and attributes of protected memory region from the IOPMP entry structure.

Parameters

in	*entry*	The pointer to the IOPMP entry structure

Returns

The permissions and attributes of protected memory region

5.1.4.52 iopmp_init()

```
enum iopmp_error iopmp_init (
            IOPMP_t * iopmp,
            uintptr_t addr,
            uint8_t srcmd_fmt,
            uint8_t mdcfg_fmt,
            uint32_t impid)
```

Initialize the IOPMP instance. Read the intial states and prepare the IOPMP driver operations.

Parameters

in	*iopmp*	The IOPMP instance to be initialized
in	*addr*	The base memory-mapped address of the IOPMP
in	*srcmd_fmt*	The SRCMD_FMT of this IOPMP instance
in	*mdcfg_fmt*	The MDCFG_FMT of this IOPMP instance
in	*impid*	The implementation ID of this IOPMP instance

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if some features are not supported

5.1.4.53 iopmp_get_vendor_id()

```
enum iopmp_error iopmp_get_vendor_id (
            IOPMP_t * iopmp,
            uint32_t * vendor)
```

Get the vendor ID of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got
out	*vendor*	Pointer to integer to store vendor ID

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given `vendor` is NULL
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support get the vendor ID of IOPMP

5.1.4.54 iopmp_get_specver()

```
enum iopmp_error iopmp_get_specver (
            IOPMP_t * iopmp,
            uint32_t * specver)
```

Get the specification version of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got
out	*specver*	Pointer to integer to store specification version

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given specver is NULL
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support get the specification version of the IOPMP

5.1.4.55 iopmp_get_impid()

```
enum iopmp_error iopmp_get_impid (
            IOPMP_t * iopmp,
            uint32_t * impid)
```

Get the implementation ID of the IOPMP.

Parameters

in	*iopmp*	The IOPMP instance to be got
out	*impid*	Pointer to integer to store implementation ID

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given impid is NULL
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support get the implementation ID of the IOPMP

5.1.4.56 iopmp_lock_prio_entry_num()

```
enum iopmp_error iopmp_lock_prio_entry_num (
            IOPMP_t * iopmp)
```

Lock number of priority entry if the IOPMP HWCFG0.prient_prog=1.

Parameters

in	*iopmp*	The IOPMP instance

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support lock the number of entries matched with priority

5.1.4.57 iopmp_lock_rrid_transl()

```
enum iopmp_error iopmp_lock_rrid_transl (
            IOPMP_t * iopmp)
```

Lock the RRID tagged to outgoing transactions if the IOPMP HWCFG0.rrid_transl_prog=1.

Parameters

in	*iopmp*	The IOPMP instance

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support lock the RRID tagged to outgoing transactions

5.1.4.58 iopmp_set_enable()

```
enum iopmp_error iopmp_set_enable (
            IOPMP_t * iopmp)
```

Enable the IOPMP checker.

Parameters

in	*iopmp*	The IOPMP instance to be set

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support enable the checker

5.1.4.59 iopmp_set_prio_entry_num()

```
enum iopmp_error iopmp_set_prio_entry_num (
            IOPMP_t * iopmp,
            uint16_t * num_entry)
```

Set the number of entries matched with priority.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*num_entry*	Input the number of entries to be matched with priority. Output WARL value.

Return values

IOPMP_OK	if successes
IOPMP_ERR_REG_IS_LOCKED	if HWCFG0.prient_prog is 0
IOPMP_ERR_INVALID_PARAMETER	if num_entry is NULL
IOPMP_ERR_ILLEGAL_VALUE	if the written num_entry does not match the actual value. The actual value is output via

5.1.4.60 iopmp_set_rrid_transl()

```
enum iopmp_error iopmp_set_rrid_transl (
            IOPMP_t * iopmp,
            uint16_t * rrid_transl)
```

Set the RRID tagged to outgoing transactions.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*rrid_transl*	Input the RRID tagged to outgoing transactions. Output WARL value

Return values

IOPMP_OK	if successes
IOPMP_ERR_REG_IS_LOCKED	if HWCFG0.rrid_transl_prog is 0
IOPMP_ERR_INVALID_PARAMETER	if given `rrid_transl` is NULL
IOPMP_ERR_ILLEGAL_VALUE	if the written `rrid_transl` does not match the actual value. The actual value is output

5.1.4.61 iopmp_stall_transactions_by_mds()

```
enum iopmp_error iopmp_stall_transactions_by_mds (
            IOPMP_t * iopmp,
            uint64_t * mds,
            bool exempt,
            bool polling)
```

Stall the transactions related to given MD bitmap and poll the stall status until stalling takes effect if necessary.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*mds*	Input the MD bitmap to be stalled. Output WARL value
in	*exempt*	Stall transactions with exempt selected MDs
in	*polling*	Set true to poll the stall status until stalling takes effect

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support stall
IOPMP_ERR_ILLEGAL_VALUE	if the written `mds` does not match the actual value. The actual value is output via `mds`
IOPMP_ERR_NOT_ALLOWED	if MDSTALL has already been written and libiopmp expects user resumes the transactions firs

5.1.4.62 iopmp_resume_transactions()

```
enum iopmp_error iopmp_resume_transactions (
            IOPMP_t * iopmp,
            bool polling)
```

Resume the stalled transactions previously stalled, and poll the resume status until resuming takes effect if necessary.

Parameters

in	*iopmp*	The IOPMP instance to be resumed
in	*polling*	Set true to poll the resume status until resuming takes effect

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support stall or resuming of stall
IOPMP_ERR_ILLEGAL_VALUE	if the written `mds` does not match the actual value
IOPMP_ERR_NOT_ALLOWED	if there was no transactions being stalled

5.1.4.63 iopmp_transactions_are_stalled()

```
enum iopmp_error iopmp_transactions_are_stalled (
            IOPMP_t * iopmp,
            bool polling)
```

Check if the requested stall transactions takes effect.

Parameters

`in`	*iopmp*	The IOPMP instance to be checked
`in`	*polling*	Set true to poll the stall status until stalling takes effect

Return values

1	if the stall has taken effect
0	if the stall has not taken effect yet
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support stall
IOPMP_ERR_NOT_EXIST	if `iopmp` did not stall any transactions by iopmp_stall_transactions_by_mds()

5.1.4.64 iopmp_transactions_are_resumed()

```
enum iopmp_error iopmp_transactions_are_resumed (
            IOPMP_t * iopmp,
            bool polling)
```

Check if the requested resume transactions takes effect.

Parameters

`in`	*iopmp*	The IOPMP instance to be checked
`in`	*polling*	Set true to poll the resume status until resuming takes effect

Return values

1	if the resuming has taken effect
0	if the resuming has not taken effect yet
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support stall
IOPMP_ERR_NOT_EXIST	if `iopmp` did not resume any transactions by iopmp_resume_transactions()

5.1.4.65 iopmp_stall_cherry_pick_rrid()

```
enum iopmp_error iopmp_stall_cherry_pick_rrid (
            IOPMP_t * iopmp,
            uint32_t * rrid,
            bool select,
            enum iopmp_rridscp_stat * stat)
```

Select or deselect the transactions with specific RRIDs to stall.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*rrid*	Input the RRID to be stalled. Output WARL value
in	*select*	Set true select or false to deselect
out	*stat*	The pointer to store enum iopmp_rridscp_stat

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given rrid is NULL or invalid
IOPMP_ERR_OUT_OF_BOUNDS	if given rrid is out of bounds
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support stall by RRID
IOPMP_ERR_ILLEGAL_VALUE	if the written rrid does not match the actual value. The actual value is output via rrid

Note

Although this function returns IOPMP_OK, the caller must check stat to determine the state of the operation.

After RRIDSCP is written, the action to stall desired transactions may not take effect immediately in some implementations. To determine whether the action takes effect, one can call iopmp_transactions_are_stalled().

5.1.4.66 iopmp_query_stall_stat_by_rrid()

```
enum iopmp_error iopmp_query_stall_stat_by_rrid (
            IOPMP_t * iopmp,
            uint32_t * rrid,
            enum iopmp_rridscp_stat * stat)
```

Query the stall status of given RRID.

Parameters

in	*iopmp*	The IOPMP instance to be queried
in,out	*rrid*	Input the RRID to be queried. Output WARL value
out	*stat*	The pointer to store enum iopmp_rridscp_stat

Return values

Positive	value for stall status of rrid
IOPMP_ERR_OUT_OF_BOUNDS	if given rrid is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given rrid or stat is NULL
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support stall by RRID or querying of stall

Note

Although this function returns IOPMP_OK, the caller must check stat to determine the state of the operation.

After RRIDSCP is written, the action to stall desired transactions may not take effect immediately in some implementations. To determine whether the action takes effect, one can call iopmp_transactions_are_stalled().

5.1.4.67 iopmp_get_locked_md()

```
enum iopmp_error iopmp_get_locked_md (
            IOPMP_t * iopmp,
            uint64_t * mds,
            bool * mdlck_lock)
```

Get locked MDs and MDLCK.l.

Parameters

in	*iopmp*	The IOPMP instance to be set
out	*mds*	Pointer to integer to store locked MD bitmap
out	*mdlck_lock*	Pointer to integer to store MDLCK.l

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given `mds` or `mdlck_lock` is NULL

5.1.4.68 iopmp_lock_md()

```
enum iopmp_error iopmp_lock_md (
            IOPMP_t * iopmp,
            uint64_t * mds,
            bool mdlck_lock)
```

Lock MDs.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*mds*	Input the MD bitmap to be locked. Output WARL value
in	*mdlck_lock*	Set 1 to lock MDLCK and MDLCKH registers

Return values

IOPMP_OK	if successes or both `mds` and `mdlck_lock` are 0
IOPMP_ERR_INVALID_PARAMETER	if `mds` is NULL
IOPMP_ERR_REG_IS_LOCKED	if MDLCK and MDLCKH have already been locked
IOPMP_ERR_NOT_SUPPORTED	if `mds` is out-of-bounds
IOPMP_ERR_ILLEGAL_VALUE	if the written value does not match the actual value. The actual values are output in `mds`

Note

If MDLCK.l has already been set to 1, this API always expects `mdlck_lock` be 1.

5.1.4.69 iopmp_lock_mdcfg()

```
enum iopmp_error iopmp_lock_mdcfg (
            IOPMP_t * iopmp,
            uint32_t * md_num,
            bool lock)
```

Lock MDCFG(0) ∼ MDCFG(md_num - 1)

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*md_num*	Input the number of MD to be locked. Output WARL value
in	*lock*	Set 1 to lock MDCFGLCK register

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if `md_num` is NULL
IOPMP_ERR_OUT_OF_BOUNDS	if value of `md_num` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if MDCFGLCK has already been locked
IOPMP_ERR_NOT_ALLOWED	if `md_num` is not monotonically increased
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement MDCFG table format 0

5.1.4.70 iopmp_is_mdcfglck_locked()

```
enum iopmp_error iopmp_is_mdcfglck_locked (
            IOPMP_t * iopmp,
            bool * locked)
```

Check if MDCFGLCK was locked.

Parameters

`in`	*iopmp*	The IOPMP instance to be checked
`out`	*locked*	The pointer to an integer to store MDCFGLCK.l

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if `locked` is NULL
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement MDCFG table format 0

5.1.4.71 iopmp_get_locked_mdcfg_num()

```
enum iopmp_error iopmp_get_locked_mdcfg_num (
            IOPMP_t * iopmp,
            uint32_t * md_num)
```

Get number of MDs whose MDCFG were locked by MDCFGLCK.

Parameters

`in`	*iopmp*	The IOPMP instance to be checked
`out`	*md_num*	The pointer to an integer to store MDCFGLCK.f

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if `md_num` is NULL
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement MDCFG table format 0

5.1.4.72 iopmp_lock_entries()

```
enum iopmp_error iopmp_lock_entries (
            IOPMP_t * iopmp,
            uint32_t * entry_num,
            bool lock)
```

Lock ENTRY_ADDR[0 ~ (entry_num-1)], ENTRY_ADDRH[0 ~ (entry_num-1)], ENTRY_CFG[0 ~ (entry_num-1)], and ENTRY_USER_CFG[0 ~ (entry_num-1)].

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*entry_num*	Input the number of entry to be locked. Output WARL value
in	*lock*	Set 1 to lock ENTRYLCK register

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if `entry_num` is NULL
IOPMP_ERR_OUT_OF_BOUNDS	if value of `entry_num` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if ENTRYLCK has already been locked
IOPMP_ERR_NOT_ALLOWED	if value of `entry_num` is not monotonically increased
IOPMP_ERR_ILLEGAL_VALUE	if the written `entry_num` does not match the actual value. The actual value is output via

5.1.4.73 iopmp_lock_err_cfg()

```
enum iopmp_error iopmp_lock_err_cfg (
            IOPMP_t * iopmp)
```

Lock fields of ERR_CFG register.

Parameters

in	*iopmp*	The IOPMP instance to be set

Returns

IOPMP_OK

5.1.4.74 iopmp_set_global_intr()

```
enum iopmp_error iopmp_set_global_intr (
            IOPMP_t * iopmp,
            bool enable)
```

Enable/Disable global interrupt.

Parameters

in	*iopmp*	The IOPMP instance to be set
in	*enable*	1(enable) or 0(disable)

Return values

IOPMP_OK	if successes
IOPMP_ERR_REG_IS_LOCKED	if ERR_CFG register is locked

5.1.4.75 iopmp_set_global_err_resp()

```
enum iopmp_error iopmp_set_global_err_resp (
            IOPMP_t * iopmp,
            bool * suppress)
```

Suppress/express global error responses.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*suppress*	Input 1(suppress) or 0(express). Output WARL value

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given suppress is NULL
IOPMP_ERR_REG_IS_LOCKED	if ERR_CFG register is locked
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support configure global error responses
IOPMP_ERR_ILLEGAL_VALUE	if the written suppress does not match the actual value. The actual value is output via s

5.1.4.76 iopmp_set_msi_en()

```
enum iopmp_error iopmp_set_msi_en (
            IOPMP_t * iopmp,
            bool * enable)
```

Enable/disable IOPMP trigger message-signaled interrupts on errors.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*enable*	True to enable or false to disable. Output WARL value

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support MSI
IOPMP_ERR_REG_IS_LOCKED	if ERR_CFG.l=1
IOPMP_ERR_ILLEGAL_VALUE	if the written value do not match the actual value

5.1.4.77 iopmp_get_msi_addr()

```
enum iopmp_error iopmp_get_msi_addr (
            IOPMP_t * iopmp,
            uint64_t * msiaddr64)
```

Get the address to trigger message-signaled interrupts.

Parameters

in	*iopmp*	The IOPMP instance to be got
out	*msiaddr64*	Pointer to 64-bit integer to store MSI address

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support MSI
IOPMP_ERR_INVALID_PARAMETER	if given msiaddr64 is NULL

Note

If HWCFG0.addrh_en=0, the msiaddr64 contains bits 33 to 2 of the MSI address

If HWCFG0.addrh_en=1, the msiaddr64 contains bits 63 to 0 of the MSI address

5.1.4.78 iopmp_get_msi_data()

```
enum iopmp_error iopmp_get_msi_data (
            IOPMP_t * iopmp,
            uint16_t * msidata)
```

Get the data to trigger message-signaled interrupts.

Parameters

in	*iopmp*	The IOPMP instance to be got
out	*msidata*	Pointer to 16-bit integer to store MSI data

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support MSI
IOPMP_ERR_INVALID_PARAMETER	if given `msidata` is NULL

5.1.4.79 iopmp_set_msi_info()

```
enum iopmp_error iopmp_set_msi_info (
            IOPMP_t * iopmp,
            uint64_t * msiaddr64,
            uint16_t * msidata)
```

Set address and data of message-signaled interrupts.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*msiaddr64*	Input 64-bit MSI address. Output WARL value
in,out	*msidata*	Input 11-bit MSI data. Output WARL value

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given `msiaddr64` or `msidata` is NULL
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support MSI, or `msiaddr64` has high-32 bit but IOPMP does not sup
IOPMP_ERR_REG_IS_LOCKED	if ERR_CFG.l=1
IOPMP_ERR_ILLEGAL_VALUE	if the written values do not match the actual values. The actual values are output in `msia`

5.1.4.80 iopmp_get_and_clear_msi_werr()

```
enum iopmp_error iopmp_get_and_clear_msi_werr (
            IOPMP_t * iopmp,
            bool * msi_werr)
```

Check if there is an MSI write error and clear the flag.

Parameters

in	*iopmp*	The IOPMP instance to be checked
out	*msi_werr*	The pointer to flag

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given msi_werr is NULL
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support MSI

5.1.4.81 iopmp_set_stall_violation_en()

```
enum iopmp_error iopmp_set_stall_violation_en (
            IOPMP_t * iopmp,
            bool * enable)
```

Enable or disable the IOPMP faults stalled transactions.

Parameters

in	*iopmp*	The IOPMP instance to be set
in,out	*enable*	Input 1 to enable, 0 to disable. Output WARL value

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support stall
IOPMP_ERR_INVALID_PARAMETER	if given enable is NULL
IOPMP_ERR_ILLEGAL_VALUE	if enable can't be written into iopmp

5.1.4.82 iopmp_invalidate_error()

```
enum iopmp_error iopmp_invalidate_error (
            IOPMP_t * iopmp)
```

Invalidate the error record by clearing ERR_INFO.v bit.

Parameters

in	*iopmp*	The IOPMP instance to be invalidated

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not support clear error interrupt pending bit

5.1.4.83 iopmp_capture_error()

```
enum iopmp_error iopmp_capture_error (
            IOPMP_t * iopmp,
            IOPMP_ERR_REPORT_t * err_report,
            bool invalidate)
```

Capture an IOPMP error information.

Parameters

in	*iopmp*	The IOPMP instance to be captured
out	*err_report*	The pointer to IOPMP error report structure
in	*invalidate*	Flag to clear V bit after reading error report

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given `err_report` is NULL
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support capture error
IOPMP_ERR_NOT_EXIST	if there is no an pending error

5.1.4.84 iopmp_mfr_get_sv_window()

```
enum iopmp_error iopmp_mfr_get_sv_window (
            IOPMP_t * iopmp,
            uint16_t * svi,
            uint16_t * svw)
```

Get subsequent violation window, if IOPMP supports MFR extension.

Parameters

in	*iopmp*	The IOPMP instance to be allocated
in,out	*svi*	When calling, user can specify start index of search windows. When this function returns with IOPMP_OK, sv
out	*svw*	When this function returns with IOPMP_OK, svw indicates the content of window which has subsequent violat

Return values

IOPMP_OK	if at least one subsequent violation is found
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not support MFR extension
IOPMP_ERR_INVALID_PARAMETER	if given `svi` or `svw` is NULL
IOPMP_ERR_NOT_EXIST	if there is no any subsequent violation

5.1.4.85 iopmp_lock_srcmd_table_fmt_0()

```
enum iopmp_error iopmp_lock_srcmd_table_fmt_0 (
            IOPMP_t * iopmp,
            uint32_t rrid)
```

Lock SRCMD_EN(rrid), SRCMD_ENH(rrid), SRCMD_R(rrid), SRCMD_RH(rrid), SRCMD_W(rrid), and SRCMD_↩ WH(rrid) if any.

Parameters

in	*iopmp*	The IOPMP instance to be set
in	*rrid*	The RRID to be locked

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` SRCMD_FMT!=0
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds

Note

This operation is only supported by SRCMD_FMT=0

5.1.4.86 iopmp_is_srcmd_table_fmt_0_locked()

```
enum iopmp_error iopmp_is_srcmd_table_fmt_0_locked (
            IOPMP_t * iopmp,
            uint32_t rrid,
            bool * locked)
```

Check if SRCMD_EN(rrid), SRCMD_ENH(rrid), SRCMD_R(rrid), SRCMD_RH(rrid), SRCMD_W(rrid), and SRCMD_WH(rrid) if any, have been locked.

Parameters

`in`	*iopmp*	The IOPMP instance to be got
`in`	*rrid*	The RRID to be got
`out`	*locked*	The pointer to an integer to store SRCMD_EN.l

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` SRCMD_FMT!=0
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `locked` is NULL

Note

This operation is only supported by SRCMD_FMT=0

5.1.4.87 iopmp_lock_srcmd_table_fmt_2()

```
enum iopmp_error iopmp_lock_srcmd_table_fmt_2 (
            IOPMP_t * iopmp,
            uint32_t mdidx)
```

Lock SRCMD_PERM(mdidx) and SRCMD_PERMH(mdidx)

Parameters

`in`	*iopmp*	The IOPMP instance to be set
`in`	*mdidx*	The index of MD to be locked

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` SRCMD_FMT!=2
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if MDLCK has been locked

Note

This operation is only supported by SRCMD_FMT=2

5.1.4.88 iopmp_is_srcmd_table_fmt_2_locked()

```
enum iopmp_error iopmp_is_srcmd_table_fmt_2_locked (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            bool * locked)
```

Check if SRCMD_PERM(mdidx) and SRCMD_PERMH(mdidx) have been locked.

Parameters

`in`	*iopmp*	The IOPMP instance to be got
`in`	*mdidx*	The index of MD to be got
`out`	*locked*	The pointer to an integer to store SRCMD_EN.l

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` SRCMD_FMT!=2
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `locked` is NULL

Note

This operation is only supported by SRCMD_FMT=2

5.1.4.89 iopmp_get_rrid_md_association()

```
enum iopmp_error iopmp_get_rrid_md_association (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t * mds,
            bool * lock)
```

Get the associated MD bitmap and lock bit of given RRID.

Parameters

`in`	*iopmp*	The IOPMP instance to be got
`in`	*rrid*	The RRID to be got
`out`	*mds*	The pointer to an integer to store SRCMD_EN.md
`out`	*lock*	The pointer to an integer to store SRCMD_EN.l

Return values

IOPMP_OK	if successes
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `lock` or `md` is NULL

5.1.4.90 iopmp_set_rrid_md_association()

```
enum iopmp_error iopmp_set_rrid_md_association (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t mds_set,
            uint64_t mds_clr,
            uint64_t * mds,
            bool lock)
```

Associate/Disassociate the given RRID with the given MD bitmap.

Parameters

`in`	*iopmp*	The IOPMP instance to be set
`in`	*rrid*	The RRID to be set
`in`	*mds_set*	The desired MDs to be associated with `rrid`
`in`	*mds_clr*	The desired MDs to be disassociated with `rrid`
`out`	*mds*	The pointer to an integer to store WARL value of SRCMD_EN.md after setting
`in`	*lock*	Set 1 to lock SRCMD_EN[rrid]

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` SRCMD_FMT!=0
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` or `mds_set` or `mds_clr` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `mds` is NULL
IOPMP_ERR_REG_IS_LOCKED	if SRCMD_EN[rrid] has been locked or some or MDs are locked by MDLCK
IOPMP_ERR_ILLEGAL_VALUE	if the written `mds` does not match the actual value

5.1.4.91 iopmp_set_md_permission()

```
enum iopmp_error iopmp_set_md_permission (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint32_t mdidx,
            bool * r,
            bool * w)
```

(srcmd_fmt=2 only) Set single RRID's r/w permissions to MD

Parameters

`in`	*iopmp*	The IOPMP instance to be set
`in`	*rrid*	The RRID to be set
`in`	*mdidx*	The desired MD to be given permission
`in,out`	*r*	Set true to give the read permission to `rrid` Output WARL value
`in,out`	*w*	Set true to give the write permission to `rrid` Output WARL value

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement SRCMD table format 2
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` or `mdidx` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if MD(`mdidx`) has been locked by MDLCK
IOPMP_ERR_ILLEGAL_VALUE	if the written `r` or `w` do not match the actual values

5.1.4.92 iopmp_set_md_permission_multi()

```
enum iopmp_error iopmp_set_md_permission_multi (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            IOPMP_SRCMD_PERM_CFG_t * cfg)
```

(srcmd_fmt=2 only) Set multiple RRID's r/w permissions to MD

Parameters

`in`	*iopmp*	The IOPMP instance to be set
`in`	*mdidx*	The desired MD to be given permission
`in`	*cfg*	The configuration structure for SRCMD format 2

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement SRCMD table format 2
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `cfg` is NULL
IOPMP_ERR_REG_IS_LOCKED	if MD(`mdidx`) has been locked by MDLCK
IOPMP_ERR_ILLEGAL_VALUE	if the written permissions in `cfg` do not match the actual values

5.1.4.93 iopmp_set_srcmd_perm_cfg()

```
enum iopmp_error iopmp_set_srcmd_perm_cfg (
            IOPMP_SRCMD_PERM_CFG_t * cfg,
            uint32_t rrid,
            bool r,
            bool w)
```

Helper function used to set struct iopmp_srcmd_perm_config.

Parameters

`in`	*cfg*	Pointer to struct iopmp_srcmd_perm_config
`in`	*rrid*	Desired RRID to be set
`in`	*r*	Set true to give RRID read permission; false to clear read permission
`in`	*w*	Set true to give RRID write permission; false to clear write permission

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given `cfg` is NULL
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds

5.1.4.94 iopmp_set_srcmd_perm_cfg_nocheck()

```
void iopmp_set_srcmd_perm_cfg_nocheck (
            IOPMP_SRCMD_PERM_CFG_t * cfg,
            uint32_t rrid,
            bool r,
            bool w)
```

Helper function used to set struct iopmp_srcmd_perm_config. This is similar to iopmp_set_srcmd_perm_cfg() but there are no checks on cfg and RRID.

Parameters

`in`	*cfg*	Pointer to struct iopmp_srcmd_perm_config
`in`	*rrid*	Desired RRID to be set
`in`	*r*	Set true to give RRID read permission; false to clear read permission
`in`	*w*	Set true to give RRID write permission; false to clear write permission

5.1.4.95 iopmp_sps_set_rrid_md_read()

```
enum iopmp_error iopmp_sps_set_rrid_md_read (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t mds_set,
            uint64_t mds_clr,
            uint64_t * mds)
```

(SPS only) Set RRID's read permission to MDs

Parameters

`in`	*iopmp*	The IOPMP instance
`in`	*rrid*	The RRID to be set
`in`	*mds_set*	The desired MDs to set permission to `rrid`
`in`	*mds_clr*	The desired MDs to clear permission to `rrid`
`out`	*mds*	The pointer to an integer to store WARL value of SRCMD_R.md after setting

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement SPS extension
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` or `mds` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if register has been locked by SRCMD_EN.l
IOPMP_ERR_ILLEGAL_VALUE	if the written `mds` does not match the actual values

5.1.4.96 iopmp_sps_get_rrid_md_read()

```
enum iopmp_error iopmp_sps_get_rrid_md_read (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t * mds)
```

(SPS only) Get RRID's read permission to MDs

Parameters

in	*iopmp*	The IOPMP instance
in	*rrid*	The RRID to be checked
out	*mds*	Pointer to variable to output permission

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement SPS extension
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `mds` is NULL

5.1.4.97 iopmp_sps_set_rrid_md_write()

```
enum iopmp_error iopmp_sps_set_rrid_md_write (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t mds_set,
            uint64_t mds_clr,
            uint64_t * mds)
```

(SPS only) Set RRID's write permission to MDs

Parameters

in	*iopmp*	The IOPMP instance
in	*rrid*	The RRID to be set
in	*mds_set*	The desired MDs to set permission to `rrid`
in	*mds_clr*	The desired MDs to clear permission to `rrid`
out	*mds*	The pointer to an integer to store WARL value of SRCMD_W.md after setting

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement SPS extension
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` or `mds` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if register has been locked by SRCMD_EN.l
IOPMP_ERR_ILLEGAL_VALUE	if the written `mds` does not match the actual values

5.1.4.98 iopmp_sps_get_rrid_md_write()

```
enum iopmp_error iopmp_sps_get_rrid_md_write (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t * mds)
```

(SPS only) Get RRID's write permission to MDs

Parameters

in	*iopmp*	The IOPMP instance
in	*rrid*	The RRID to be checked
out	*mds*	Pointer to variable to output permission

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not implement SPS extension
IOPMP_ERR_OUT_OF_BOUNDS	if given rrid is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given mds is NULL

5.1.4.99 iopmp_sps_set_rrid_md_rw()

```
enum iopmp_error iopmp_sps_set_rrid_md_rw (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t mds_set_r,
            uint64_t mds_clr_r,
            uint64_t mds_set_w,
            uint64_t mds_clr_w,
            uint64_t * mds_r,
            uint64_t * mds_w)
```

(SPS only) Set RRID's read and write permission to MDs

Parameters

in	*iopmp*	The IOPMP instance
in	*rrid*	The RRID to be set
in	*mds_set_r*	The desired MDs to set R permission to rrid
in	*mds_clr_r*	The desired MDs to clear R permission to rrid
in	*mds_set_w*	The desired MDs to set W permission to rrid
in	*mds_clr_w*	The desired MDs to clear W permission to rrid
out	*mds_r*	The pointer to an integer to store WARL value of SRCMD_R.md after setting
out	*mds_w*	The pointer to an integer to store WARL value of SRCMD_W.md after setting

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if iopmp does not implement SPS extension
IOPMP_ERR_OUT_OF_BOUNDS	if given rrid or mds_r or mds_w is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if register has been locked by SRCMD_EN.l
IOPMP_ERR_ILLEGAL_VALUE	if the written mds_r or mds_w does not match the actual values

5.1.4.100 iopmp_sps_get_rrid_md_rw()

```
enum iopmp_error iopmp_sps_get_rrid_md_rw (
            IOPMP_t * iopmp,
            uint32_t rrid,
            uint64_t * mds_r,
            uint64_t * mds_w)
```

(SPS only) Get RRID's read and write permission to multiple MDs

Parameters

in	*iopmp*	The IOPMP instance
in	*rrid*	The RRID to be set
out	*mds_r*	Pointer to variable to output read permission
out	*mds_w*	Pointer to variable to output write permission

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_SUPPORTED	if `iopmp` does not implement SPS extension
IOPMP_ERR_OUT_OF_BOUNDS	if given `rrid` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `mds_r` or `mds_w` is NULL

5.1.4.101 iopmp_get_md_entry_association()

```
enum iopmp_error iopmp_get_md_entry_association (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            uint32_t * entry_idx_start,
            uint32_t * num_entry)
```

Get start index and number of the entries belong to MD[mdidx].

Parameters

in	*iopmp*	The IOPMP instance to be got
in	*mdidx*	The index of target MD
out	*entry_idx_start*	The pointer to an integer to return start index
out	*num_entry*	The pointer to an integer to return number of entry

Return values

IOPMP_OK	if successes
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `entry_idx_start` or `num_entry` is NULL

5.1.4.102 iopmp_set_md_entry_association_multi()

```
enum iopmp_error iopmp_set_md_entry_association_multi (
            IOPMP_t * iopmp,
            uint32_t mdidx_start,
            uint32_t * num_entries,
            uint32_t md_num)
```

Associate given entries with given multiple MDs.

Parameters

in	*iopmp*	The IOPMP instance to be set
in	*mdidx_start*	The start index of target MDs
in,out	*num_entries*	Input the number of entries to be associated. Output actual number of entries be associated.
in	*md_num*	The number of target MDs to be set

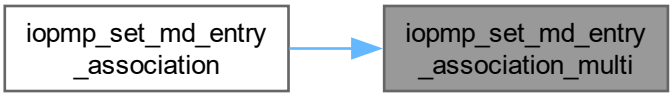
Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_ALLOWED	if `iopmp` MDCFG format is not 0
IOPMP_ERR_INVALID_PARAMETER	if given `num_entries` is NULL
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx_start` or `md_num` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if MDCFG of given `mdidx_start` has been locked by MDCFGLCK.f
IOPMP_ERR_ILLEGAL_VALUE	if the written `num_entries` does not match the actual value. The actual value is output

Note

This function must be called only when IOPMP MDCFG format is 0

Here is the caller graph for this function:



5.1.4.103 iopmp_set_md_entry_association()

```
enum iopmp_error iopmp_set_md_entry_association (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            uint32_t * num_entry)  [inline], [static]
```

Associate given entries with given MD(mdidx)

Parameters

`in`	*iopmp*	The IOPMP instance to be set
`in`	*mdidx*	The index of target MD
`in,out`	*num_entry*	Input the number of entries to be associated. Output actual number of entries be associated.

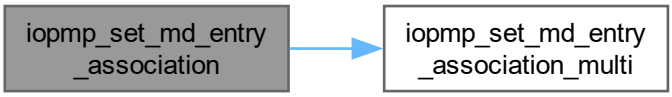
Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_ALLOWED	if `iopmp` MDCFG format is not 0
IOPMP_ERR_INVALID_PARAMETER	if given `num_entry` is NULL
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx` or `num_entry` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if MDCFG of given `mdidx` has been locked by MDCFGLCK.f
IOPMP_ERR_ILLEGAL_VALUE	if the written `num_entry` does not match the actual value. The actual value is output in

Note

This function must be called only when IOPMP MDCFG format is 0

Here is the call graph for this function:



5.1.4.104 iopmp_get_md_entry_num()

```
enum iopmp_error iopmp_get_md_entry_num (
            IOPMP_t * iopmp,
            uint32_t * md_entry_num)
```

Get value of HWCFG0.md_entry_num if IOPMP model is xxx-K.

Parameters

in	*iopmp*	The IOPMP instance to be got
out	*md_entry_num*	The pointer to an integer to return value

Return values

IOPMP_OK	if successes
IOPMP_ERR_INVALID_PARAMETER	if given `md_entry_num` is NULL

5.1.4.105 iopmp_set_md_entry_num()

```
enum iopmp_error iopmp_set_md_entry_num (
            IOPMP_t * iopmp,
            uint32_t * md_entry_num)
```

Program value of HWCFG0.md_entry_num.

Parameters

in	*iopmp*	The IOPMP instance to be programmed
in,out	*md_entry_num*	Input the drsired value of md_entry_num. Output WARL value.

Return values

IOPMP_OK	if successes
IOPMP_ERR_NOT_ALLOWED	if IOPMP MDCFG format is not 2

IOPMP_ERR_INVALID_PARAMETER	if given `md_entry_num` is NULL
IOPMP_ERR_REG_IS_LOCKED	if IOPMP has been enabled
IOPMP_ERR_OUT_OF_BOUNDS	if given `md_entry_num` is out-of-bounds
IOPMP_ERR_ILLEGAL_VALUE	if the written `md_entry_num` does not match the actual value. The actual value is outpu

Note

This function must be called only when IOPMP's MDCFG_FMT=2

5.1.4.106 iopmp_encode_entry()

```
enum iopmp_error iopmp_encode_entry (
            IOPMP_t * iopmp,
            struct iopmp_entry * entries,
            uint32_t num_entry,
            uint64_t addr,
            uint64_t size,
            enum iopmp_entry_flags flags,
            uint64_t private_data)
```

Encode IOPMP entry from given memory region and flags.

Parameters

`in`	*iopmp*	The IOPMP instance
`out`	*entries*	The array of entry to be output
`in`	*num_entry*	Number of entries in `entries`
`in`	*addr*	Address of the memory region
`in`	*size*	Size of the memory region
`in`	*flags*	Flags of the entry for this memory region
`in`	*private_data*	Private data that can be used in specific model

Return values

1	if successes and the memory region is encoded as NAPOT entry or as TOR entry 0
2	if successes and the memory region is encoded as two TOR entries
IOPMP_ERR_INVALID_PARAMETER	if given `entries` is NULL or `num_entry` is 0 or `size` is 0 or `addr` or `size` is not ali
IOPMP_ERR_NOT_SUPPORTED	if memory region should be encoded as TOR entry, but `iopmp` does not support TOR ent
IOPMP_ERR_NOT_ALLOWED	if memory region should be encoded as TOR, but only one entry in given `entries` ; or if

Note

Caller is responsible for providing the permission bits and per-entry interrupt/error suppression bits via `flags` parameter.

The address-matching mode of the entry will be determined by this API. Caller doesn't need to provide the address matching mode via `flags` parameter, such as IOPMP_ENTRY_A_TOR or IOPMP_ENTRY_A_↩ NAPOT. Caller can check the address-matching mode by entry->a field after returning from this API.

If caller wants to encode the the entry as "OFF" address-matching mode, caller must provide IOPMP_↩ ENTRY_FORCE_OFF via `flags` parameter.

In general, a TOR region will be encoded into two entries. However, the specification permits the PMP entry 0 having TOR address-matching mode. In this case, caller must provide IOPMP_ENTRY_FIRST_TOR via `flags` parameter. The API returns 1 whereas the entry is encoded as TOR address-matching mode.

If caller wants to encode TOR entries on an NAPOT-able region, caller must provide IOPMP_ENTRY_↩ FORCE_TOR via `flags` parameter.

If caller wants to specify the entry is whether a priority entry or a non-priority entry, caller can provide IOPMP↩ _ENTRY_PRIO or IOPMP_ENTRY_NON_PRIO via `flags` parameter. The iopmp_set_entries() and similar APIs will check the priority on the entry. If the caller provides neither of them, the iopmp_set_entries() and similar APIs won't check the priority on the entry.

Currently, the `private_data` is used in a specific model with SRCMD format 2 and MDCFG format 1 and HWCFG0.md_entry_num=0 configurations. In this case, the `private_data` encodes {SRCMD_PERM(H) | SRCMD_PERM} for the entry associated with a single MD.

5.1.4.107 iopmp_set_entries_to_md()

```
enum iopmp_error iopmp_set_entries_to_md (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            const struct iopmp_entry * entry_array,
            uint32_t idx_start,
            uint32_t num_entry)
```

Set the entries belong to given MD to IOPMP.

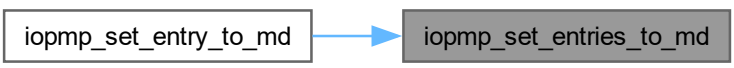
Parameters

in	*iopmp*	The IOPMP instance to be written
in	*mdidx*	The index of target MD
in	*entry_array*	The array of entries
in	*idx_start*	The local index of entry in target MD
in	*num_entry*	The number of entries to be set

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx`, `idx_start` or `num_entry` is out of bounds
IOPMP_ERR_INVALID_PRIORITY	if priority of entry is invalid
IOPMP_ERR_REG_IS_LOCKED	if entries from `idx_start` have been locked by ENTRYLCK.f

Here is the caller graph for this function:



5.1.4.108 iopmp_set_entry_to_md()

```
enum iopmp_error iopmp_set_entry_to_md (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            const struct iopmp_entry * entry,
            uint32_t idx)  [inline], [static]
```

Set single entry belong to given MD to IOPMP.

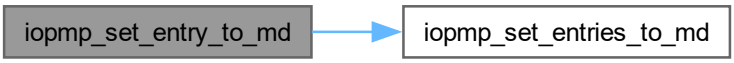
Parameters

in	*iopmp*	The IOPMP instance to be written
in	*mdidx*	The index of target MD
in	*entry*	The pointer to the entry
in	*idx*	The local index of entry in target MD

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx`, `idx_start` or `num_entry` is out of bounds
IOPMP_ERR_INVALID_PRIORITY	if priority of entry is invalid
IOPMP_ERR_REG_IS_LOCKED	if entries from `idx_start` have been locked by ENTRYLCK.f

Here is the call graph for this function:



5.1.4.109 iopmp_get_entries_from_md()

```
enum iopmp_error iopmp_get_entries_from_md (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            struct iopmp_entry * entry_array,
            uint32_t idx_start,
            uint32_t num_entry)
```

Get the entries belong to given MD from IOPMP.

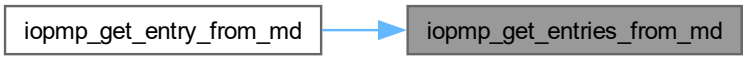
Parameters

in	*iopmp*	The IOPMP instance to be read
in	*mdidx*	The index of target MD
out	*entry_array*	The array of entries
in	*idx_start*	The local start index of entries in target MD
in	*num_entry*	The number of entries to be read

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx`, `idx_start` or `num_entry` is out of bounds

Here is the caller graph for this function:



5.1.4.110 iopmp_get_entry_from_md()

```
enum iopmp_error iopmp_get_entry_from_md (
            IOPMP_t * iopmp,
            uint32_t mdidx,
            struct iopmp_entry * entry,
            uint32_t idx)  [inline], [static]
```

Get single entry belong to given MD from IOPMP.

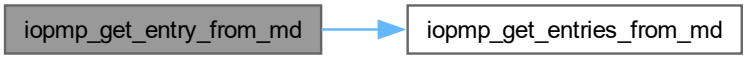
Parameters

`in`	*iopmp*	The IOPMP instance to be read
`in`	*mdidx*	The index of target MD
`out`	*entry*	The pointer to the entry
`in`	*idx*	The local start index of entries in target MD

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx`, `idx_start` or `num_entry` is out of bounds

Here is the call graph for this function:



5.1.4.111 iopmp_get_entries()

```
enum iopmp_error iopmp_get_entries (
            IOPMP_t * iopmp,
            struct iopmp_entry * entry_array,
            uint32_t idx_start,
            uint32_t num_entry)
```

Get the global entries from IOPMP.

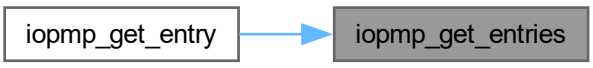
Parameters

in	*iopmp*	The IOPMP instance to be read
out	*entry_array*	The array of entries
in	*idx_start*	The global start index of target entries
in	*num_entry*	The number of entries to be read

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx_start` or `num_entry` is out of bounds

Here is the caller graph for this function:



5.1.4.112 iopmp_get_entry()

```
enum iopmp_error iopmp_get_entry (
            IOPMP_t * iopmp,
            struct iopmp_entry * entry,
            uint32_t idx)  [inline], [static]
```

Get single global entry from IOPMP.

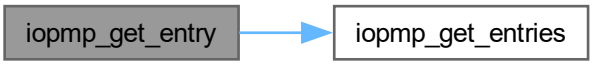
Parameters

in	*iopmp*	The IOPMP instance to be read
out	*entry*	The pointer to the entry
in	*idx*	The global start index of target entries

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx_start` or `num_entry` is out of bounds

Here is the call graph for this function:



5.1.4.113 iopmp_set_entries()

```
enum iopmp_error iopmp_set_entries (
            IOPMP_t * iopmp,
            const struct iopmp_entry * entry_array,
            uint32_t idx_start,
            uint32_t num_entry)
```

Set the global entries into IOPMP.

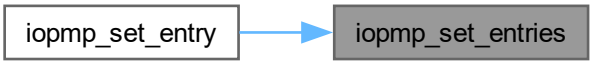
Parameters

`in`	*iopmp*	The IOPMP instance to be written
`in`	*entry_array*	The array of entries
`in`	*idx_start*	The global start index of target entries
`in`	*num_entry*	The number of entries to be written

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx_start` or `num_entry` is out of bounds
IOPMP_ERR_INVALID_PRIORITY	if priority of entry is invalid
IOPMP_ERR_REG_IS_LOCKED	if entries from `idx_start` have been locked by ENTRYLCK.f

Here is the caller graph for this function:



5.1.4.114 iopmp_set_entry()

```
enum iopmp_error iopmp_set_entry (
            IOPMP_t * iopmp,
            const struct iopmp_entry * entry,
            uint32_t idx)  [inline], [static]
```

Set single global entry into IOPMP.

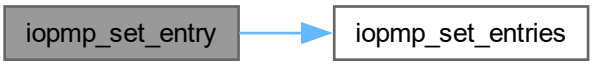
Parameters

in	*iopmp*	The IOPMP instance to be written
in	*entry*	The pointer to the entry
in	*idx*	The global start index of target entries

Return values

IOPMP_OK	on success
IOPMP_ERR_INVALID_PARAMETER	if given `entry_array` is NULL or `num_entry` is 0
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx_start` or `num_entry` is out of bounds
IOPMP_ERR_INVALID_PRIORITY	if priority of entry is invalid
IOPMP_ERR_REG_IS_LOCKED	if entries from `idx_start` have been locked by ENTRYLCK.f

Here is the call graph for this function:



5.1.4.115 iopmp_clear_entries_in_md()

```
enum iopmp_error iopmp_clear_entries_in_md (
            IOPMP_t * iopmp,
            uint32_t mdidx)
```

Clear IOPMP entries in MD.

Parameters

in	*iopmp*	The IOPMP instance
in	*mdidx*	The index of target MD

Return values

IOPMP_OK	on success
IOPMP_ERR_OUT_OF_BOUNDS	if given `mdidx` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if entries in MD `mdidx` have been locked by ENTRYLCK.f

5.1.4.116 iopmp_clear_entries()

```
enum iopmp_error iopmp_clear_entries (
            IOPMP_t * iopmp,
            uint32_t idx_start,
            uint32_t num_entry)
```

Clear IOPMP entries.

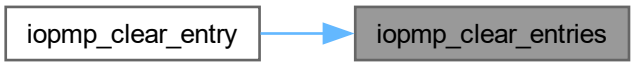
Parameters

in	*iopmp*	The IOPMP instance
in	*idx_start*	The global start index of target entries
in	*num_entry*	The number of entries to be cleared

Return values

IOPMP_OK	on success
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx_start` or `num_entry` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if some of entries have been locked by ENTRYLCK.f

Here is the caller graph for this function:



5.1.4.117 iopmp_clear_entry()

```
enum iopmp_error iopmp_clear_entry (
            IOPMP_t * iopmp,
            uint32_t idx)  [inline], [static]
```

Clear single global entry.

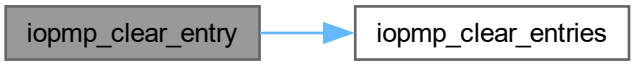
Parameters

in	*iopmp*	The IOPMP instance
in	*idx*	The global index of target entry

Return values

IOPMP_OK	on success
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx` is out of bounds
IOPMP_ERR_REG_IS_LOCKED	if some of entries have been locked by ENTRYLCK.f

Here is the call graph for this function:



5.1.4.118 iopmp_entries_get_belong_md()

```
enum iopmp_error iopmp_entries_get_belong_md (
            IOPMP_t * iopmp,
            uint32_t idx_start,
            uint32_t num_entry,
            uint64_t * mds)
```

Get the MD bitmap that given index range of IOPMP entries belong to.

Parameters

in	*iopmp*	The IOPMP instance
in	*idx_start*	The global start index of target entries
in	*num_entry*	The number of entries to be checked
out	*mds*	Pointer to integer to store MD bitmap

Return values

IOPMP_OK	on success
IOPMP_ERR_OUT_OF_BOUNDS	if given `idx_start` or `num_entry` is out of bounds
IOPMP_ERR_INVALID_PARAMETER	if given `mds` is NULL

5.2 libiopmp.h

Go to the documentation of this file.

```
00001 /*
00002  * Copyright 2018-2025 Andes Technology Corporation. All rights reserved.
00003  *
00004  * Licensed under the Apache License, Version 2.0 (the "License");
00005  * you may not use this file except in compliance with the License.
00006  * You may obtain a copy of the License at
00007  *
00008  *     http://www.apache.org/licenses/LICENSE-2.0
00009  *
00010  * Unless required by applicable law or agreed to in writing, software
00011  * distributed under the License is distributed on an "AS IS" BASIS,
00012  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00013  * See the License for the specific language governing permissions and
00014  * limitations under the License.
00015  */
```

```
00016
00017 #ifndef __LIBIOPMP_H__
00018 #define __LIBIOPMP_H__
00019
00020 #include <stdbool.h>
00021 #include <stddef.h>
00022 #include <stdint.h>
00023
00024 /*****************************************************************************/
00025 /* libiopmp data structure.                                                  */
00026 /*****************************************************************************/
00031 struct iopmp_instance {
00033     uintptr_t addr;
00035     uint32_t granularity;
00037     uint64_t entry_addr_bits;
00039     struct iopmp_operations_generic *ops_generic;
00041     struct iopmp_operations_specific *ops_specific;
00043     struct iopmp_operations_sps *ops_sps;
00044
00046     uintptr_t addr_entry_array;
00047
00049     uint32_t vendor;
00051     uint32_t impid;
00053     uint16_t rrid_num;
00055     uint16_t entry_num;
00057     uint16_t prio_entry_num;
00059     uint16_t rrid_transl;
00061     uint8_t specver;
00063     uint8_t md_num;
00065     uint8_t md_entry_num;
00066
00068     uint8_t mdlck_lock;
00070     uint64_t mdlck_md;
00072     uint8_t mdcfglck_lock;
00074     uint8_t mdcfglck_f;
00076     uint8_t entrylck_lock;
00078     uint16_t entrylck_f;
00079
00085     uint64_t msiaddr64;
00087     uint16_t msidata;
00088
00090     struct {
00092         unsigned int init : 1;
00094         unsigned int mdcfg_fmt : 2;
00096         unsigned int srcmd_fmt : 2;
00098         unsigned int tor_en : 1;
00100         unsigned int sps_en : 1;
00102         unsigned int user_cfg_en : 1;
00104         unsigned int prient_prog : 1;
00109         unsigned int rrid_transl_en : 1;
00111         unsigned int rrid_transl_prog : 1;
00116         unsigned int chk_x : 1;
00121         unsigned int no_x : 1;
00126         unsigned int no_w : 1;
00128         unsigned int stall_en : 1;
00133         unsigned int peis : 1;
00137         unsigned int pees : 1;
00142         unsigned int mfr_en : 1;
00147         unsigned int addrh_en : 1;
00149         unsigned int enable : 1;
00151         unsigned int err_cfg_lock : 1;
00153         unsigned int intr_enable : 1;
00155         unsigned int err_resp_suppress : 1;
00157         unsigned int msi_en : 1;
00159         unsigned int stall_violation_en : 1;
00161         unsigned int support_stall_by_rrid : 1;
00163         unsigned int support_stall_by_md : 1;
00165         unsigned int is_stalling : 1;
00166     };
00167 };
00168
00176 enum iopmp_prient_flags {
00178     IOPMP_PRIENT_ANY = 0,
00180     IOPMP_PRIENT_PRIORITY = (1 « 0),
00182     IOPMP_PRIENT_NON_PRIORITY = (1 « 1),
00183 };
00184
00190 struct iopmp_entry {
00192     union {
00193         struct {
00195             uint32_t addrl;
00197             uint32_t addrh;
00198         };
00200         uint64_t addr;
00201     };
00202
00204     union {
```

```
00205         struct {
00207             uint32_t r   : 1;
00209             uint32_t w   : 1;
00211             uint32_t x   : 1;
00213             uint32_t a   : 2;
00215             uint32_t sire: 1;
00217             uint32_t siwe: 1;
00219             uint32_t sixe: 1;
00221             uint32_t sere: 1;
00223             uint32_t sewe: 1;
00225             uint32_t sexe: 1;
00227             uint32_t rsv : 21;
00228         };
00230         uint32_t cfg;
00231     };
00232
00234     enum iopmp_prient_flags prient_flag;
00235
00244     uint64_t private_data;
00245 };
00246
00248 enum iopmp_errinfo_ttype {
00250     IOPMP_ERRINFO_TTYPE_RSVD       = 0x00,
00252     IOPMP_ERRINFO_TTYPE_READ       = 0x01,
00254     IOPMP_ERRINFO_TTYPE_WRITE      = 0x02,
00256     IOPMP_ERRINFO_TTYPE_INST_FETCH = 0x03,
00257 };
00258
00260 enum iopmp_errinfo_etype {
00262     IOPMP_ERRINFO_ETYPE_NONE         = 0x00,
00264     IOPMP_ERRINFO_ETYPE_READ         = 0x01,
00266     IOPMP_ERRINFO_ETYPE_WRITE        = 0x02,
00268     IOPMP_ERRINFO_ETYPE_INST_FETCH   = 0x03,
00270     IOPMP_ERRINFO_ETYPE_PART_HIT     = 0x04,
00272     IOPMP_ERRINFO_ETYPE_NOT_HIT      = 0x05,
00274     IOPMP_ERRINFO_ETYPE_UNKNOWN_RRID = 0x06,
00276     IOPMP_ERRINFO_ETYPE_STALL        = 0x07,
00278     IOPMP_ERRINFO_ETYPE_RESERVED_0   = 0x08,
00280     IOPMP_ERRINFO_ETYPE_RESERVED_1   = 0x09,
00282     IOPMP_ERRINFO_ETYPE_RESERVED_2   = 0x0A,
00284     IOPMP_ERRINFO_ETYPE_RESERVED_3   = 0x0B,
00286     IOPMP_ERRINFO_ETYPE_RESERVED_4   = 0x0C,
00288     IOPMP_ERRINFO_ETYPE_RESERVED_5   = 0x0D,
00290     IOPMP_ERRINFO_ETYPE_USER_DEF_0   = 0x0E,
00292     IOPMP_ERRINFO_ETYPE_USER_DEF_1   = 0x0F,
00293 };
00294
00296 struct iopmp_err_report {
00298     uint64_t addr;
00300     uint32_t rrid;
00302     uint32_t eid;
00304     enum iopmp_errinfo_ttype ttype;
00306     enum iopmp_errinfo_etype etype;
00308     bool msi_werr;
00310     bool svc;
00311 };
00312
00313 typedef struct iopmp_instance IOPMP_t;
00314
00315 typedef struct iopmp_entry IOPMP_Entry_t;
00316
00317 typedef struct iopmp_err_report IOPMP_ERR_REPORT_t;
00318
00320 #define IOPMP_MAX_RRID_SRCMD_FMT_2  32
00321
00330 struct iopmp_srcmd_perm_config {
00332 #define IOPMP_SRCMD_PERM_R      (1 « 0)
00334 #define IOPMP_SRCMD_PERM_W      (1 « 1)
00336 #define IOPMP_SRCMD_PERM_MASK   (IOPMP_SRCMD_PERM_W | IOPMP_SRCMD_PERM_R)
00337
00343     uint64_t srcmd_perm_mask;
00344
00351     uint64_t srcmd_perm_val;
00352 };
00353 typedef struct iopmp_srcmd_perm_config IOPMP_SRCMD_PERM_CFG_t;
00354
00362 #define IOPMP_SRCMD_PERM_CFG_SET_DIRECT(cfg, mask, val) \
00363     do {                                                \
00364         IOPMP_SRCMD_PERM_CFG_t *__cfg = (cfg);          \
00365         __cfg->srcmd_perm_mask = mask;                  \
00366         __cfg->srcmd_perm_val  = val;                   \
00367     } while (0);
00368
00369 /*****************************************************************************/
00370 /* Supported IOPMP implementation ID                                         */
00371 /*****************************************************************************/
00373 enum iopmp_impid {
```

```
00375     IOPMP_IMPID_NOT_SPECIFIED = 0xFFFFFFFF,
00376 };
00377
00378 /*****************************************************************************/
00379 /* MDCFG_FMT and SRCMD_FMT and models                                        */
00380 /*****************************************************************************/
00382 enum iopmp_srcmd_fmt {
00384     IOPMP_SRCMD_FMT_0,
00386     IOPMP_SRCMD_FMT_1,
00388     IOPMP_SRCMD_FMT_2,
00390     IOPMP_SRCMD_FMT_RESERVED,
00392     IOPMP_SRCMD_FMT_MAX,
00393 };
00394
00396 enum iopmp_mdcfg_fmt {
00398     IOPMP_MDCFG_FMT_0,
00400     IOPMP_MDCFG_FMT_1,
00402     IOPMP_MDCFG_FMT_2,
00404     IOPMP_MDCFG_FMT_RESERVED,
00406     IOPMP_MDCFG_FMT_MAX,
00407 };
00408
00410 enum iopmp_model {
00412     IOPMP_MODEL_FULL        = 0,
00414     IOPMP_MODEL_RAPID_K     = 1,
00416     IOPMP_MODEL_DYNAMIC_K   = 2,
00418     IOPMP_MODEL_RESERVED_3  = 3,
00420     IOPMP_MODEL_ISOLATION   = 4,
00422     IOPMP_MODEL_COMPACT_K   = 5,
00424     IOPMP_MODEL_6           = 6,
00426     IOPMP_MODEL_RESERVED_7  = 7,
00428     IOPMP_MODEL_8           = 8,
00430     IOPMP_MODEL_9           = 9,
00432     IOPMP_MODEL_RESERVED_10 = 10,
00434     IOPMP_MODEL_RESERVED_11 = 11,
00436     IOPMP_MODEL_RESERVED_12 = 12,
00438     IOPMP_MODEL_RESERVED_13 = 13,
00440     IOPMP_MODEL_RESERVED_14 = 14,
00442     IOPMP_MODEL_RESERVED_15 = 15
00443 };
00444
00446 enum iopmp_rridscp_op {
00448     IOPMP_RRIDSCP_OP_QUERY = 0,
00450     IOPMP_RRIDSCP_OP_STALL = 1,
00452     IOPMP_RRIDSCP_OP_DONT_STALL = 2,
00454     IOPMP_RRIDSCP_OP_RESERVED = 3
00455 };
00456
00458 enum iopmp_rridscp_stat {
00460     IOPMP_RRIDSCP_STAT_NOT_IMPL = 0,
00462     IOPMP_RRIDSCP_STAT_STALLED = 1,
00464     IOPMP_RRIDSCP_STAT_NOT_STALLED = 2,
00466     IOPMP_RRIDSCP_STAT_ERR_RRID = 3
00467 };
00468
00469 /*****************************************************************************/
00470 /* The flags used when calling iopmp_encode_entry()                          */
00471 /*****************************************************************************/
00473 enum iopmp_entry_flags {
00475     IOPMP_ENTRY_R = (1UL << 0),
00477     IOPMP_ENTRY_W = (1UL << 1),
00479     IOPMP_ENTRY_X = (1UL << 2),
00481     IOPMP_ENTRY_RW = (IOPMP_ENTRY_R | IOPMP_ENTRY_W),
00483     IOPMP_ENTRY_RX = (IOPMP_ENTRY_R | IOPMP_ENTRY_X),
00485     IOPMP_ENTRY_RWX = (IOPMP_ENTRY_R | IOPMP_ENTRY_W | IOPMP_ENTRY_X),
00486
00488     IOPMP_ENTRY_A_OFF = (0UL << 3),
00490     IOPMP_ENTRY_A_TOR = (1UL << 3),
00492     IOPMP_ENTRY_A_NA4 = (2UL << 3),
00494     IOPMP_ENTRY_A_NAPOT = (3UL << 3),
00496     IOPMP_ENTRY_A_MASK = (3UL << 3),
00497
00499     IOPMP_ENTRY_SIRE = (1UL << 5),
00501     IOPMP_ENTRY_SIWE = (1UL << 6),
00503     IOPMP_ENTRY_SIXE = (1UL << 7),
00505     IOPMP_ENTRY_SIE_MASK = (7UL << 5),
00507     IOPMP_ENTRY_SERE = (1UL << 8),
00509     IOPMP_ENTRY_SEWE = (1UL << 9),
00511     IOPMP_ENTRY_SEXE = (1UL << 10),
00513     IOPMP_ENTRY_SEE_MASK = (7UL << 8),
00514
00516     IOPMP_ENTRY_FORCE_OFF = (1UL << 27),
00518     IOPMP_ENTRY_FIRST_TOR = (1UL << 28),
00520     IOPMP_ENTRY_FORCE_TOR = (1UL << 29),
00521
00523     IOPMP_ENTRY_PRIO = (1UL << 30),
00525     IOPMP_ENTRY_NON_PRIO = (1UL << 31),
```

```
00526
00528     IOPMP_ENTRY_SW_FLAGS_MASK = (IOPMP_ENTRY_FORCE_OFF | IOPMP_ENTRY_FIRST_TOR |
00529                                  IOPMP_ENTRY_FORCE_TOR | IOPMP_ENTRY_PRIO |
00530                                  IOPMP_ENTRY_NON_PRIO),
00531 };
00532
00533 /*****************************************************************************/
00534 /* API Error codes                                                           */
00535 /*****************************************************************************/
00537 enum iopmp_error {
00539     IOPMP_OK                    =  0,
00541     IOPMP_ERR_NOT_SUPPORTED     = -1,
00543     IOPMP_ERR_OUT_OF_BOUNDS     = -2,
00545     IOPMP_ERR_REG_IS_LOCKED     = -3,
00547     IOPMP_ERR_NOT_ALLOWED       = -4,
00549     IOPMP_ERR_NOT_EXIST         = -5,
00551     IOPMP_ERR_NOT_AVAILABLE     = -6,
00553     IOPMP_ERR_INVALID_PARAMETER = -7,
00555     IOPMP_ERR_INVALID_PRIORITY  = -8,
00557     IOPMP_ERR_ILLEGAL_VALUE     = -9,
00558 };
00559
00560 /*****************************************************************************/
00561 /* Helper macros and functions to get libiopmp version information           */
00562 /*****************************************************************************/
00564 #define LIBIOPMP_VERSION_MAJOR          0
00566 #define LIBIOPMP_VERSION_MINOR          1
00568 #define LIBIOPMP_VERSION_EXTRA          0
00569
00571 #define LIBIOPMP_VERSION_MAJOR_SHIFT    16
00573 #define LIBIOPMP_VERSION_MAJOR_MASK     0xffff
00574
00576 #define LIBIOPMP_VERSION_MINOR_SHIFT    8
00578 #define LIBIOPMP_VERSION_MINOR_MASK     0xff
00579
00581 #define LIBIOPMP_VERSION_EXTRA_SHIFT    0
00583 #define LIBIOPMP_VERSION_EXTRA_MASK     0xff
00584
00594 #define LIBIOPMP_VERSION(__major, __minor, __extra) \
00595 ((((__major) & LIBIOPMP_VERSION_MAJOR_MASK) << LIBIOPMP_VERSION_MAJOR_SHIFT) | \
00596  (((__minor) & LIBIOPMP_VERSION_MINOR_MASK) << LIBIOPMP_VERSION_MINOR_SHIFT) | \
00597  (((__extra) & LIBIOPMP_VERSION_EXTRA_MASK) << LIBIOPMP_VERSION_EXTRA_SHIFT))
00598
00604 int libiopmp_major_version(void);
00605
00611 int libiopmp_minor_version(void);
00612
00618 int libiopmp_extra_version(void);
00619
00630 bool libiopmp_check_version(int major, int minor, int extra);
00631
00632 /*****************************************************************************/
00633 /* Helper macros to get/set local variables                                  */
00634 /*****************************************************************************/
00643 static inline bool iopmp_is_initialized(IOPMP_t *iopmp)
00644 {
00645     return iopmp && iopmp->init;
00646 }
00647
00655 static inline uintptr_t iopmp_get_base_addr(IOPMP_t *iopmp)
00656 {
00657     return iopmp->addr;
00658 }
00659
00667 static inline uintptr_t iopmp_get_base_addr_entry_array(IOPMP_t *iopmp)
00668 {
00669     return iopmp->addr_entry_array;
00670 }
00671
00679 static inline uint32_t iopmp_get_granularity(IOPMP_t *iopmp)
00680 {
00681     return iopmp->granularity;
00682 }
00683
00691 static inline enum iopmp_mdcfg_fmt iopmp_get_mdcfg_fmt(IOPMP_t *iopmp)
00692 {
00693     return iopmp->mdcfg_fmt;
00694 }
00695
00703 static inline enum iopmp_srcmd_fmt iopmp_get_srcmd_fmt(IOPMP_t *iopmp)
00704 {
00705     return iopmp->srcmd_fmt;
00706 }
00707
00716 static inline bool iopmp_get_support_tor(IOPMP_t *iopmp)
00717 {
00718     return iopmp->tor_en;
```

```
00719 }
00720
00729 static inline bool iopmp_get_support_sps(IOPMP_t *iopmp)
00730 {
00731     return iopmp->sps_en && iopmp->ops_sps;
00732 }
00733
00742 static inline bool iopmp_get_support_user_entry_cfg(IOPMP_t *iopmp)
00743 {
00744     return iopmp->user_cfg_en;
00745 }
00746
00755 static inline bool iopmp_get_support_programmable_prio_entry(IOPMP_t *iopmp)
00756 {
00757     return iopmp->prient_prog;
00758 }
00759
00768 static inline bool iopmp_get_support_rrid_transl(IOPMP_t *iopmp)
00769 {
00770     return iopmp->rrid_transl_en;
00771 }
00772
00781 static inline bool iopmp_get_rrid_transl_prog(IOPMP_t *iopmp)
00782 {
00783     return iopmp->rrid_transl_prog;
00784 }
00785
00793 static inline uint16_t iopmp_get_rrid_transl(IOPMP_t *iopmp)
00794 {
00795     return iopmp->rrid_transl;
00796 }
00797
00806 static inline bool iopmp_get_support_chk_x(IOPMP_t *iopmp)
00807 {
00808     return iopmp->chk_x;
00809 }
00810
00819 static inline bool iopmp_get_no_x(IOPMP_t *iopmp)
00820 {
00821     return iopmp->no_x;
00822 }
00823
00833 static inline bool iopmp_get_no_w(IOPMP_t *iopmp)
00834 {
00835     return iopmp->no_w;
00836 }
00837
00846 static inline bool iopmp_get_support_stall(IOPMP_t *iopmp)
00847 {
00848     return iopmp->stall_en;
00849 }
00850
00859 static inline bool iopmp_get_support_peis(IOPMP_t *iopmp)
00860 {
00861     return iopmp->peis;
00862 }
00863
00872 static inline bool iopmp_get_support_pees(IOPMP_t *iopmp)
00873 {
00874     return iopmp->pees;
00875 }
00876
00885 static inline bool iopmp_get_support_mfr(IOPMP_t *iopmp)
00886 {
00887     return iopmp->mfr_en;
00888 }
00889
00897 static inline uint32_t iopmp_get_md_num(IOPMP_t *iopmp)
00898 {
00899     return iopmp->md_num;
00900 }
00901
00911 static inline uint32_t iopmp_get_addrh_en(IOPMP_t *iopmp)
00912 {
00913     return iopmp->addrh_en;
00914 }
00915
00924 static inline bool iopmp_get_enable(IOPMP_t *iopmp)
00925 {
00926     return iopmp->enable;
00927 }
00928
00936 static inline uint32_t iopmp_get_rrid_num(IOPMP_t *iopmp)
00937 {
00938     return iopmp->rrid_num;
00939 }
00940
```

```
00948 static inline uint32_t iopmp_get_entry_num(IOPMP_t *iopmp)
00949 {
00950     return iopmp->entry_num;
00951 }
00952
00960 static inline uint16_t iopmp_get_prio_entry_num(IOPMP_t *iopmp)
00961 {
00962     return iopmp->prio_entry_num;
00963 }
00964
00973 static inline bool iopmp_get_support_stall_by_md(IOPMP_t *iopmp)
00974 {
00975     return iopmp->support_stall_by_md;
00976 }
00977
00986 static inline bool iopmp_get_support_stall_by_rrid(IOPMP_t *iopmp)
00987 {
00988     return iopmp->support_stall_by_rrid;
00989 }
00990
00999 static inline bool iopmp_is_err_cfg_locked(IOPMP_t *iopmp)
01000 {
01001     return iopmp->err_cfg_lock;
01002 }
01003
01012 static inline bool iopmp_get_global_intr(IOPMP_t *iopmp)
01013 {
01014     return iopmp->intr_enable;
01015 }
01016
01025 static inline bool iopmp_get_global_err_resp(IOPMP_t *iopmp)
01026 {
01027     return iopmp->err_resp_suppress;
01028 }
01029
01038 static inline bool iopmp_get_stall_violation_en(IOPMP_t *iopmp)
01039 {
01040     return iopmp->stall_violation_en;
01041 }
01042
01051 static inline bool iopmp_get_msi_en(IOPMP_t *iopmp)
01052 {
01053     return iopmp->msi_en;
01054 }
01055
01064 static inline bool iopmp_is_mdlck_locked(IOPMP_t *iopmp)
01065 {
01066     return iopmp->mdlck_lock;
01067 }
01068
01077 static inline bool iopmp_is_entrylck_locked(IOPMP_t *iopmp)
01078 {
01079     return iopmp->entrylck_lock;
01080 }
01081
01089 static inline uint32_t iopmp_get_locked_entry_num(IOPMP_t *iopmp)
01090 {
01091     return iopmp->entrylck_f;
01092 }
01093
01101 static inline uint64_t iopmp_err_report_get_addr(IOPMP_ERR_REPORT_t *err_report)
01102 {
01103     return err_report->addr;
01104 }
01105
01113 static inline uint32_t iopmp_err_report_get_rrid(IOPMP_ERR_REPORT_t *err_report)
01114 {
01115     return err_report->rrid;
01116 }
01117
01126 static inline uint32_t iopmp_err_report_get_eid(IOPMP_ERR_REPORT_t *err_report)
01127 {
01128     return err_report->eid;
01129 }
01130
01140 static inline bool iopmp_err_report_is_no_hit(IOPMP_ERR_REPORT_t *err_report)
01141 {
01142     return err_report->etype == IOPMP_ERRINFO_ETYPE_NOT_HIT;
01143 }
01144
01154 static inline bool iopmp_err_report_is_part_hit(IOPMP_ERR_REPORT_t *err_report)
01155 {
01156     return err_report->etype == IOPMP_ERRINFO_ETYPE_PART_HIT;
01157 }
01158
01166 static inline enum iopmp_errinfo_ttype
01167 iopmp_err_report_get_ttype(IOPMP_ERR_REPORT_t *err_report)
```

```
01168 {
01169     return err_report->ttype;
01170 }
01171
01182 static inline bool iopmp_err_report_get_msi_werr(IOPMP_ERR_REPORT_t *err_report)
01183 {
01184     return err_report->msi_werr;
01185 }
01186
01194 static inline enum iopmp_errinfo_etype
01195 iopmp_err_report_get_etype(IOPMP_ERR_REPORT_t *err_report)
01196 {
01197     return err_report->etype;
01198 }
01199
01208 static inline bool iopmp_err_report_get_svc(IOPMP_ERR_REPORT_t *err_report)
01209 {
01210     return err_report->svc;
01211 }
01212
01221 static inline uint64_t iopmp_entry_get_addr(IOPMP_Entry_t *entry)
01222 {
01223     return entry->addr;
01224 }
01225
01234 static inline uint32_t iopmp_entry_get_cfg(IOPMP_Entry_t *entry)
01235 {
01236     return entry->cfg;
01237 }
01238
01239 /*****************************************************************************/
01240 /* API for IOPMP                                                             */
01241 /*****************************************************************************/
01255 enum iopmp_error iopmp_init(IOPMP_t *iopmp, uintptr_t addr, uint8_t srcmd_fmt,
01256                             uint8_t mdcfg_fmt, uint32_t impid);
01257
01269 enum iopmp_error iopmp_get_vendor_id(IOPMP_t *iopmp, uint32_t *vendor);
01270
01283 enum iopmp_error iopmp_get_specver(IOPMP_t *iopmp, uint32_t *specver);
01284
01296 enum iopmp_error iopmp_get_impid(IOPMP_t *iopmp, uint32_t *impid);
01297
01307 enum iopmp_error iopmp_lock_prio_entry_num(IOPMP_t *iopmp);
01308
01319 enum iopmp_error iopmp_lock_rrid_transl(IOPMP_t *iopmp);
01320
01330 enum iopmp_error iopmp_set_enable(IOPMP_t *iopmp);
01331
01345 enum iopmp_error iopmp_set_prio_entry_num(IOPMP_t *iopmp, uint16_t *num_entry);
01346
01360 enum iopmp_error iopmp_set_rrid_transl(IOPMP_t *iopmp, uint16_t *rrid_transl);
01361
01380 enum iopmp_error iopmp_stall_transactions_by_mds(IOPMP_t *iopmp, uint64_t *mds,
01381                                                  bool exempt, bool polling);
01382
01398 enum iopmp_error iopmp_resume_transactions(IOPMP_t *iopmp, bool polling);
01399
01413 enum iopmp_error iopmp_transactions_are_stalled(IOPMP_t *iopmp, bool polling);
01414
01428 enum iopmp_error iopmp_transactions_are_resumed(IOPMP_t *iopmp, bool polling);
01429
01452 enum iopmp_error iopmp_stall_cherry_pick_rrid(IOPMP_t *iopmp, uint32_t *rrid,
01453                                               bool select,
01454                                               enum iopmp_rridscp_stat *stat);
01455
01476 enum iopmp_error iopmp_query_stall_stat_by_rrid(IOPMP_t *iopmp, uint32_t *rrid,
01477                                                 enum iopmp_rridscp_stat *stat);
01478
01489 enum iopmp_error iopmp_get_locked_md(IOPMP_t *iopmp, uint64_t *mds,
01490                                      bool *mdlck_lock);
01491
01510 enum iopmp_error iopmp_lock_md(IOPMP_t *iopmp, uint64_t *mds, bool mdlck_lock);
01511
01528 enum iopmp_error iopmp_lock_mdcfg(IOPMP_t *iopmp, uint32_t *md_num, bool lock);
01529
01541 enum iopmp_error iopmp_is_mdcfglck_locked(IOPMP_t *iopmp, bool *locked);
01542
01554 enum iopmp_error iopmp_get_locked_mdcfg_num(IOPMP_t *iopmp, uint32_t *md_num);
01555
01574 enum iopmp_error iopmp_lock_entries(IOPMP_t *iopmp, uint32_t *entry_num,
01575                                     bool lock);
01583 enum iopmp_error iopmp_lock_err_cfg(IOPMP_t *iopmp);
01584
01594 enum iopmp_error iopmp_set_global_intr(IOPMP_t *iopmp, bool enable);
01595
01611 enum iopmp_error iopmp_set_global_err_resp(IOPMP_t *iopmp, bool *suppress);
01612
```

```
01626 enum iopmp_error iopmp_set_msi_en(IOPMP_t *iopmp, bool *enable);
01627
01643 enum iopmp_error iopmp_get_msi_addr(IOPMP_t *iopmp, uint64_t *msiaddr64);
01644
01655 enum iopmp_error iopmp_get_msi_data(IOPMP_t *iopmp, uint16_t *msidata);
01656
01675 enum iopmp_error iopmp_set_msi_info(IOPMP_t *iopmp, uint64_t *msiaddr64,
01676                                     uint16_t *msidata);
01677
01688 enum iopmp_error iopmp_get_and_clear_msi_werr(IOPMP_t *iopmp, bool *msi_werr);
01689
01702 enum iopmp_error iopmp_set_stall_violation_en(IOPMP_t *iopmp, bool *enable);
01703
01713 enum iopmp_error iopmp_invalidate_error(IOPMP_t *iopmp);
01714
01727 enum iopmp_error iopmp_capture_error(IOPMP_t *iopmp,
01728                                      IOPMP_ERR_REPORT_t *err_report,
01729                                      bool invalidate);
01730
01748 enum iopmp_error iopmp_mfr_get_sv_window(IOPMP_t *iopmp, uint16_t *svi,
01749                                          uint16_t *svw);
01750
01764 enum iopmp_error iopmp_lock_srcmd_table_fmt_0(IOPMP_t *iopmp, uint32_t rrid);
01765
01781 enum iopmp_error iopmp_is_srcmd_table_fmt_0_locked(IOPMP_t *iopmp,
01782                                                    uint32_t rrid,
01783                                                    bool *locked);
01784
01798 enum iopmp_error iopmp_lock_srcmd_table_fmt_2(IOPMP_t *iopmp, uint32_t mdidx);
01799
01814 enum iopmp_error iopmp_is_srcmd_table_fmt_2_locked(IOPMP_t *iopmp,
01815                                                    uint32_t mdidx,
01816                                                    bool *locked);
01817
01830 enum iopmp_error iopmp_get_rrid_md_association(IOPMP_t *iopmp, uint32_t rrid,
01831                                                uint64_t *mds, bool *lock);
01832
01854 enum iopmp_error iopmp_set_rrid_md_association(IOPMP_t *iopmp, uint32_t rrid,
01855                                                uint64_t mds_set,
01856                                                uint64_t mds_clr,
01857                                                uint64_t *mds,
01858                                                bool lock);
01859
01879 enum iopmp_error iopmp_set_md_permission(IOPMP_t *iopmp, uint32_t rrid,
01880                                          uint32_t mdidx, bool *r, bool *w);
01881
01898 enum iopmp_error iopmp_set_md_permission_multi(IOPMP_t *iopmp, uint32_t mdidx,
01899                                                IOPMP_SRCMD_PERM_CFG_t *cfg);
01900
01915 enum iopmp_error iopmp_set_srcmd_perm_cfg(IOPMP_SRCMD_PERM_CFG_t *cfg,
01916                                           uint32_t rrid, bool r, bool w);
01917
01929 void iopmp_set_srcmd_perm_cfg_nocheck(IOPMP_SRCMD_PERM_CFG_t *cfg,
01930                                       uint32_t rrid, bool r, bool w);
01931
01949 enum iopmp_error iopmp_sps_set_rrid_md_read(IOPMP_t *iopmp, uint32_t rrid,
01950                                             uint64_t mds_set,
01951                                             uint64_t mds_clr,
01952                                             uint64_t *mds);
01953
01966 enum iopmp_error iopmp_sps_get_rrid_md_read(IOPMP_t *iopmp, uint32_t rrid,
01967                                             uint64_t *mds);
01968
01986 enum iopmp_error iopmp_sps_set_rrid_md_write(IOPMP_t *iopmp, uint32_t rrid,
01987                                              uint64_t mds_set,
01988                                              uint64_t mds_clr,
01989                                              uint64_t *mds);
01990
02003 enum iopmp_error iopmp_sps_get_rrid_md_write(IOPMP_t *iopmp, uint32_t rrid,
02004                                              uint64_t *mds);
02005
02028 enum iopmp_error iopmp_sps_set_rrid_md_rw(IOPMP_t *iopmp, uint32_t rrid,
02029                                           uint64_t mds_set_r,
02030                                           uint64_t mds_clr_r,
02031                                           uint64_t mds_set_w,
02032                                           uint64_t mds_clr_w,
02033                                           uint64_t *mds_r,
02034                                           uint64_t *mds_w);
02035
02049 enum iopmp_error iopmp_sps_get_rrid_md_rw(IOPMP_t *iopmp, uint32_t rrid,
02050                                           uint64_t *mds_r, uint64_t *mds_w);
02051
02066 enum iopmp_error iopmp_get_md_entry_association(IOPMP_t *iopmp, uint32_t mdidx,
02067                                                 uint32_t *entry_idx_start,
02068                                                 uint32_t *num_entry);
02069
02091 enum iopmp_error iopmp_set_md_entry_association_multi(IOPMP_t *iopmp,
```

```
02092                                                   uint32_t mdidx_start,
02093                                                   uint32_t *num_entries,
02094                                                   uint32_t md_num);
02095
02116 static inline
02117 enum iopmp_error iopmp_set_md_entry_association(IOPMP_t *iopmp, uint32_t mdidx,
02118                                                 uint32_t *num_entry)
02119 {
02120     return iopmp_set_md_entry_association_multi(iopmp, mdidx, num_entry, 1);
02121 }
02122
02132 enum iopmp_error iopmp_get_md_entry_num(IOPMP_t *iopmp, uint32_t *md_entry_num);
02133
02151 enum iopmp_error iopmp_set_md_entry_num(IOPMP_t *iopmp, uint32_t *md_entry_num);
02152
02205 enum iopmp_error iopmp_encode_entry(IOPMP_t *iopmp, struct iopmp_entry *entries,
02206                                     uint32_t num_entry, uint64_t addr,
02207                                     uint64_t size,
02208                                     enum iopmp_entry_flags flags,
02209                                     uint64_t private_data);
02210
02229 enum iopmp_error iopmp_set_entries_to_md(IOPMP_t *iopmp, uint32_t mdidx,
02230                                         const struct iopmp_entry *entry_array,
02231                                         uint32_t idx_start,
02232                                         uint32_t num_entry);
02233
02251 static inline
02252 enum iopmp_error iopmp_set_entry_to_md(IOPMP_t *iopmp, uint32_t mdidx,
02253                                       const struct iopmp_entry *entry,
02254                                       uint32_t idx)
02255 {
02256     return iopmp_set_entries_to_md(iopmp, mdidx, entry, idx, 1);
02257 }
02258
02274 enum iopmp_error iopmp_get_entries_from_md(IOPMP_t *iopmp, uint32_t mdidx,
02275                                           struct iopmp_entry *entry_array,
02276                                           uint32_t idx_start,
02277                                           uint32_t num_entry);
02278
02293 static inline
02294 enum iopmp_error iopmp_get_entry_from_md(IOPMP_t *iopmp, uint32_t mdidx,
02295                                         struct iopmp_entry *entry,
02296                                         uint32_t idx)
02297 {
02298     return iopmp_get_entries_from_md(iopmp, mdidx, entry, idx, 1);
02299 }
02300
02315 enum iopmp_error iopmp_get_entries(IOPMP_t *iopmp,
02316                                   struct iopmp_entry *entry_array,
02317                                   uint32_t idx_start, uint32_t num_entry);
02318
02332 static inline
02333 enum iopmp_error iopmp_get_entry(IOPMP_t *iopmp, struct iopmp_entry *entry,
02334                                 uint32_t idx)
02335 {
02336     return iopmp_get_entries(iopmp, entry, idx, 1);
02337 }
02338
02356 enum iopmp_error iopmp_set_entries(IOPMP_t *iopmp,
02357                                   const struct iopmp_entry *entry_array,
02358                                   uint32_t idx_start, uint32_t num_entry);
02359
02376 static inline
02377 enum iopmp_error iopmp_set_entry(IOPMP_t *iopmp,
02378                                 const struct iopmp_entry *entry,
02379                                 uint32_t idx)
02380 {
02381     return iopmp_set_entries(iopmp, entry, idx, 1);
02382 }
02383
02395 enum iopmp_error iopmp_clear_entries_in_md(IOPMP_t *iopmp, uint32_t mdidx);
02396
02410 enum iopmp_error iopmp_clear_entries(IOPMP_t *iopmp, uint32_t idx_start,
02411                                     uint32_t num_entry);
02412
02424 static inline enum iopmp_error iopmp_clear_entry(IOPMP_t *iopmp, uint32_t idx)
02425 {
02426     return iopmp_clear_entries(iopmp, idx, 1);
02427 }
02428
02442 enum iopmp_error iopmp_entries_get_belong_md(IOPMP_t *iopmp, uint32_t idx_start,
02443                                             uint32_t num_entry, uint64_t *mds);
02444
02445 #endif
```

5.3 README.md File Reference

Index