

# **LIBIOPMP**

Generated by Doxygen 1.14.0



---

<b>1 libiopmp - A Library to Program RISC-V IOPMP</b>	<b>1</b>
1.1 Adjust config.mk . . . . .	1
1.2 Compilation . . . . .	2
1.2.1 Compiled by Host GCC . . . . .	2
1.2.2 Compiled by RISC-V toolchain . . . . .	2
1.3 Usage . . . . .	2
1.4 Documentation . . . . .	3
<b>2 Data Structure Index</b>	<b>5</b>
2.1 Data Structures . . . . .	5
<b>3 File Index</b>	<b>7</b>
3.1 File List . . . . .	7
<b>4 Data Structure Documentation</b>	<b>9</b>
4.1 iopmp_entry Struct Reference . . . . .	9
4.1.1 Detailed Description . . . . .	9
4.1.2 Field Documentation . . . . .	10
4.1.2.1 addrl . . . . .	10
4.1.2.2 addrh . . . . .	10
4.1.2.3 addr . . . . .	10
4.1.2.4 [union] . . . . .	10
4.1.2.5 r . . . . .	10
4.1.2.6 w . . . . .	10
4.1.2.7 x . . . . .	10
4.1.2.8 a . . . . .	11
4.1.2.9 sire . . . . .	11
4.1.2.10 siwe . . . . .	11
4.1.2.11 sixe . . . . .	11
4.1.2.12 sere . . . . .	11
4.1.2.13 sewe . . . . .	11
4.1.2.14 sexe . . . . .	11
4.1.2.15 rsv . . . . .	11
4.1.2.16 cfg . . . . .	12
4.1.2.17 [union] . . . . .	12
4.1.2.18 prient_flag . . . . .	12
4.1.2.19 private_data . . . . .	12
4.2 iopmp_err_report Struct Reference . . . . .	12
4.2.1 Detailed Description . . . . .	13
4.2.2 Field Documentation . . . . .	13
4.2.2.1 addr . . . . .	13
4.2.2.2 rrid . . . . .	13
4.2.2.3 eid . . . . .	13

4.2.2.4 <a href="#">type</a>	13
4.2.2.5 <a href="#">etype</a>	13
4.2.2.6 <a href="#">msi_werr</a>	13
4.2.2.7 <a href="#">svc</a>	14
4.3 <a href="#">iopmp_instance</a> Struct Reference	14
4.3.1 <a href="#">Detailed Description</a>	15
4.3.2 <a href="#">Field Documentation</a>	15
4.3.2.1 <a href="#">addr</a>	15
4.3.2.2 <a href="#">granularity</a>	15
4.3.2.3 <a href="#">entry_addr_bits</a>	15
4.3.2.4 <a href="#">ops_generic</a>	15
4.3.2.5 <a href="#">ops_specific</a>	16
4.3.2.6 <a href="#">ops_sps</a>	16
4.3.2.7 <a href="#">addr_entry_array</a>	16
4.3.2.8 <a href="#">vendor</a>	16
4.3.2.9 <a href="#">impid</a>	16
4.3.2.10 <a href="#">rrid_num</a>	16
4.3.2.11 <a href="#">entry_num</a>	16
4.3.2.12 <a href="#">prio_entry_num</a>	16
4.3.2.13 <a href="#">rrid_transl</a>	17
4.3.2.14 <a href="#">specver</a>	17
4.3.2.15 <a href="#">md_num</a>	17
4.3.2.16 <a href="#">md_entry_num</a>	17
4.3.2.17 <a href="#">mdlck_lock</a>	17
4.3.2.18 <a href="#">mdlck_md</a>	17
4.3.2.19 <a href="#">mdcfglck_lock</a>	17
4.3.2.20 <a href="#">mdcfglck_f</a>	17
4.3.2.21 <a href="#">entrylck_lock</a>	18
4.3.2.22 <a href="#">entrylck_f</a>	18
4.3.2.23 <a href="#">msiaddr64</a>	18
4.3.2.24 <a href="#">msidata</a>	18
4.3.2.25 <a href="#">init</a>	18
4.3.2.26 <a href="#">mdcfg_fmt</a>	18
4.3.2.27 <a href="#">srcmd_fmt</a>	18
4.3.2.28 <a href="#">no_err_rec</a>	19
4.3.2.29 <a href="#">tor_en</a>	19
4.3.2.30 <a href="#">sps_en</a>	19
4.3.2.31 <a href="#">prio_ent_prog</a>	19
4.3.2.32 <a href="#">non_prio_en</a>	19
4.3.2.33 <a href="#">rrid_transl_en</a>	19
4.3.2.34 <a href="#">rrid_transl_prog</a>	19
4.3.2.35 <a href="#">xinr</a>	19

4.3.2.36 no_x . . . . .	20
4.3.2.37 no_w . . . . .	20
4.3.2.38 stall_en . . . . .	20
4.3.2.39 peis . . . . .	20
4.3.2.40 pees . . . . .	20
4.3.2.41 mfr_en . . . . .	20
4.3.2.42 addrh_en . . . . .	20
4.3.2.43 enable . . . . .	20
4.3.2.44 err_cfg_lock . . . . .	21
4.3.2.45 intr_enable . . . . .	21
4.3.2.46 err_resp_suppress . . . . .	21
4.3.2.47 msi_en . . . . .	21
4.3.2.48 msi_sel . . . . .	21
4.3.2.49 stallViolation_en . . . . .	21
4.3.2.50 support_stall_by_rrid . . . . .	21
4.3.2.51 support_stall_by_md . . . . .	21
4.3.2.52 is_stalling . . . . .	22
4.3.2.53 [struct] . . . . .	22
<b>4.4 iopmp_srcmd_perm_config Struct Reference . . . . .</b>	<b>22</b>
<b>4.4.1 Detailed Description . . . . .</b>	<b>22</b>
<b>4.4.2 Field Documentation . . . . .</b>	<b>22</b>
<b>4.4.2.1 srcmd_perm_mask . . . . .</b>	<b>22</b>
<b>4.4.2.2 srcmd_perm_val . . . . .</b>	<b>22</b>
<b>5 File Documentation . . . . .</b>	<b>23</b>
<b>5.1 libiopmp.h File Reference . . . . .</b>	<b>23</b>
<b>5.1.1 Macro Definition Documentation . . . . .</b>	<b>29</b>
<b>5.1.1.1 IOPMP_MAX_RRID_SRCMD_FMT_2 . . . . .</b>	<b>29</b>
<b>5.1.1.2 IOPMP_SRCMD_PERM_R . . . . .</b>	<b>29</b>
<b>5.1.1.3 IOPMP_SRCMD_PERM_W . . . . .</b>	<b>30</b>
<b>5.1.1.4 IOPMP_SRCMD_PERM_MASK . . . . .</b>	<b>30</b>
<b>5.1.1.5 IOPMP_SRCMD_PERM_CFG_SET_DIRECT . . . . .</b>	<b>30</b>
<b>5.1.1.6 LIBIOPMP_VERSION_MAJOR . . . . .</b>	<b>30</b>
<b>5.1.1.7 LIBIOPMP_VERSION_MINOR . . . . .</b>	<b>30</b>
<b>5.1.1.8 LIBIOPMP_VERSION_EXTRA . . . . .</b>	<b>30</b>
<b>5.1.1.9 LIBIOPMP_VERSION_MAJOR_SHIFT . . . . .</b>	<b>31</b>
<b>5.1.1.10 LIBIOPMP_VERSION_MAJOR_MASK . . . . .</b>	<b>31</b>
<b>5.1.1.11 LIBIOPMP_VERSION_MINOR_SHIFT . . . . .</b>	<b>31</b>
<b>5.1.1.12 LIBIOPMP_VERSION_MINOR_MASK . . . . .</b>	<b>31</b>
<b>5.1.1.13 LIBIOPMP_VERSION_EXTRA_SHIFT . . . . .</b>	<b>31</b>
<b>5.1.1.14 LIBIOPMP_VERSION_EXTRA_MASK . . . . .</b>	<b>31</b>
<b>5.1.1.15 LIBIOPMP_VERSION . . . . .</b>	<b>31</b>

---

5.1.2 Typedef Documentation . . . . .	32
5.1.2.1 IOPMP_t . . . . .	32
5.1.2.2 IOPMP_Entry_t . . . . .	32
5.1.2.3 IOPMP_ERR_REPORT_t . . . . .	32
5.1.2.4 IOPMP_SRCMD_PERM_CFG_t . . . . .	32
5.1.3 Enumeration Type Documentation . . . . .	32
5.1.3.1 iopmp_prient_flags . . . . .	32
5.1.3.2 iopmp_errinfo_ttype . . . . .	33
5.1.3.3 iopmp_errinfo_etype . . . . .	33
5.1.3.4 iopmp_impid . . . . .	33
5.1.3.5 iopmp_srcmd_fmt . . . . .	34
5.1.3.6 iopmp_mdcfg_fmt . . . . .	34
5.1.3.7 iopmp_model . . . . .	34
5.1.3.8 iopmp_rridscp_op . . . . .	35
5.1.3.9 iopmp_rridscp_stat . . . . .	35
5.1.3.10 iopmp_entry_flags . . . . .	35
5.1.3.11 iopmp_error . . . . .	36
5.1.4 Function Documentation . . . . .	36
5.1.4.1 libiopmp_major_version() . . . . .	36
5.1.4.2 libiopmp_minor_version() . . . . .	36
5.1.4.3 libiopmp_extra_version() . . . . .	37
5.1.4.4 libiopmp_check_version() . . . . .	37
5.1.4.5 iopmp_is_initialized() . . . . .	37
5.1.4.6 iopmp_get_base_addr() . . . . .	37
5.1.4.7 iopmp_get_base_addr_entry_array() . . . . .	38
5.1.4.8 iopmp_get_granularity() . . . . .	38
5.1.4.9 iopmp_get_mdcfg_fmt() . . . . .	38
5.1.4.10 iopmp_get_srcmd_fmt() . . . . .	39
5.1.4.11 iopmp_get_support_tor() . . . . .	39
5.1.4.12 iopmp_get_support_sps() . . . . .	39
5.1.4.13 iopmp_get_support_programmable_prio_entry() . . . . .	40
5.1.4.14 iopmp_get_support_rrid_transl() . . . . .	40
5.1.4.15 iopmp_get_support_chk_x() . . . . .	40
5.1.4.16 iopmp_get_no_x() . . . . .	41
5.1.4.17 iopmp_get_no_w() . . . . .	41
5.1.4.18 iopmp_get_support_stall() . . . . .	41
5.1.4.19 iopmp_get_support_peis() . . . . .	42
5.1.4.20 iopmp_get_support_pees() . . . . .	42
5.1.4.21 iopmp_get_support_mfr() . . . . .	42
5.1.4.22 iopmp_get_md_num() . . . . .	43
5.1.4.23 iopmp_get_addrh_en() . . . . .	43
5.1.4.24 iopmp_get_enable() . . . . .	43

---

5.1.4.25 iopmp_get_rrid_num()	44
5.1.4.26 iopmp_get_entry_num()	44
5.1.4.27 iopmp_get_prio_entry_num()	44
5.1.4.28 iopmp_get_support_stall_by_md()	45
5.1.4.29 iopmp_get_support_stall_by_rrid()	45
5.1.4.30 iopmp_is_err_cfg_locked()	45
5.1.4.31 iopmp_get_global_intr()	46
5.1.4.32 iopmp_get_global_err_resp()	46
5.1.4.33 iopmp_get_stallViolation_en()	46
5.1.4.34 iopmp_get_msi_sel()	47
5.1.4.35 iopmp_is_mdlock_locked()	47
5.1.4.36 iopmp_is_entrylock_locked()	47
5.1.4.37 iopmp_get_locked_entry_num()	48
5.1.4.38 iopmp_err_report_get_addr()	48
5.1.4.39 iopmp_err_report_get_rrid()	48
5.1.4.40 iopmp_err_report_get_eid()	49
5.1.4.41 iopmp_err_report_is_no_hit()	49
5.1.4.42 iopmp_err_report_is_part_hit()	49
5.1.4.43 iopmp_err_report_get_ttype()	50
5.1.4.44 iopmp_err_report_get_msi_werr()	50
5.1.4.45 iopmp_err_report_get_etype()	50
5.1.4.46 iopmp_err_report_get_svc()	51
5.1.4.47 iopmp_entry_get_addr()	51
5.1.4.48 iopmp_entry_get_cfg()	51
5.1.4.49 iopmp_init()	52
5.1.4.50 iopmp_get_vendor_id()	52
5.1.4.51 iopmp_get_specver()	53
5.1.4.52 iopmp_get_impid()	54
5.1.4.53 iopmp_lock_prio_entry_num()	54
5.1.4.54 iopmp_lock_rrid_transl()	54
5.1.4.55 iopmp_set_enable()	55
5.1.4.56 iopmp_set_prio_entry_num()	55
5.1.4.57 iopmp_get_rrid_transl_prog()	55
5.1.4.58 iopmp_get_rrid_transl()	56
5.1.4.59 iopmp_set_rrid_transl()	56
5.1.4.60 iopmp_stall_transactions_by_mds()	57
5.1.4.61 iopmp_resume_transactions()	57
5.1.4.62 iopmp_transactions_are_stalled()	57
5.1.4.63 iopmp_transactions_are_resumed()	58
5.1.4.64 iopmp_stall_cherry_pick_rrid()	58
5.1.4.65 iopmp_query_stall_stat_by_rrid()	59
5.1.4.66 iopmp_get_locked_md()	59

---

5.1.4.67 iopmp_lock_md()	60
5.1.4.68 iopmp_lock_mdcfg()	60
5.1.4.69 iopmp_is_mdcfglock_locked()	61
5.1.4.70 iopmp_get_locked_mdcfg_num()	61
5.1.4.71 iopmp_lock_entries()	61
5.1.4.72 iopmp_lock_err_cfg()	62
5.1.4.73 iopmp_set_global_intr()	62
5.1.4.74 iopmp_set_global_err_resp()	62
5.1.4.75 iopmp_set_msi_sel()	63
5.1.4.76 iopmp_get_msi_addr()	63
5.1.4.77 iopmp_get_msi_data()	64
5.1.4.78 iopmp_set_msi_info()	64
5.1.4.79 iopmp_get_and_clear_msi_werr()	64
5.1.4.80 iopmp_set_stallViolation_en()	65
5.1.4.81 iopmp_invalidate_error()	65
5.1.4.82 iopmp_capture_error()	65
5.1.4.83 iopmp_mfr_get_sv_window()	66
5.1.4.84 iopmp_lock_srcmd_table_fmt_0()	66
5.1.4.85 iopmp_is_srcmd_table_fmt_0_locked()	67
5.1.4.86 iopmp_lock_srcmd_table_fmt_2()	67
5.1.4.87 iopmp_is_srcmd_table_fmt_2_locked()	68
5.1.4.88 iopmp_get_rrid_md_association()	68
5.1.4.89 iopmp_set_rrid_md_association()	69
5.1.4.90 iopmp_set_md_permission()	69
5.1.4.91 iopmp_set_md_permission_multi()	70
5.1.4.92 iopmp_set_srcmd_perm_cfg()	70
5.1.4.93 iopmp_set_srcmd_perm_cfg_nocheck()	71
5.1.4.94 iopmp_sps_set_rrid_md_read()	71
5.1.4.95 iopmp_sps_get_rrid_md_read()	72
5.1.4.96 iopmp_sps_set_rrid_md_write()	72
5.1.4.97 iopmp_sps_get_rrid_md_write()	72
5.1.4.98 iopmp_sps_set_rrid_insn_fetch()	73
5.1.4.99 iopmp_sps_get_rrid_md_insn_fetch()	73
5.1.4.100 iopmp_sps_set_rrid_md_rwx()	74
5.1.4.101 iopmp_sps_get_rrid_md_rwx()	74
5.1.4.102 iopmp_get_md_entry_association()	75
5.1.4.103 iopmp_set_md_entry_association_multi()	75
5.1.4.104 iopmp_set_md_entry_association()	76
5.1.4.105 iopmp_get_md_entry_num()	77
5.1.4.106 iopmp_set_md_entry_num()	77
5.1.4.107 iopmp_encode_entry()	78
5.1.4.108 iopmp_set_entries_to_md()	79

5.1.4.109 iopmp_set_entry_to_md()	80
5.1.4.110 iopmp_get_entries_from_md()	80
5.1.4.111 iopmp_get_entry_from_md()	81
5.1.4.112 iopmp_get_entries()	82
5.1.4.113 iopmp_get_entry()	82
5.1.4.114 iopmp_set_entries()	83
5.1.4.115 iopmp_set_entry()	84
5.1.4.116 iopmp_clear_entries_in_md()	84
5.1.4.117 iopmp_clear_entries()	85
5.1.4.118 iopmp_clear_entry()	85
5.1.4.119 iopmp_entries_get_belong_md()	86
5.2 libiopmp.h	86
5.3 README.md File Reference	96
<b>Index</b>	<b>97</b>



# 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 **v0.8.2, 2026**
- 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. We describe each of the configurations here:

- 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:

iopmp_entry . . . . .	9
iopmp_err_report . . . . .	12
iopmp_instance . . . . .	14
iopmp_srcmd_perm_config	
Configuration used in srcmd_fmt=2 to set SRCMD_PERM(H) . . . . .	22



# **Chapter 3**

## **File Index**

### **3.1 File List**

Here is a list of all files with brief descriptions:

<a href="#">libiopmp.h</a> . . . . .	23
--------------------------------------	----



# 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 **sexr**: 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.sex
```

**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 HWCFG3.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 mdlock_lock`
- `uint64_t mdlock_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 srcomd_fmt: 2`
  - `unsigned int no_err_rec: 1`
  - `unsigned int tor_en: 1`
  - `unsigned int sps_en: 1`
  - `unsigned int prio_ent_prog: 1`
  - `unsigned int non_prio_en: 1`
  - `unsigned int rrid_transl_en: 1`
  - `unsigned int rrid_transl_prog: 1`
  - `unsigned int xinr: 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 msi_sel: 1
unsigned int stallViolation_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.I

**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.I

**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.I

#### 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 no\_err\_rec**

```
unsigned int no_err_rec
```

Flag to indicate if the Error Capture Record is not implemented

**4.3.2.29 tor\_en**

```
unsigned int tor_en
```

Flag to indicate if TOR is supported

**4.3.2.30 sps\_en**

```
unsigned int sps_en
```

Flag to indicate SPS(secondary permission settings) is supported

**4.3.2.31 prio\_ent\_prog**

```
unsigned int prio_ent_prog
```

Flag to indicates if HWCFG2.prio\_entry is programmable

**4.3.2.32 non\_prio\_en**

```
unsigned int non_prio_en
```

Flag to indicates whether the IOPMP supports non-priority entries

**4.3.2.33 rrid\_transl\_en**

```
unsigned int rrid_transl_en
```

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

**4.3.2.34 rrid\_transl\_prog**

```
unsigned int rrid_transl_prog
```

Flag to indicate if the field HWCFG3.rrid\_transl is programmable

**4.3.2.35 xinr**

```
unsigned int xinr
```

Flag to indicate if the IOPMP treats the instruction fetch accesses as data read accesses

**4.3.2.36 no\_x**

```
unsigned int no_x
```

Flag to indicate for xinr=0, the IOPMP with no\_x=1 always fails on an instruction fetch

**4.3.2.37 no\_w**

```
unsigned int no_w
```

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

**4.3.2.38 stall\_en**

```
unsigned int stall_en
```

Flag to indicate if the IOPMP implements stall-related features

**4.3.2.39 peis**

```
unsigned int peis
```

Flag to indicate if the IOPMP implements interrupt suppression per entry

**4.3.2.40 pees**

```
unsigned int pees
```

Flag to indicate if the IOPMP implements error suppression per entry

**4.3.2.41 mfr\_en**

```
unsigned int mfr_en
```

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

**4.3.2.42 addrh\_en**

```
unsigned int addrh_en
```

Flag to indicate if registers ENTRY\_ADDRH(i) and ERR\_MSIADDRH (if HWCFG2.msi\_en = 1) are available

**4.3.2.43 enable**

```
unsigned int enable
```

Indicate if the IOPMP checks transactions

**4.3.2.44 err\_cfg\_lock**

```
unsigned int err_cfg_lock
```

Lock fields to ERR\_CFG register

**4.3.2.45 intr\_enable**

```
unsigned int intr_enable
```

Enable the global interrupt of the IOPMP

**4.3.2.46 err\_resp\_suppress**

```
unsigned int err_resp_suppress
```

Suppress the global error responses of the IOPMP

**4.3.2.47 msi\_en**

```
unsigned int msi_en
```

Flag to indicate whether the IOPMP supports MSI extension

**4.3.2.48 msi\_sel**

```
unsigned int msi_sel
```

Flag to indicate whether the IOPMP triggers MSI or wired interrupt

**4.3.2.49 stallViolation\_en**

```
unsigned int stallViolation_en
```

Flag to indicate whether the IOPMP faults stalled transactions

**4.3.2.50 supportStallByRrid**

```
unsigned int supportStallByRrid
```

Flag to indicate if stall by RRID is supported

**4.3.2.51 supportStallByMd**

```
unsigned int supportStallByMd
```

Flag to indicate if stall by MD is supported

### 4.3.2.52 `is_stalling`

```
unsigned int is_stalling
```

Flag to indicate if IOPMP is stalling some transactions

### 4.3.2.53 [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 HWCFG3.mdcfg\_fmt of the IOPMP.*
- `static enum iopmp_srcmd_fmt iopmp_get_srcmd_fmt (IOPMP_t *iopmp)`  
*Get HWCFG3.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_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_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 HWCFG2.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_stallViolation_en` (`IOPMP_t` \*`iopmp`)  
*Check if the IOPMP faults stalled transactions.*
- static bool `iopmp_get_msi_sel` (`IOPMP_t` \*`iopmp`)  
*Check if the IOPMP triggers interrupt by MSI.*
- static bool `iopmp_is_mdclk_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`)

- static bool `iopmp_err_report_is_no_hit` (`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_part_hit` (`IOPMP_ERR_REPORT_t` \*`err_report`)
 

*Check if the type of violation is "not hit any 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 initial 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 HWCFG2.prio\_ent\_prog=1.*
- enum `iopmp_error` `iopmp_lock_rrid_transl` (`IOPMP_t` \*`iopmp`)
 

*Lock the RRID tagged to outgoing transactions if the IOPMP HWCFG3.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_get_rrid_transl_prog` (`IOPMP_t` \*`iopmp`, `bool` \*`rrid_transl_prog`)
 

*Check if HWCFG3.rrid\_transl is programmable.*
- enum `iopmp_error` `iopmp_get_rrid_transl` (`IOPMP_t` \*`iopmp`, `uint16_t` \*`rrid_transl`)
 

*Get the RRID tagged to outgoing transactions.*
- 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`)
 

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

- 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.I.*
  - 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 MDCFGLOCK 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 MDCFGLOCK.*
  - 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_sel (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_stallViolation_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\\_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\\_insn\\_fetch](#) ([IOPMP\\_t](#) \*iopmp, [uint32\\_t](#) rrid, [uint64\\_t](#) mds\_set, [uint64\\_t](#) mds\_clr, [uint64\\_t](#) \*mds)
 

*(SPS only) Set RRID's instruction fetch permission to MDs*
  - enum [iopmp\\_error iopmp\\_sps\\_get\\_rrid\\_md\\_insn\\_fetch](#) ([IOPMP\\_t](#) \*iopmp, [uint32\\_t](#) rrid, [uint64\\_t](#) \*mds)
 

*(SPS only) Get RRID's instruction fetch permission to MDs*
  - enum [iopmp\\_error iopmp\\_sps\\_set\\_rrid\\_md\\_rwx](#) ([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\_set\_x, [uint64\\_t](#) mds\_clr\_x, [uint64\\_t](#) mds\_r, [uint64\\_t](#) mds\_w, [uint64\\_t](#) mds\_x)
 

*(SPS only) Set RRID's read/write/instruction fetch permission to MDs*
  - enum [iopmp\\_error iopmp\\_sps\\_get\\_rrid\\_md\\_rwx](#) ([IOPMP\\_t](#) \*iopmp, [uint32\\_t](#) rrid, [uint64\\_t](#) \*mds\_r, [uint64\\_t](#) \*mds\_w, [uint64\\_t](#) \*mds\_x)
 

*(SPS only) Get RRID's read/write/instruction fetch 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 HWCFG3.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 HWCFG3.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	<i>cfg</i>	pointer to struct <code>iopmp_srcmd_perm_config</code>
in	<i>mask</i>	Desired value of <code>srcmd_perm_mask</code>
in	<i>val</i>	Desired value of <code>srcmd_perm_val</code>

### 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	<code>__major</code>	The major version
in	<code>__minor</code>	The minor version
in	<code>__extra</code>	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. HWCFG3.md_entry_num is fixed
IOPMP_MDCFG_FMT_2	Format 2. No MDCFG Table. HWCFG3.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_TOR   IOPMP_ENTRY_PRIO   IOPMP_ENTRY_NON_PRIO)

### 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	<i>major</i>	The major version
in	<i>minor</i>	The minor version
in	<i>extra</i>	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	<i>iopmp</i>	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	<i>iopmp</i>	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	<i>iopmp</i>	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	<i>iopmp</i>	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_mdconfig_fmt (
    IOPMP_t * iopmp) [inline], [static]
```

Get HWCFG3.mdcfg\_fmt of the IOPMP.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
----	--------------	------------------------------

**Returns**

HWCFG3.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 HWCFG3.srcmd\_fmt of the IOPMP.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
----	--------------	------------------------------

**Returns**

HWCFG3.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	<i>iopmp</i>	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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG2.sps_en = 1 and the SPS operations are implemented
0	if HWCFG2.sps_en = 0

**5.1.4.13 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG2.prio_ent_prog = 1
0	if HWCFG2.prio_ent_prog = 0

**5.1.4.14 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG3.rrid_transl_en = 1
0	if HWCFG3.rrid_transl_en = 0

**5.1.4.15 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**

<code>in</code>	<code>iopmp</code>	The IOPMP instance to be checked
-----------------	--------------------	----------------------------------

**Return values**

<code>1</code>	if HWCFG3.xinx = 0
<code>0</code>	if HWCFG3.xinx = 1

**5.1.4.16 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**

<code>in</code>	<code>iopmp</code>	The IOPMP instance to be checked
-----------------	--------------------	----------------------------------

**Return values**

<code>1</code>	if HWCFG3.no_x = 1
<code>0</code>	if HWCFG3.no_x = 0

**5.1.4.17 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 no rule matched.

**Parameters**

<code>in</code>	<code>iopmp</code>	The IOPMP instance to be checked
-----------------	--------------------	----------------------------------

**Return values**

<code>1</code>	if HWCFG3.no_w = 1
<code>0</code>	if HWCFG3.no_w = 0

**5.1.4.18 iopmp\_get\_support\_stall()**

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

Check if the IOPMP implements stall-related features.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG2.stall_en = 1
0	if HWCFG2.stall_en = 0

**5.1.4.19 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG2.peis = 1
0	if HWCFG2.peis = 0

**5.1.4.20 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG2.pees = 1
0	if HWCFG2.pees = 0

**5.1.4.21 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if HWCFG2.mfr_en = 1
0	if HWCFG2.mfr_en = 0

**5.1.4.22 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	<i>iopmp</i>	The IOPMP instance to be got
----	--------------	------------------------------

**Returns**

HWCFG0.md\_num

**5.1.4.23 iopmp\_get\_addrh\_en()**

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

Check if ENTRY\_ADDRH(i) and ERR\_MSIADDRH (if HWCFG2.msi\_en = 1) are available.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

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

**5.1.4.24 iopmp\_get\_enable()**

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

Check if the IOPMP checks transactions.

**Parameters**

<code>in</code>	<code>iopmp</code>	The IOPMP instance to be checked
-----------------	--------------------	----------------------------------

**Return values**

<code>1</code>	if HWCFG0.enable = 1
<code>0</code>	if HWCFG0.enable = 0

**5.1.4.25 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**

<code>in</code>	<code>iopmp</code>	The IOPMP instance to be got
-----------------	--------------------	------------------------------

**Returns**

`HWCFG1.rrid_num`

**5.1.4.26 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**

<code>in</code>	<code>iopmp</code>	The IOPMP instance to be got
-----------------	--------------------	------------------------------

**Returns**

`HWCFG1.entry_num`

**5.1.4.27 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	<i>iopmp</i>	The IOPMP instance to be got
----	--------------	------------------------------

**Returns**

HWCFG2.prio\_entry

**5.1.4.28 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

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

**5.1.4.29 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if RRIDSCP is implemented
0	if RRIDSCP is not implemented

**5.1.4.30 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if ERR_CFG.I = 1
0	if ERR_CFG.I = 0

**5.1.4.31 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

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

**5.1.4.32 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	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

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

**5.1.4.33 iopmp\_get\_stallViolation\_en()**

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

Check if the IOPMP faults stalled transactions.

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

## Return values

1	if ERR_CFG.stallViolation_en = 1
0	if ERR_CFG.stallViolation_en = 0

**5.1.4.34 iopmp\_get\_msi\_sel()**

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

Check if the IOPMP triggers interrupt by MSI.

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

## Return values

1	if ERR_CFG.msi_sel = 1
0	if ERR_CFG.msi_sel = 0

**5.1.4.35 iopmp\_is\_mdlock\_locked()**

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

Check if MDLCK register has been locked.

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

## Return values

1	if MDLCK.I = 1
0	if MDLCK.I = 0

**5.1.4.36 iopmp\_is\_entrylck\_locked()**

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

Check if ENTRYLCK register has been locked.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be checked
----	--------------	----------------------------------

**Return values**

1	if ENTRYLCK.I = 1
0	if ENTRYLCK.I = 0

**5.1.4.37 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	<i>iopmp</i>	The IOPMP instance to be got
----	--------------	------------------------------

**Returns**

ENTRYLCK.f

**5.1.4.38 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	<i>err_report</i>	The pointer to the error report
----	-------------------	---------------------------------

**Returns**

Errored address[65:2]

**5.1.4.39 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	<i>err_report</i>	The pointer to the error report
----	-------------------	---------------------------------

**Returns**

Errored RRID

**5.1.4.40 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	<i>err_report</i>	The pointer to the error report
----	-------------------	---------------------------------

**Returns**

The index pointing to the entry that catches the violation

**5.1.4.41 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	<i>err_report</i>	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"

**5.1.4.42 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	<i>err_report</i>	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"

**5.1.4.43 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	<i>err_report</i>	The pointer to the error report
----	-------------------	---------------------------------

**Returns**

The transaction type

**5.1.4.44 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	<i>err_report</i>	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

**5.1.4.45 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**

<code>in</code>	<code>err_report</code>	The pointer to the error report
-----------------	-------------------------	---------------------------------

**Returns**

The type of violation

**5.1.4.46 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**

<code>in</code>	<code>err_report</code>	The pointer to the error report
-----------------	-------------------------	---------------------------------

**Return values**

<code>1</code>	if there is a subsequent violation caught in ERR_MFR
<code>0</code>	if there is no subsequent violation

**5.1.4.47 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**

<code>in</code>	<code>entry</code>	The pointer to the IOPMP entry structure
-----------------	--------------------	--

**Returns**

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

**5.1.4.48 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	<i>entry</i>	The pointer to the IOPMP entry structure
----	--------------	--

**Returns**

The permissions and attributes of protected memory region

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

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if some features are not supported

**5.1.4.50 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	<i>iopmp</i>	The IOPMP instance to be got
out	<i>vendor</i>	Pointer to integer to store vendor ID

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given vendor is NULL
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support get the vendor ID of IOPMP

#### 5.1.4.51 iopmp\_get\_specver()

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

Get the specification version of the IOPMP.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
out	<i>specver</i>	Pointer to integer to store specification version

**Return values**

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

**5.1.4.52 iopmp\_get\_impid()**

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

Get the implementation ID of the IOPMP.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
out	<i>impid</i>	Pointer to integer to store implementation ID

**Return values**

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

**5.1.4.53 iopmp\_lock\_prio\_entry\_num()**

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

Lock number of priority entry if the IOPMP HWCFG2.prio\_ent\_prog=1.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance
----	--------------	--------------------

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support non-priority entries

**5.1.4.54 iopmp\_lock\_rrid\_transl()**

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

Lock the RRID tagged to outgoing transactions if the IOPMP HWCFG3.rrid\_transl\_prog=1.

## Parameters

in	<i>iopmp</i>	The IOPMP instance
----	--------------	--------------------

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support lock the RRID tagged to outgoing transactions

**5.1.4.55 iopmp\_set\_enable()**

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

Enable the IOPMP checker.

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be set
----	--------------	------------------------------

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support enable the checker

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

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support non-priority entries
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if HWCFG2.prio_ent_prog is 0
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if <i>num_entry</i> is NULL
<i>IOPMP_ERR_ILLEGAL_VALUE</i>	if the written <i>num_entry</i> does not match the actual value. The actual value is output via

**5.1.4.57 iopmp\_get\_rrid\_transl\_prog()**

```
enum iopmp_error iopmp_get_rrid_transl_prog (
    IOPMP_t * iopmp,
    bool * rrid_transl_prog)
```

Check if HWCFG3.rrid\_transl is programmable.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
out	<i>rrid_transl_prog</i>	Output true if HWCFG3.rrid_transl is programmable. Otherwise output false.

**Return values**

<i>IOPMP_OK</i>	if HWCFG3.rrid_transl_en is 1
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if HWCFG3.rrid_transl_en is 0
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>rrid_transl_prog</i> is NULL

**5.1.4.58 iopmp\_get\_rrid\_transl()**

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

Get the RRID tagged to outgoing transactions.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
out	<i>rrid_transl</i>	Output the RRID tagged to outgoing transactions

**Return values**

<i>IOPMP_OK</i>	if HWCFG3.rrid_transl_en is 1
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if HWCFG3.rrid_transl_en is 0
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>rrid_transl</i> is NULL

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

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if HWCFG3.rrid_transl_en is 0
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if HWCFG3.rrid_transl_prog is 0
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>rrid_transl</i> is NULL
<i>IOPMP_ERR_ILLEGAL_VALUE</i>	if the written <i>rrid_transl</i> does not match the actual value. The actual value is output

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

#### Return values

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

### 5.1.4.61 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	<i>iopmp</i>	The IOPMP instance to be resumed
in	<i>polling</i>	Set true to poll the resume status until resuming takes effect

#### Return values

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

### 5.1.4.62 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	<i>iopmp</i>	The IOPMP instance to be checked
in	<i>polling</i>	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
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support stall
<i>IOPMP_ERR_NOT_EXIST</i>	if <i>iopmp</i> did not stall any transactions by <a href="#">iopmp_stall_transactions_by_mds()</a>

**5.1.4.63 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	<i>iopmp</i>	The IOPMP instance to be checked
in	<i>polling</i>	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
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support stall
<i>IOPMP_ERR_NOT_EXIST</i>	if <i>iopmp</i> did not resume any transactions by <a href="#">iopmp_resume_transactions()</a>

**5.1.4.64 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	<i>iopmp</i>	The IOPMP instance to be set
in, out	<i>rrid</i>	Input the RRID to be stalled. Output WARL value
in	<i>select</i>	Set true select or false to deselect
out	<i>stat</i>	The pointer to store enum <a href="#">iopmp_rridscp_stat</a>

## Return values

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

## 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\(\)](#).

5.1.4.65 **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	<i>iopmp</i>	The IOPMP instance to be queried
in, out	<i>rrid</i>	Input the RRID to be queried. Output WARL value
out	<i>stat</i>	The pointer to store enum <a href="#">iopmp_rridscp_stat</a>

## Return values

<i>Positive</i>	value for stall status of <i>rrid</i>
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>rrid</i> is out of bounds
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>rrid</i> or <i>stat</i> is NULL
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> 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\(\)](#).

5.1.4.66 **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.I.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be set
out	<i>mds</i>	Pointer to integer to store locked MD bitmap
out	<i>mdlck_lock</i>	Pointer to integer to store MDLCK.I

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>mds</i> or <i>mdlck_lock</i> is NULL

**5.1.4.67 iopmp\_lock\_md()**

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

Lock MDs.

**Parameters**

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

**Return values**

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

**Note**

If MDLCK.I has already been set to 1, this API always expects *mdlck\_lock* be 1.

**5.1.4.68 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	<i>iopmp</i>	The IOPMP instance to be set
in,out	<i>md_num</i>	Input the number of MD to be locked. Output WARL value
in	<i>lock</i>	Set 1 to lock MDCFGLOCK register

## Return values

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

**5.1.4.69 iopmp\_is\_mdcfglck\_locked()**

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

Check if MDCFGLOCK was locked.

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be checked
out	<i>locked</i>	The pointer to an integer to store MDCFGLOCK.f

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if <i>locked</i> is NULL
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not implement MDCFG table format 0

**5.1.4.70 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 MDCFGLOCK.

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be checked
out	<i>md_num</i>	The pointer to an integer to store MDCFGLOCK.f

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if <i>md_num</i> is NULL
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not implement MDCFG table format 0

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

**Return values**

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

**5.1.4.72 iopmp\_lock\_err\_cfg()**

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

Lock fields of ERR\_CFG register.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be set
----	--------------	------------------------------

**Returns**

*IOPMP\_OK*

**5.1.4.73 iopmp\_set\_global\_intr()**

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

Enable/Disable global interrupt.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be set
in	<i>enable</i>	1(enable) or 0(disable)

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if ERR_CFG register is locked

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

## Return values

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

**5.1.4.75 iopmp\_set\_msi\_sel()**

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

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

## Parameters

in	<i>iopmp</i>	The IOPMP instance to be set
in, out	<i>enable</i>	True to enable or false to disable. Output WARL value

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support MSI
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if ERR_CFG.I=1
<i>IOPMP_ERR_ILLEGAL_VALUE</i>	if the written value do not match the actual value

**5.1.4.76 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	<i>iopmp</i>	The IOPMP instance to be got
out	<i>msiaddr64</i>	Pointer to 64-bit integer to store MSI address

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support MSI
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>msiaddr64</i> 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

### 5.1.4.77 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	<i>iopmp</i>	The IOPMP instance to be got
out	<i>msidata</i>	Pointer to 16-bit integer to store MSI data

#### Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support MSI
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>msidata</i> is NULL

### 5.1.4.78 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	<i>iopmp</i>	The IOPMP instance to be set
in, out	<i>msiaddr64</i>	Input 64-bit MSI address. Output WARL value
in, out	<i>msidata</i>	Input 11-bit MSI data. Output WARL value

#### Return values

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

### 5.1.4.79 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	<i>iopmp</i>	The IOPMP instance to be checked
out	<i>msi_werr</i>	The pointer to flag

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>msi_werr</i> is NULL
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support MSI

**5.1.4.80 iopmp\_set\_stallViolation\_en()**

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

Enable or disable the IOPMP faults stalled transactions.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be set
in, out	<i>enable</i>	Input 1 to enable, 0 to disable. Output WARL value

**Return values**

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

**5.1.4.81 iopmp\_invalidate\_error()**

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

Invalidate the error record by clearing ERR\_INFO.v bit.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be invalidated
----	--------------	--------------------------------------

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> does not support clear error interrupt pending bit

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

**Return values**

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

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

**Return values**

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

**5.1.4.84 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	<i>iopmp</i>	The IOPMP instance to be set
in	<i>rrid</i>	The RRID to be locked

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> SRCMD_FMT!=0
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>rrid</i> is out of bounds

## Note

This operation is only supported by SRCMD\_FMT=0

**5.1.4.85 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	<i>iopmp</i>	The IOPMP instance to be got
in	<i>rrid</i>	The RRID to be got
out	<i>locked</i>	The pointer to an integer to store SRCMD_EN.I

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp</i> SRCMD_FMT!=0
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>rrid</i> is out of bounds
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>locked</i> is NULL

## Note

This operation is only supported by SRCMD\_FMT=0

**5.1.4.86 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	<i>iopmp</i>	The IOPMP instance to be set
in	<i>mdidx</i>	The index of MD to be locked

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp_SRCMD_FMT!=2</i>
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>rrid</i> is out of bounds
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if MDLCK has been locked

## Note

This operation is only supported by SRCMD\_FMT=2

**5.1.4.87 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

<i>in</i>	<i>iopmp</i>	The IOPMP instance to be got
<i>in</i>	<i>mdidx</i>	The index of MD to be got
<i>out</i>	<i>locked</i>	The pointer to an integer to store SRCMD_EN.I

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if <i>iopmp_SRCMD_FMT!=2</i>
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>mdidx</i> is out of bounds
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>locked</i> is NULL

## Note

This operation is only supported by SRCMD\_FMT=2

**5.1.4.88 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

<i>in</i>	<i>iopmp</i>	The IOPMP instance to be got
<i>in</i>	<i>rrid</i>	The RRID to be got
<i>out</i>	<i>mds</i>	The pointer to an integer to store SRCMD_EN.md
<i>out</i>	<i>lock</i>	The pointer to an integer to store SRCMD_EN.I

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>rrid</i> is out of bounds
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>lock</i> or <i>md</i> is NULL

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

## Return values

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

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

## Return values

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

**5.1.4.91 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	<i>iopmp</i>	The IOPMP instance to be set
in	<i>mdidx</i>	The desired MD to be given permission
in	<i>cfg</i>	The configuration structure for SRCMD format 2

## Return values

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

**5.1.4.92 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	<i>cfg</i>	Pointer to struct <i>iopmp_srcmd_perm_config</i>
in	<i>rrid</i>	Desired RRID to be set
in	<i>r</i>	Set true to give RRID read permission; false to clear read permission
in	<i>w</i>	Set true to give RRID write permission; false to clear write permission

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>cfg</i> is NULL
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>rrid</i> is out of bounds

**5.1.4.93 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	<i>cfg</i>	Pointer to struct <a href="#">iopmp_srcmd_perm_config</a>
in	<i>rrid</i>	Desired RRID to be set
in	<i>r</i>	Set true to give RRID read permission; false to clear read permission
in	<i>w</i>	Set true to give RRID write permission; false to clear write permission

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

## Return values

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

#### 5.1.4.95 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	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be checked
out	<i>mds</i>	Pointer to variable to output permission

##### Return values

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

#### 5.1.4.96 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	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be set
in	<i>mds_set</i>	The desired MDs to set permission to <i>rrid</i>
in	<i>mds_clr</i>	The desired MDs to clear permission to <i>rrid</i>
out	<i>mds</i>	The pointer to an integer to store WARL value of SRCMD_W.md after setting

##### Return values

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

#### 5.1.4.97 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	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be checked
out	<i>mds</i>	Pointer to variable to output permission

## Return values

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

**5.1.4.98 iopmp\_sps\_set\_rrid\_insn\_fetch()**

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

(SPS only) Set RRID's instruction fetch permission to MDs

## Parameters

in	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be set
in	<i>mds_set</i>	The desired MDs to set permission to <i>rrid</i>
in	<i>mds_clr</i>	The desired MDs to clear permission to <i>rrid</i>
out	<i>mds</i>	The pointer to an integer to store WARL value of SRCMD_X.md after setting

## Return values

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

**5.1.4.99 iopmp\_sps\_get\_rrid\_md\_insn\_fetch()**

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

(SPS only) Get RRID's instruction fetch permission to MDs

## Parameters

in	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be checked
out	<i>mds</i>	Pointer to variable to output permission

## Return values

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

**5.1.4.100 iopmp\_sps\_set\_rrid\_md\_rwx()**

```
enum iopmp_error iopmp_sps_set_rrid_md_rwx (
    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_set_x,
    uint64_t mds_clr_x,
    uint64_t * mds_r,
    uint64_t * mds_w,
    uint64_t * mds_x)
```

(SPS only) Set RRID's read/write/instruction fetch permission to MDs

## Parameters

in	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be set
in	<i>mds_set_r</i>	The desired MDs to set R permission to <i>rrid</i>
in	<i>mds_clr_r</i>	The desired MDs to clear R permission to <i>rrid</i>
in	<i>mds_set_w</i>	The desired MDs to set W permission to <i>rrid</i>
in	<i>mds_clr_w</i>	The desired MDs to clear W permission to <i>rrid</i>
in	<i>mds_set_x</i>	The desired MDs to set X permission to <i>rrid</i>
in	<i>mds_clr_x</i>	The desired MDs to clear X permission to <i>rrid</i>
out	<i>mds_r</i>	The pointer to an integer to store WARL value of SRCMD_R.md after setting
out	<i>mds_w</i>	The pointer to an integer to store WARL value of SRCMD_W.md after setting
out	<i>mds_x</i>	The pointer to an integer to store WARL value of SRCMD_X.md after setting

## Return values

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

**5.1.4.101 iopmp\_sps\_get\_rrid\_md\_rwx()**

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

(SPS only) Get RRID's read/write/instruction fetch permission to multiple MDs

## Parameters

in	<i>iopmp</i>	The IOPMP instance
in	<i>rrid</i>	The RRID to be set
out	<i>mds_r</i>	Pointer to variable to output read permission
out	<i>mds_w</i>	Pointer to variable to output write permission
out	<i>mds_x</i>	Pointer to variable to output instruction fetch permission

## Return values

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

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

## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>mdidx</i> is out of bounds
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>entry_idx_start</i> or <i>num_entry</i> is NULL

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

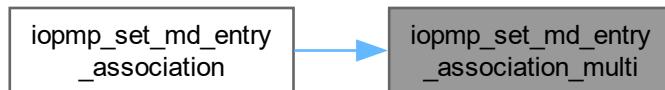
## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_ALLOWED</i>	if <i>iopmp</i> MDCFG format is not 0
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>num_entries</i> is NULL
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>mdidx_start</i> or <i>md_num</i> is out of bounds
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if MDCFG of given <i>mdidx_start</i> has been locked by MDCFGLOCK.f
<i>IOPMP_ERR_ILLEGAL_VALUE</i>	if the written <i>num_entries</i> 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:



#### 5.1.4.104 *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	<i>iopmp</i>	The IOPMP instance to be set
in	<i>mdidx</i>	The index of target MD
in,out	<i>num_entry</i>	Input the number of entries to be associated. Output actual number of entries be associated.

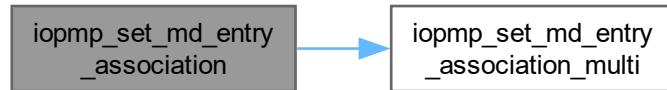
## Return values

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_ALLOWED</i>	if <i>iopmp</i> MDCFG format is not 0
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>num_entry</i> is NULL
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>mdidx</i> or <i>num_entry</i> is out of bounds
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if MDCFG of given <i>mdidx</i> has been locked by MDCFGLOCK.f
<i>IOPMP_ERR_ILLEGAL_VALUE</i>	if the written <i>num_entry</i> 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 call graph for this function:

**5.1.4.105 iopmp\_get\_md\_entry\_num()**

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

Get value of HWCFG3.md\_entry\_num if IOPMP model is xxx-K.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be got
out	<i>md_entry_num</i>	The pointer to an integer to return value

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>md_entry_num</i> is NULL

**5.1.4.106 iopmp\_set\_md\_entry\_num()**

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

Program value of HWCFG3.md\_entry\_num.

**Parameters**

in	<i>iopmp</i>	The IOPMP instance to be programmed
in, out	<i>md_entry_num</i>	Input the drsired value of <i>md_entry_num</i> . Output WARL value.

**Return values**

<i>IOPMP_OK</i>	if successes
<i>IOPMP_ERR_NOT_ALLOWED</i>	if IOPMP MDCFG format is not 2

<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <code>md_entry_num</code> is NULL
<i>IOPMP_ERR_REG_IS_LOCKED</i>	if IOPMP has been enabled
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <code>md_entry_num</code> is out-of-bounds
<i>IOPMP_ERR_ILLEGAL_VALUE</i>	if the written <code>md_entry_num</code> does not match the actual value. The actual value is output.

**Note**

This function must be called only when IOPMP's MDCFG\_FMT=2

**5.1.4.107 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	<i>iopmp</i>	The IOPMP instance
out	<i>entries</i>	The array of entry to be output
in	<i>num_entry</i>	Number of entries in <code>entries</code>
in	<i>addr</i>	Address of the memory region
in	<i>size</i>	Size of the memory region
in	<i>flags</i>	Flags of the entry for this memory region
in	<i>private_data</i>	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
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <code>entries</code> is NULL or <code>num_entry</code> is 0 or <code>size</code> is 0 or <code>addr</code> or <code>size</code> is not aligned
<i>IOPMP_ERR_NOT_SUPPORTED</i>	if memory region should be encoded as TOR entry, but <code>iopmp</code> does not support TOR entry
<i>IOPMP_ERR_NOT_ALLOWED</i>	if memory region should be encoded as TOR, but only one entry in given <code>entries</code> ; 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 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 HWCFG3.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.

#### 5.1.4.108 `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	<code>iopmp</code>	The IOPMP instance to be written
in	<code>mdidx</code>	The index of target MD
in	<code>entry_array</code>	The array of entries
in	<code>idx_start</code>	The local index of entry in target MD
in	<code>num_entry</code>	The number of entries to be set

##### Return values

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

Here is the caller graph for this function:



### 5.1.4.109 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	<i>iopmp</i>	The IOPMP instance to be written
in	<i>mdidx</i>	The index of target MD
in	<i>entry</i>	The pointer to the entry
in	<i>idx</i>	The local index of entry in target MD

#### Return values

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

Here is the call graph for this function:



### 5.1.4.110 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	<i>iopmp</i>	The IOPMP instance to be read
in	<i>mdidx</i>	The index of target MD
out	<i>entry_array</i>	The array of entries
in	<i>idx_start</i>	The local start index of entries in target MD
in	<i>num_entry</i>	The number of entries to be read

## Return values

<i>IOPMP_OK</i>	on success
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>entry_array</i> is NULL or <i>num_entry</i> is 0
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>mdidx</i> , <i>idx_start</i> or <i>num_entry</i> is out of bounds

Here is the caller graph for this function:

**5.1.4.111 *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	<i>iopmp</i>	The IOPMP instance to be read
in	<i>mdidx</i>	The index of target MD
out	<i>entry</i>	The pointer to the entry
in	<i>idx</i>	The local start index of entries in target MD

## Return values

<i>IOPMP_OK</i>	on success
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>entry_array</i> is NULL or <i>num_entry</i> is 0
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>mdidx</i> , <i>idx_start</i> or <i>num_entry</i> is out of bounds

Here is the call graph for this function:



### 5.1.4.112 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	<i>iopmp</i>	The IOPMP instance to be read
out	<i>entry_array</i>	The array of entries
in	<i>idx_start</i>	The global start index of target entries
in	<i>num_entry</i>	The number of entries to be read

#### Return values

<i>IOPMP_OK</i>	on success
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>entry_array</i> is NULL or <i>num_entry</i> is 0
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>idx_start</i> or <i>num_entry</i> is out of bounds

Here is the caller graph for this function:



### 5.1.4.113 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	<i>iopmp</i>	The IOPMP instance to be read
out	<i>entry</i>	The pointer to the entry
in	<i>idx</i>	The global start index of target entries

## Return values

<i>IOPMP_OK</i>	on success
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>entry_array</i> is NULL or <i>num_entry</i> is 0
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>idx_start</i> or <i>num_entry</i> is out of bounds

Here is the call graph for this function:

**5.1.4.114 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	<i>iopmp</i>	The IOPMP instance to be written
in	<i>entry_array</i>	The array of entries
in	<i>idx_start</i>	The global start index of target entries
in	<i>num_entry</i>	The number of entries to be written

## Return values

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

Here is the caller graph for this function:



### 5.1.4.115 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	<i>iopmp</i>	The IOPMP instance to be written
in	<i>entry</i>	The pointer to the entry
in	<i>idx</i>	The global start index of target entries

#### Return values

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

Here is the call graph for this function:



### 5.1.4.116 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	<i>iopmp</i>	The IOPMP instance
in	<i>mdidx</i>	The index of target MD

#### Return values

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

### 5.1.4.117 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	<i>iopmp</i>	The IOPMP instance
in	<i>idx_start</i>	The global start index of target entries
in	<i>num_entry</i>	The number of entries to be cleared

#### Return values

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

Here is the caller graph for this function:



### 5.1.4.118 iopmp\_clear\_entry()

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

Clear single global entry.

#### Parameters

in	<i>iopmp</i>	The IOPMP instance
in	<i>idx</i>	The global index of target entry

#### Return values

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

Here is the call graph for this function:



#### 5.1.4.119 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	<i>iopmp</i>	The IOPMP instance
in	<i>idx_start</i>	The global start index of target entries
in	<i>num_entry</i>	The number of entries to be checked
out	<i>mds</i>	Pointer to integer to store MD bitmap

##### Return values

<i>IOPMP_OK</i>	on success
<i>IOPMP_ERR_OUT_OF_BOUNDS</i>	if given <i>idx_start</i> or <i>num_entry</i> is out of bounds
<i>IOPMP_ERR_INVALID_PARAMETER</i>	if given <i>mds</i> 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 /***** libiopmp data structure. ****/
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 mdclk_lock;
00070     uint64_t mdclk_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 msicaddr64;
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 no_err_rec : 1;
00100        unsigned int tor_en : 1;
00102        unsigned int sps_en : 1;
00104        unsigned int prio_ent_prog : 1;
00106        unsigned int non_prio_en : 1;
00111        unsigned int rrid_transl_en : 1;
00113        unsigned int rrid_transl_prog : 1;
00118        unsigned int xinxr : 1;
00123        unsigned int no_x : 1;
00128        unsigned int no_w : 1;
00130        unsigned int stall_en : 1;
00135        unsigned int peis : 1;
00139        unsigned int pees : 1;
00144        unsigned int mfr_en : 1;
00149        unsigned int addrh_en : 1;
00151        unsigned int enable : 1;
00153        unsigned int err_cfg_lock : 1;
00155        unsigned int intr_enable : 1;
00157        unsigned int err_resp_suppress : 1;
00159        unsigned int msi_en : 1;
00161        unsigned int msi_sel : 1;
00163        unsigned int stallViolation_en : 1;
00165        unsigned int support_stall_by_rrid : 1;
00167        unsigned int support_stall_by_md : 1;
00169        unsigned int is_stalling : 1;
00170    };
00171 };
00172
00180 enum iopmp_prient_flags {
00182     IOPMP_PRIENT_ANY = 0,
00184     IOPMP_PRIENT_PRIORITY = (1 << 0),
00186     IOPMP_PRIENT_NON_PRIORITY = (1 << 1),
00187 };
00188
00194 struct iopmp_entry {
00196     union {
00197         struct {
00199             uint32_t addr1;
00201             uint32_t addrh;
00202         };
00204         uint64_t addr;
00205     };

```

```

00206
00208     union {
00209         struct {
00211             uint32_t r : 1;
00213             uint32_t w : 1;
00215             uint32_t x : 1;
00217             uint32_t a : 2;
00219             uint32_t siren : 1;
00221             uint32_t siwe : 1;
00223             uint32_t sive : 1;
00225             uint32_t sere : 1;
00227             uint32_t sewe : 1;
00229             uint32_t sexe : 1;
00231             uint32_t rsv : 21;
00232         };
00234     uint32_t cfg;
00235 };
00236
00238 enum iopmp_prient_flags prient_flag;
00239
00248 uint64_t private_data;
00249 };
00250
00252 enum iopmp_errinfo_ttype {
00254     IOPMP_ERRINFO_TTYPE_RSVD      = 0x00,
00256     IOPMP_ERRINFO_TTYPE_READ     = 0x01,
00258     IOPMP_ERRINFO_TTYPE_WRITE    = 0x02,
00260     IOPMP_ERRINFO_TTYPE_INST_FETCH = 0x03,
00261 };
00262
00264 enum iopmp_errinfo_etype {
00266     IOPMP_ERRINFO_ETYPE_NONE      = 0x00,
00268     IOPMP_ERRINFO_ETYPE_READ     = 0x01,
00270     IOPMP_ERRINFO_ETYPE_WRITE    = 0x02,
00272     IOPMP_ERRINFO_ETYPE_INST_FETCH = 0x03,
00274     IOPMP_ERRINFO_ETYPE_PART_HIT = 0x04,
00276     IOPMP_ERRINFO_ETYPE_NOT_HIT  = 0x05,
00278     IOPMP_ERRINFO_ETYPE_UNKNOWN_RRID = 0x06,
00280     IOPMP_ERRINFO_ETYPE_STALL    = 0x07,
00282     IOPMP_ERRINFO_ETYPE_RESERVED_0 = 0x08,
00284     IOPMP_ERRINFO_ETYPE_RESERVED_1 = 0x09,
00286     IOPMP_ERRINFO_ETYPE_RESERVED_2 = 0x0A,
00288     IOPMP_ERRINFO_ETYPE_RESERVED_3 = 0x0B,
00290     IOPMP_ERRINFO_ETYPE_RESERVED_4 = 0x0C,
00292     IOPMP_ERRINFO_ETYPE_RESERVED_5 = 0x0D,
00294     IOPMP_ERRINFO_ETYPE_USER_DEF_0 = 0x0E,
00296     IOPMP_ERRINFO_ETYPE_USER_DEF_1 = 0x0F,
00297 };
00298
00300 struct iopmp_err_report {
00302     uint64_t addr;
00304     uint32_t rrid;
00306     uint32_t eid;
00308     enum iopmp_errinfo_ttype ttype;
00310     enum iopmp_errinfo_etype etype;
00312     bool msi_werr;
00314     bool svc;
00315 };
00316
00317 typedef struct iopmp_instance IOPMP_t;
00318
00319 typedef struct iopmp_entry IOPMP_Entry_t;
00320
00321 typedef struct iopmp_err_report IOPMP_ERR_REPORT_t;
00322
00324 #define IOPMP_MAX_RRID_SRCMD_FMT_2 32
00325
00334 struct iopmp_srcmd_perm_config {
00336 #define IOPMP_SRCMD_PERM_R      (1 << 0)
00338 #define IOPMP_SRCMD_PERM_W      (1 << 1)
00340 #define IOPMP_SRCMD_PERM_MASK   (IOPMP_SRCMD_PERM_W | IOPMP_SRCMD_PERM_R)
00341
00347     uint64_t srcmd_perm_mask;
00348
00355     uint64_t srcmd_perm_val;
00356 };
00357 typedef struct iopmp_srcmd_perm_config IOPMP_SRCMD_PERM_CFG_t;
00358
00366 #define IOPMP_SRCMD_PERM_CFG_SET_DIRECT(cfg, mask, val) \
00367     do { \
00368         IOPMP_SRCMD_PERM_CFG_t *__cfg = (cfg); \
00369         __cfg->srcmd_perm_mask = mask; \
00370         __cfg->srcmd_perm_val = val; \
00371     } while (0);
00372
00373 /***** Supported IOPMP implementation ID *****/
00374 /* Supported IOPMP implementation ID */

```

```

00375 /*****
00377 enum iopmp_impid {
00379   IOPMP_IMPID_NOT_SPECIFIED = 0xFFFFFFFF,
00380 };
00381
00382 /* MDCFG_FMT and SRCMD_FMT and models */
00383 */
00384 */
00385 enum iopmp_srcmd_fmt {
00386   IOPMP_SRCMD_FMT_0,
00387   IOPMP_SRCMD_FMT_1,
00388   IOPMP_SRCMD_FMT_2,
00389   IOPMP_SRCMD_FMT_RESERVED,
00390   IOPMP_SRCMD_FMT_MAX,
00391 };
00392
00393 enum iopmp_mdcfg_fmt {
00394   IOPMP_MDCFG_FMT_0,
00395   IOPMP_MDCFG_FMT_1,
00396   IOPMP_MDCFG_FMT_2,
00397   IOPMP_MDCFG_FMT_RESERVED,
00398   IOPMP_MDCFG_FMT_MAX,
00399 };
00400
00401 enum iopmp_model {
00402   IOPMP_MODEL_FULL      = 0,
00403   IOPMP_MODEL_RAPID_K    = 1,
00404   IOPMP_MODEL_DYNAMIC_K  = 2,
00405   IOPMP_MODEL_RESERVED_3 = 3,
00406   IOPMP_MODEL_ISOLATION  = 4,
00407   IOPMP_MODEL_COMPACT_K  = 5,
00408   IOPMP_MODEL_6           = 6,
00409   IOPMP_MODEL_RESERVED_7 = 7,
00410   IOPMP_MODEL_RESERVED_8 = 8,
00411   IOPMP_MODEL_RESERVED_9 = 9,
00412   IOPMP_MODEL_RESERVED_10 = 10,
00413   IOPMP_MODEL_RESERVED_11 = 11,
00414   IOPMP_MODEL_RESERVED_12 = 12,
00415   IOPMP_MODEL_RESERVED_13 = 13,
00416   IOPMP_MODEL_RESERVED_14 = 14,
00417   IOPMP_MODEL_RESERVED_15 = 15
00418 };
00419
00420 enum iopmp_rridscp_op {
00421   IOPMP_RRIDSCP_OP_QUERY = 0,
00422   IOPMP_RRIDSCP_OP_STALL = 1,
00423   IOPMP_RRIDSCP_OP_DONT_STALL = 2,
00424   IOPMP_RRIDSCP_OP_RESERVED = 3
00425 };
00426
00427 enum iopmp_rridscp_stat {
00428   IOPMP_RRIDSCP_STAT_NOT_IMPL = 0,
00429   IOPMP_RRIDSCP_STAT_STALLED = 1,
00430   IOPMP_RRIDSCP_STAT_NOT_STALLED = 2,
00431   IOPMP_RRIDSCP_STAT_ERR_RRID = 3
00432 };
00433
00434 /*****
00435 /* The flags used when calling iopmp_encode_entry() */
00436 */
00437 */
00438 enum iopmp_entry_flags {
00439   IOPMP_ENTRY_R = (1UL << 0),
00440   IOPMP_ENTRY_W = (1UL << 1),
00441   IOPMP_ENTRY_X = (1UL << 2),
00442   IOPMP_ENTRY_RW = (IOPMP_ENTRY_R | IOPMP_ENTRY_W),
00443   IOPMP_ENTRY_RX = (IOPMP_ENTRY_R | IOPMP_ENTRY_X),
00444   IOPMP_ENTRY_RWX = (IOPMP_ENTRY_R | IOPMP_ENTRY_W | IOPMP_ENTRY_X),
00445
00446   IOPMP_ENTRY_A_OFF = (0UL << 3),
00447   IOPMP_ENTRY_A_TOR = (1UL << 3),
00448   IOPMP_ENTRY_A_NA4 = (2UL << 3),
00449   IOPMP_ENTRY_A_NAPOT = (3UL << 3),
00450   IOPMP_ENTRY_A_MASK = (3UL << 3),
00451
00452   IOPMP_ENTRY_SIRE = (1UL << 5),
00453   IOPMP_ENTRY_SIWE = (1UL << 6),
00454   IOPMP_ENTRY_SIXE = (1UL << 7),
00455   IOPMP_ENTRY_SIE_MASK = (7UL << 5),
00456   IOPMP_ENTRY_SERE = (1UL << 8),
00457   IOPMP_ENTRY_SEWE = (1UL << 9),
00458   IOPMP_ENTRY_SEXE = (1UL << 10),
00459   IOPMP_ENTRY_SEE_MASK = (7UL << 8),
00460
00461   IOPMP_ENTRY_FORCE_OFF = (1UL << 27),
00462   IOPMP_ENTRY_FIRST_TOR = (1UL << 28),
00463   IOPMP_ENTRY_FORCE_TOR = (1UL << 29),
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02
```

```

00527     IOPMP_ENTRY_PRIO = (1UL << 30),
00529     IOPMP_ENTRY_NON_PRIO = (1UL << 31),
00530
00532     IOPMP_ENTRY_SW_FLAGS_MASK = (IOPMP_ENTRY_FORCE_OFF | IOPMP_ENTRY_FIRST_TOR |
00533                                     IOPMP_ENTRY_FORCE_TOR | IOPMP_ENTRY_PRIO |
00534                                     IOPMP_ENTRY_NON_PRIO),
00535 };
00536
00537 /***** API Error codes *****/
00538 /* API Error codes */
00539 /*****
00540 enum iopmp_error {
00541     IOPMP_OK              = 0,
00542     IOPMP_ERR_NOT_SUPPORTED = -1,
00543     IOPMP_ERR_OUT_OF_BOUNDS = -2,
00544     IOPMP_ERR_REG_IS_LOCKED = -3,
00545     IOPMP_ERR_NOT_ALLOWED  = -4,
00546     IOPMP_ERR_NOT_EXIST    = -5,
00547     IOPMP_ERR_NOT_AVAILABLE = -6,
00548     IOPMP_ERR_INVALID_PARAMETER = -7,
00549     IOPMP_ERR_INVALID_PRIORITY = -8,
00550     IOPMP_ERR_ILLEGAL_VALUE  = -9,
00551 };
00552
00553 /***** Helper macros and functions to get libiopmp version information *****/
00554 /* Helper macros and functions to get libiopmp version information */
00555 /*****
00556 #define LIBIOPMP_VERSION_MAJOR          0
00557 #define LIBIOPMP_VERSION_MINOR          1
00558 #define LIBIOPMP_VERSION_EXTRA           0
00559
00560 #define LIBIOPMP_VERSION_MAJOR_SHIFT     16
00561 #define LIBIOPMP_VERSION_MAJOR_MASK      0xffff
00562
00563 #define LIBIOPMP_VERSION_MINOR_SHIFT     8
00564 #define LIBIOPMP_VERSION_MINOR_MASK      0xff
00565
00566 #define LIBIOPMP_VERSION_EXTRA_SHIFT     0
00567 #define LIBIOPMP_VERSION_EXTRA_MASK      0xff
00568
00569 #define LIBIOPMP_VERSION(_major, _minor, _extra) \
00570   (((_major) & LIBIOPMP_VERSION_MAJOR_MASK) << LIBIOPMP_VERSION_MAJOR_SHIFT) | \
00571   (((_minor) & LIBIOPMP_VERSION_MINOR_MASK) << LIBIOPMP_VERSION_MINOR_SHIFT) | \
00572   (((_extra) & LIBIOPMP_VERSION_EXTRA_MASK) << LIBIOPMP_VERSION_EXTRA_SHIFT))
00573
00574 int libiopmp_major_version(void);
00575
00576 int libiopmp_minor_version(void);
00577
00578 int libiopmp_extra_version(void);
00579
00580 bool libiopmp_check_version(int major, int minor, int extra);
00581
00582 /***** Helper macros to get/set local variables *****/
00583 /* Helper macros to get/set local variables */
00584 /*****
00585 static inline bool iopmp_is_initialized(IOPMP_t *iopmp)
00586 {
00587     return iopmp && iopmp->init;
00588 }
00589
00590 static inline uintptr_t iopmp_get_base_addr(IOPMP_t *iopmp)
00591 {
00592     return iopmp->addr;
00593 }
00594
00595 static inline uintptr_t iopmp_get_base_addr_entry_array(IOPMP_t *iopmp)
00596 {
00597     return iopmp->addr_entry_array;
00598 }
00599
00600 static inline uint32_t iopmp_get_granularity(IOPMP_t *iopmp)
00601 {
00602     return iopmp->granularity;
00603 }
00604
00605 static inline enum iopmp_mdcfg_fmt iopmp_get_mdcfg_fmt(IOPMP_t *iopmp)
00606 {
00607     return iopmp->mdcfg_fmt;
00608 }
00609
00610 static inline enum iopmp_srcmd_fmt iopmp_get_srcmd_fmt(IOPMP_t *iopmp)
00611 {
00612     return iopmp->srcmd_fmt;
00613 }
00614
00615 static inline bool iopmp_get_support_tor(IOPMP_t *iopmp)
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02178
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209

```

```
00721 {
00722     return iopmp->tor_en;
00723 }
00724
00725 static inline bool iopmp_get_support_sps(IOPMP_t *iopmp)
00726 {
00727     return iopmp->sps_en && iopmp->ops_sps;
00728 }
00729
00730 static inline bool iopmp_get_support_programmable_prio_entry(IOPMP_t *iopmp)
00731 {
00732     return iopmp->prio_ent_prog;
00733 }
00734
00735 static inline bool iopmp_get_support_rrid_transl(IOPMP_t *iopmp)
00736 {
00737     return iopmp->rrid_transl_en;
00738 }
00739
00740 static inline bool iopmp_get_support_chk_x(IOPMP_t *iopmp)
00741 {
00742     return !iopmp->xinx;
00743 }
00744
00745 static inline bool iopmp_get_no_x(IOPMP_t *iopmp)
00746 {
00747     return iopmp->no_x;
00748 }
00749
00750 static inline bool iopmp_get_no_w(IOPMP_t *iopmp)
00751 {
00752     return iopmp->no_w;
00753 }
00754
00755 static inline bool iopmp_get_support_stall(IOPMP_t *iopmp)
00756 {
00757     return iopmp->stall_en;
00758 }
00759
00760 static inline bool iopmp_get_support_peis(IOPMP_t *iopmp)
00761 {
00762     return iopmp->peis;
00763 }
00764
00765 static inline bool iopmp_get_support_pees(IOPMP_t *iopmp)
00766 {
00767     return iopmp->pees;
00768 }
00769
00770 static inline uint32_t iopmp_get_md_num(IOPMP_t *iopmp)
00771 {
00772     return iopmp->md_num;
00773 }
00774
00775 static inline uint32_t iopmp_get_addrh_en(IOPMP_t *iopmp)
00776 {
00777     return iopmp->addrh_en;
00778 }
00779
00780 static inline bool iopmp_get_enable(IOPMP_t *iopmp)
00781 {
00782     return iopmp->enable;
00783 }
00784
00785 static inline uint32_t iopmp_get_rrid_num(IOPMP_t *iopmp)
00786 {
00787     return iopmp->rrid_num;
00788 }
00789
00790 static inline uint32_t iopmp_get_entry_num(IOPMP_t *iopmp)
00791 {
00792     return iopmp->entry_num;
00793 }
00794
00795 static inline uint16_t iopmp_get_prio_entry_num(IOPMP_t *iopmp)
00796 {
00797     return iopmp->prio_entry_num;
00798 }
00799
00800 static inline bool iopmp_get_support_stall_by_md(IOPMP_t *iopmp)
00801 {
00802     return iopmp->support_stall_by_md;
```

```
00942 }
00943
00952 static inline bool iopmp_get_support_stall_by_rrid(IOPMP_t *iopmp)
00953 {
00954     return iopmp->support_stall_by_rrid;
00955 }
00956
00965 static inline bool iopmp_is_err_cfg_locked(IOPMP_t *iopmp)
00966 {
00967     return iopmp->err_cfg_lock;
00968 }
00969
00978 static inline bool iopmp_get_global_intr(IOPMP_t *iopmp)
00979 {
00980     return iopmp->intr_enable;
00981 }
00982
00991 static inline bool iopmp_get_global_err_resp(IOPMP_t *iopmp)
00992 {
00993     return iopmp->err_resp_suppress;
00994 }
00995
01004 static inline bool iopmp_get_stallViolation_en(IOPMP_t *iopmp)
01005 {
01006     return iopmp->stall_violation_en;
01007 }
01008
01017 static inline bool iopmp_get_msi_sel(IOPMP_t *iopmp)
01018 {
01019     return iopmp->msi_sel;
01020 }
01021
01030 static inline bool iopmp_is_mdlock_locked(IOPMP_t *iopmp)
01031 {
01032     return iopmp->mdlock_lock;
01033 }
01034
01043 static inline bool iopmp_is_entrylck_locked(IOPMP_t *iopmp)
01044 {
01045     return iopmp->entrylck_lock;
01046 }
01047
01055 static inline uint32_t iopmp_get_locked_entry_num(IOPMP_t *iopmp)
01056 {
01057     return iopmp->entrylck_f;
01058 }
01059
01067 static inline uint64_t iopmp_err_report_get_addr(IOPMP_ERR_REPORT_t *err_report)
01068 {
01069     return err_report->addr;
01070 }
01071
01079 static inline uint32_t iopmp_err_report_get_rrid(IOPMP_ERR_REPORT_t *err_report)
01080 {
01081     return err_report->rrid;
01082 }
01083
01092 static inline uint32_t iopmp_err_report_get_eid(IOPMP_ERR_REPORT_t *err_report)
01093 {
01094     return err_report->eid;
01095 }
01096
01106 static inline bool iopmp_err_report_is_no_hit(IOPMP_ERR_REPORT_t *err_report)
01107 {
01108     return err_report->etype == IOPMP_ERRINFOETYPE_NOT_HIT;
01109 }
01110
01120 static inline bool iopmp_err_report_is_part_hit(IOPMP_ERR_REPORT_t *err_report)
01121 {
01122     return err_report->etype == IOPMP_ERRINFOETYPE_PART_HIT;
01123 }
01124
01132 static inline enum iopmp_errinfo_ttype
01133 iopmp_err_report_get_ttype(IOPMP_ERR_REPORT_t *err_report)
01134 {
01135     return err_report->ttype;
01136 }
01137
01148 static inline bool iopmp_err_report_get_msi_werr(IOPMP_ERR_REPORT_t *err_report)
01149 {
01150     return err_report->msi_werr;
01151 }
01152
01160 static inline enum iopmp_errinfo_etype
01161 iopmp_err_report_get_etype(IOPMP_ERR_REPORT_t *err_report)
01162 {
01163     return err_report->etype;
```

```
01164 }
01165
01174 static inline bool iopmp_err_report_get_svc(IOPMP_ERR_REPORT_t *err_report)
01175 {
01176     return err_report->svc;
01177 }
01178
01187 static inline uint64_t iopmp_entry_get_addr(IOPMP_Entry_t *entry)
01188 {
01189     return entry->addr;
01190 }
01191
01200 static inline uint32_t iopmp_entry_get_cfg(IOPMP_Entry_t *entry)
01201 {
01202     return entry->cfg;
01203 }
01204
01205 /***** API for IOPMP *****/
01206 /* API for IOPMP */
01207 /***** API for IOPMP *****/
01221 enum iopmp_error iopmp_init(IOPMP_t *iopmp, uintptr_t addr, uint8_t srcmd_fmt,
01222                                 uint8_t mdcfg_fmt, uint32_t impid);
01223
01235 enum iopmp_error iopmp_get_vendor_id(IOPMP_t *iopmp, uint32_t *vendor);
01236
01249 enum iopmp_error iopmp_get_specver(IOPMP_t *iopmp, uint32_t *specver);
01250
01262 enum iopmp_error iopmp_get_impid(IOPMP_t *iopmp, uint32_t *impid);
01263
01273 enum iopmp_error iopmp_lock_prio_entry_num(IOPMP_t *iopmp);
01274
01285 enum iopmp_error iopmp_lock_rrid_transl(IOPMP_t *iopmp);
01286
01296 enum iopmp_error iopmp_set_enable(IOPMP_t *iopmp);
01297
01313 enum iopmp_error iopmp_set_prio_entry_num(IOPMP_t *iopmp, uint16_t *num_entry);
01314
01326 enum iopmp_error iopmp_get_rrid_transl_prog(IOPMP_t *iopmp,
01327                                              bool *rrid_transl_prog);
01328
01339 enum iopmp_error iopmp_get_rrid_transl(IOPMP_t *iopmp, uint16_t *rrid_transl);
01340
01355 enum iopmp_error iopmp_set_rrid_transl(IOPMP_t *iopmp, uint16_t *rrid_transl);
01356
01375 enum iopmp_error iopmp_stall_transactions_by_mds(IOPMP_t *iopmp, uint64_t *mds,
01376                                              bool exempt, bool polling);
01377
01393 enum iopmp_error iopmp_resume_transactions(IOPMP_t *iopmp, bool polling);
01394
01408 enum iopmp_error iopmp_transactions_are_stalled(IOPMP_t *iopmp, bool polling);
01409
01423 enum iopmp_error iopmp_transactions_are_resumed(IOPMP_t *iopmp, bool polling);
01424
01447 enum iopmp_error iopmp_stall_cherry_pick_rrid(IOPMP_t *iopmp, uint32_t *rrid,
01448                                              bool select,
01449                                              enum iopmp_rridscp_stat *stat);
01450
01471 enum iopmp_error iopmp_query_stall_stat_by_rrid(IOPMP_t *iopmp, uint32_t *rrid,
01472                                              enum iopmp_rridscp_stat *stat);
01473
01484 enum iopmp_error iopmp_get_locked_md(IOPMP_t *iopmp, uint64_t *mds,
01485                                              bool *mdlck_lock);
01486
01505 enum iopmp_error iopmp_lock_md(IOPMP_t *iopmp, uint64_t *mds, bool mdlck_lock);
01506
01523 enum iopmp_error iopmp_lock_mdcfg(IOPMP_t *iopmp, uint32_t *md_num, bool lock);
01524
01536 enum iopmp_error iopmp_is_mdcfg_lck_locked(IOPMP_t *iopmp, bool *locked);
01537
01549 enum iopmp_error iopmp_get_locked_mdcfg_num(IOPMP_t *iopmp, uint32_t *md_num);
01550
01569 enum iopmp_error iopmp_lock_entries(IOPMP_t *iopmp, uint32_t *entry_num,
01570                                              bool lock);
01578 enum iopmp_error iopmp_lock_err_cfg(IOPMP_t *iopmp);
01579
01589 enum iopmp_error iopmp_set_global_intr(IOPMP_t *iopmp, bool enable);
01590
01606 enum iopmp_error iopmp_set_global_err_resp(IOPMP_t *iopmp, bool *suppress);
01607
01621 enum iopmp_error iopmp_set_msi_sel(IOPMP_t *iopmp, bool *enable);
01622
01638 enum iopmp_error iopmp_get_msi_addr(IOPMP_t *iopmp, uint64_t *msiaddr64);
01639
01650 enum iopmp_error iopmp_get_msi_data(IOPMP_t *iopmp, uint16_t *msidata);
01651
01670 enum iopmp_error iopmp_set_msi_info(IOPMP_t *iopmp, uint64_t *msiaddr64,
01671                                              uint16_t *msidata);
```

```

01672
01683 enum iopmp_error iopmp_get_and_clear_msi_werr(IOPMP_t *iopmp, bool *msi_werr);
01684
01697 enum iopmp_error iopmp_set_stallViolation_en(IOPMP_t *iopmp, bool *enable);
01698
01708 enum iopmp_error iopmp_invalidate_error(IOPMP_t *iopmp);
01709
01722 enum iopmp_error iopmp_capture_error(IOPMP_t *iopmp,
01723                                     IOPMP_ERR_REPORT_t *err_report,
01724                                     bool invalidate);
01725
01743 enum iopmp_error iopmp_mfr_get_sv_window(IOPMP_t *iopmp, uint16_t *svi,
01744                                         uint16_t *svw);
01745
01759 enum iopmp_error iopmp_lock_srcmd_table_fmt_0(IOPMP_t *iopmp, uint32_t rrid);
01760
01776 enum iopmp_error iopmp_is_srcmd_table_fmt_0_locked(IOPMP_t *iopmp,
01777                                         uint32_t rrid,
01778                                         bool *locked);
01779
01793 enum iopmp_error iopmp_lock_srcmd_table_fmt_2(IOPMP_t *iopmp, uint32_t mdidx);
01794
01809 enum iopmp_error iopmp_is_srcmd_table_fmt_2_locked(IOPMP_t *iopmp,
01810                                         uint32_t mdidx,
01811                                         bool *locked);
01812
01825 enum iopmp_error iopmp_get_rrid_md_association(IOPMP_t *iopmp, uint32_t rrid,
01826                                         uint64_t *mds, bool *lock);
01827
01849 enum iopmp_error iopmp_set_rrid_md_association(IOPMP_t *iopmp, uint32_t rrid,
01850                                         uint64_t mds_set,
01851                                         uint64_t mds_clr,
01852                                         uint64_t *mds,
01853                                         bool lock);
01854
01874 enum iopmp_error iopmp_set_md_permission(IOPMP_t *iopmp, uint32_t rrid,
01875                                         uint32_t mdidx, bool *r, bool *w);
01876
01893 enum iopmp_error iopmp_set_md_permission_multi(IOPMP_t *iopmp, uint32_t mdidx,
01894                                         IOPMP_SRCMD_PERM_CFG_t *cfg);
01895
01910 enum iopmp_error iopmp_set_srcmd_perm_cfg(IOPMP_SRCMD_PERM_CFG_t *cfg,
01911                                         uint32_t rrid, bool r, bool w);
01912
01924 void iopmp_set_srcmd_perm_cfg_nocheck(IOPMP_SRCMD_PERM_CFG_t *cfg,
01925                                         uint32_t rrid, bool r, bool w);
01926
01944 enum iopmp_error iopmp_sps_set_rrid_md_read(IOPMP_t *iopmp, uint32_t rrid,
01945                                         uint64_t mds_set,
01946                                         uint64_t mds_clr,
01947                                         uint64_t *mds);
01948
01961 enum iopmp_error iopmp_sps_get_rrid_md_read(IOPMP_t *iopmp, uint32_t rrid,
01962                                         uint64_t *mds);
01963
01981 enum iopmp_error iopmp_sps_set_rrid_md_write(IOPMP_t *iopmp, uint32_t rrid,
01982                                         uint64_t mds_set,
01983                                         uint64_t mds_clr,
01984                                         uint64_t *mds);
01985
01998 enum iopmp_error iopmp_sps_get_rrid_md_write(IOPMP_t *iopmp, uint32_t rrid,
01999                                         uint64_t *mds);
02000
02018 enum iopmp_error iopmp_sps_set_rrid_insn_fetch(IOPMP_t *iopmp, uint32_t rrid,
02019                                         uint64_t mds_set,
02020                                         uint64_t mds_clr,
02021                                         uint64_t *mds);
02022
02035 enum iopmp_error iopmp_sps_get_rrid_md_insn_fetch(IOPMP_t *iopmp, uint32_t rrid,
02036                                         uint64_t *mds);
02037
02064 enum iopmp_error iopmp_sps_set_rrid_md_rwx(IOPMP_t *iopmp, uint32_t rrid,
02065                                         uint64_t mds_set_r,
02066                                         uint64_t mds_clr_r,
02067                                         uint64_t mds_set_w,
02068                                         uint64_t mds_clr_w,
02069                                         uint64_t mds_set_x,
02070                                         uint64_t mds_clr_x,
02071                                         uint64_t *mds_r,
02072                                         uint64_t *mds_w,
02073                                         uint64_t *mds_x);
02074
02091 enum iopmp_error iopmp_sps_get_rrid_md_rwx(IOPMP_t *iopmp, uint32_t rrid,
02092                                         uint64_t *mds_r, uint64_t *mds_w,
02093                                         uint64_t *mds_x);
02094
02109 enum iopmp_error iopmp_get_md_entry_association(IOPMP_t *iopmp, uint32_t mdidx,

```

```
02110                               uint32_t *entry_idx_start,
02111                               uint32_t *num_entry);
02112
02113 enum iopmp_error iopmp_set_md_entry_association_multi(IOPMP_t *iopmp,
02114                                                               uint32_t mdidx_start,
02115                                                               uint32_t *num_entries,
02116                                                               uint32_t md_num);
02117
02118 static inline
02119 enum iopmp_error iopmp_set_md_entry_association(IOPMP_t *iopmp, uint32_t mdidx,
02120                                                   uint32_t *num_entry)
02121 {
02122     return iopmp_set_md_entry_association_multi(iopmp, mdidx, num_entry, 1);
02123 }
02124
02125 enum iopmp_error iopmp_get_md_entry_num(IOPMP_t *iopmp, uint32_t *md_entry_num);
02126
02127 enum iopmp_error iopmp_set_md_entry_num(IOPMP_t *iopmp, uint32_t *md_entry_num);
02128
02129 enum iopmp_error iopmp_encode_entry(IOPMP_t *iopmp, struct iopmp_entry *entries,
02130                                       uint32_t num_entry, uint64_t addr,
02131                                       uint64_t size,
02132                                       enum iopmp_entry_flags flags,
02133                                       uint64_t private_data);
02134
02135 enum iopmp_error iopmp_set_entries_to_md(IOPMP_t *iopmp, uint32_t mdidx,
02136                                             const struct iopmp_entry *entry_array,
02137                                             uint32_t idx_start,
02138                                             uint32_t num_entry);
02139
02140 static inline
02141 enum iopmp_error iopmp_set_entry_to_md(IOPMP_t *iopmp, uint32_t mdidx,
02142                                         const struct iopmp_entry *entry,
02143                                         uint32_t idx)
02144 {
02145     return iopmp_set_entries_to_md(iopmp, mdidx, entry, idx, 1);
02146 }
02147
02148 enum iopmp_error iopmp_get_entries_from_md(IOPMP_t *iopmp, uint32_t mdidx,
02149                                               struct iopmp_entry *entry_array,
02150                                               uint32_t idx_start,
02151                                               uint32_t num_entry);
02152
02153 static inline
02154 enum iopmp_error iopmp_get_entry_from_md(IOPMP_t *iopmp, uint32_t mdidx,
02155                                             struct iopmp_entry *entry,
02156                                             uint32_t idx)
02157 {
02158     return iopmp_get_entries_from_md(iopmp, mdidx, entry, idx, 1);
02159 }
02160
02161 enum iopmp_error iopmp_get_entries(IOPMP_t *iopmp,
02162                                       struct iopmp_entry *entry_array,
02163                                       uint32_t idx_start, uint32_t num_entry);
02164
02165 static inline
02166 enum iopmp_error iopmp_get_entry(IOPMP_t *iopmp, struct iopmp_entry *entry,
02167                                   uint32_t idx)
02168 {
02169     return iopmp_get_entries(iopmp, entry, idx, 1);
02170 }
02171
02172 enum iopmp_error iopmp_set_entries(IOPMP_t *iopmp,
02173                                       const struct iopmp_entry *entry_array,
02174                                       uint32_t idx_start, uint32_t num_entry);
02175
02176 static inline
02177 enum iopmp_error iopmp_set_entry(IOPMP_t *iopmp,
02178                                   const struct iopmp_entry *entry,
02179                                   uint32_t idx)
02180 {
02181     return iopmp_set_entries(iopmp, entry, idx, 1);
02182 }
02183
02184 enum iopmp_error iopmp_clear_entries_in_md(IOPMP_t *iopmp, uint32_t mdidx);
02185
02186 enum iopmp_error iopmp_clear_entries(IOPMP_t *iopmp, uint32_t idx_start,
02187                                         uint32_t num_entry);
02188
02189 static inline enum iopmp_error iopmp_clear_entry(IOPMP_t *iopmp, uint32_t idx)
02190 {
02191     return iopmp_clear_entries(iopmp, idx, 1);
02192 }
02193
02194 enum iopmp_error iopmp_entries_get_belong_md(IOPMP_t *iopmp, uint32_t idx_start,
02195                                                 uint32_t num_entry, uint64_t *mds);
02196
02197
```

```
02488 #endif
```

### 5.3 README.md File Reference

# Index

a  
    iopmp\_entry, 10  
addr  
    iopmp\_entry, 10  
    iopmp\_err\_report, 13  
    iopmp\_instance, 15  
addr\_entry\_array  
    iopmp\_instance, 16  
addrh  
    iopmp\_entry, 10  
addrh\_en  
    iopmp\_instance, 20  
addrl  
    iopmp\_entry, 10  
cfg  
    iopmp\_entry, 11  
eid  
    iopmp\_err\_report, 13  
enable  
    iopmp\_instance, 20  
entry\_addr\_bits  
    iopmp\_instance, 15  
entry\_num  
    iopmp\_instance, 16  
entrylck\_f  
    iopmp\_instance, 18  
entrylck\_lock  
    iopmp\_instance, 17  
err\_cfg\_lock  
    iopmp\_instance, 20  
err\_resp\_suppress  
    iopmp\_instance, 21  
etype  
    iopmp\_err\_report, 13  
granularity  
    iopmp\_instance, 15  
impid  
    iopmp\_instance, 16  
init  
    iopmp\_instance, 18  
intr\_enable  
    iopmp\_instance, 21  
iopmp\_capture\_error  
    libiopmp.h, 65  
iopmp\_clear\_entries  
    libiopmp.h, 84  
iopmp\_clear\_entries\_in\_md  
    libiopmp.h, 84  
iopmp\_clear\_entry  
    libiopmp.h, 85  
iopmp\_encode\_entry  
    libiopmp.h, 78  
iopmp\_entries\_get\_belong\_md  
    libiopmp.h, 86  
iopmp\_entry, 9  
    a, 10  
    addr, 10  
    addrh, 10  
    addrl, 10  
    cfg, 11  
    prient\_flag, 12  
    private\_data, 12  
    r, 10  
    rsv, 11  
    sere, 11  
    sewe, 11  
    sexe, 11  
    sire, 11  
    siwe, 11  
    sixe, 11  
    w, 10  
    x, 10  
IOPMP\_ENTRY\_A\_MASK  
    libiopmp.h, 35  
IOPMP\_ENTRY\_A\_NA4  
    libiopmp.h, 35  
IOPMP\_ENTRY\_A\_NAPOT  
    libiopmp.h, 35  
IOPMP\_ENTRY\_A\_OFF  
    libiopmp.h, 35  
IOPMP\_ENTRY\_A\_TOR  
    libiopmp.h, 35  
IOPMP\_ENTRY\_FIRST\_TOR  
    libiopmp.h, 36  
iopmp\_entry\_flags  
    libiopmp.h, 35  
IOPMP\_ENTRY\_FORCE\_OFF  
    libiopmp.h, 36  
IOPMP\_ENTRY\_FORCE\_TOR  
    libiopmp.h, 36  
iopmp\_entry\_get\_addr  
    libiopmp.h, 51  
iopmp\_entry\_get\_cfg  
    libiopmp.h, 51  
IOPMP\_ENTRY\_NON\_PRIO

libiopmp.h, 36  
**IOPMP\_ENTRY\_PRIO**  
 libiopmp.h, 36  
**IOPMP\_ENTRY\_R**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_RW**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_RX**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SEE\_MASK**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SERE**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SEWE**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SEXE**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SIE\_MASK**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SIRE**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SIWE**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SIXE**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_SW\_FLAGS\_MASK**  
 libiopmp.h, 36  
**IOPMP\_Entry\_t**  
 libiopmp.h, 32  
**IOPMP\_ENTRY\_W**  
 libiopmp.h, 35  
**IOPMP\_ENTRY\_X**  
 libiopmp.h, 35  
**IOPMP\_ERR\_ILLEGAL\_VALUE**  
 libiopmp.h, 36  
**IOPMP\_ERR\_INVALID\_PARAMETER**  
 libiopmp.h, 36  
**IOPMP\_ERR\_INVALID\_PRIORITY**  
 libiopmp.h, 36  
**IOPMP\_ERR\_NOT\_ALLOWED**  
 libiopmp.h, 36  
**IOPMP\_ERR\_NOT\_AVAILABLE**  
 libiopmp.h, 36  
**IOPMP\_ERR\_NOT\_EXIST**  
 libiopmp.h, 36  
**IOPMP\_ERR\_NOT\_SUPPORTED**  
 libiopmp.h, 36  
**IOPMP\_ERR\_OUT\_OF\_BOUNDS**  
 libiopmp.h, 36  
**IOPMP\_ERR\_REG\_IS\_LOCKED**  
 libiopmp.h, 36  
**iopmp\_err\_report**, 12  
 addr, 13  
 eid, 13  
 etype, 13  
 msi\_werr, 13  
 rrid, 13  
 svc, 13  
 ttype, 13  
**iopmp\_err\_report\_get\_addr**  
 libiopmp.h, 48  
**iopmp\_err\_report\_get\_eid**  
 libiopmp.h, 49  
**iopmp\_err\_report\_get\_etype**  
 libiopmp.h, 50  
**iopmp\_err\_report\_get\_msi\_werr**  
 libiopmp.h, 50  
**iopmp\_err\_report\_get\_rrid**  
 libiopmp.h, 48  
**iopmp\_err\_report\_get\_svc**  
 libiopmp.h, 51  
**iopmp\_err\_report\_get\_ttype**  
 libiopmp.h, 50  
**iopmp\_err\_report\_is\_no\_hit**  
 libiopmp.h, 49  
**iopmp\_err\_report\_is\_part\_hit**  
 libiopmp.h, 49  
**IOPMP\_ERR\_REPORT\_t**  
 libiopmp.h, 32  
**iopmp\_errinfo\_etype**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_INST\_FETCH**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_NONE**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_NOT\_HIT**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_PART\_HIT**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_READ**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_RESERVED\_0**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_RESERVED\_1**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_RESERVED\_2**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_RESERVED\_3**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_RESERVED\_4**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_RESERVED\_5**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_STALL**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_UNKNOWN\_RRID**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_USER\_DEF\_0**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_USER\_DEF\_1**  
 libiopmp.h, 33  
**IOPMP\_ERRINFO\_ETYPE\_WRITE**  
 libiopmp.h, 33  
**iopmp\_errinfo\_ttype**

libiopmp.h, 33  
IOPMP\_ERRINFO\_TTYPE\_INST\_FETCH  
    libiopmp.h, 33  
IOPMP\_ERRINFO\_TTYPE\_READ  
    libiopmp.h, 33  
IOPMP\_ERRINFO\_TTYPE\_RSVD  
    libiopmp.h, 33  
IOPMP\_ERRINFO\_TTYPE\_WRITE  
    libiopmp.h, 33  
iopmp\_error  
    libiopmp.h, 36  
iopmp\_get\_addrh\_en  
    libiopmp.h, 43  
iopmp\_get\_and\_clear\_msi\_werr  
    libiopmp.h, 64  
iopmp\_get\_base\_addr  
    libiopmp.h, 37  
iopmp\_get\_base\_addr\_entry\_array  
    libiopmp.h, 38  
iopmp\_get\_enable  
    libiopmp.h, 43  
iopmp\_get\_entries  
    libiopmp.h, 81  
iopmp\_get\_entries\_from\_md  
    libiopmp.h, 80  
iopmp\_get\_entry  
    libiopmp.h, 82  
iopmp\_get\_entry\_from\_md  
    libiopmp.h, 81  
iopmp\_get\_entry\_num  
    libiopmp.h, 44  
iopmp\_get\_global\_err\_resp  
    libiopmp.h, 46  
iopmp\_get\_global\_intr  
    libiopmp.h, 46  
iopmp\_get\_granularity  
    libiopmp.h, 38  
iopmp\_get\_impid  
    libiopmp.h, 54  
iopmp\_get\_locked\_entry\_num  
    libiopmp.h, 48  
iopmp\_get\_locked\_md  
    libiopmp.h, 59  
iopmp\_get\_locked\_mdcfg\_num  
    libiopmp.h, 61  
iopmp\_get\_md\_entry\_association  
    libiopmp.h, 75  
iopmp\_get\_md\_entry\_num  
    libiopmp.h, 77  
iopmp\_get\_md\_num  
    libiopmp.h, 43  
iopmp\_get\_mdcfg\_fmt  
    libiopmp.h, 38  
iopmp\_get\_msi\_addr  
    libiopmp.h, 63  
iopmp\_get\_msi\_data  
    libiopmp.h, 63  
iopmp\_get\_msi\_sel  
    libiopmp.h, 47  
iopmp\_get\_no\_w  
    libiopmp.h, 41  
iopmp\_get\_no\_x  
    libiopmp.h, 41  
iopmp\_get\_prio\_entry\_num  
    libiopmp.h, 44  
iopmp\_get\_rrid\_md\_association  
    libiopmp.h, 68  
iopmp\_get\_rrid\_num  
    libiopmp.h, 44  
iopmp\_get\_rrid\_transl  
    libiopmp.h, 56  
iopmp\_get\_rrid\_transl\_prog  
    libiopmp.h, 55  
iopmp\_get\_specver  
    libiopmp.h, 52  
iopmp\_get\_srcmd\_fmt  
    libiopmp.h, 39  
iopmp\_get\_stallViolation\_en  
    libiopmp.h, 46  
iopmp\_get\_support\_chk\_x  
    libiopmp.h, 40  
iopmp\_get\_support\_mfr  
    libiopmp.h, 42  
iopmp\_get\_support\_pees  
    libiopmp.h, 42  
iopmp\_get\_support\_peis  
    libiopmp.h, 42  
iopmp\_get\_support\_programmable\_prio\_entry  
    libiopmp.h, 40  
iopmp\_get\_support\_rrid\_transl  
    libiopmp.h, 40  
iopmp\_get\_support\_sps  
    libiopmp.h, 39  
iopmp\_get\_support\_stall  
    libiopmp.h, 41  
iopmp\_get\_support\_stall\_by\_md  
    libiopmp.h, 45  
iopmp\_get\_support\_stall\_by\_rrid  
    libiopmp.h, 45  
iopmp\_get\_support\_tor  
    libiopmp.h, 39  
iopmp\_get\_vendor\_id  
    libiopmp.h, 52  
iopmp\_impid  
    libiopmp.h, 33  
IOPMP\_IMPID\_NOT\_SPECIFIED  
    libiopmp.h, 34  
iopmp\_init  
    libiopmp.h, 52  
iopmp\_instance, 14  
    addr, 15  
    addr\_entry\_array, 16  
    addrh\_en, 20  
    enable, 20  
    entry\_addr\_bits, 15  
    entry\_num, 16

entrylck\_f, 18  
 entrylck\_lock, 17  
 err\_cfg\_lock, 20  
 err\_resp\_suppress, 21  
 granularity, 15  
 impid, 16  
 init, 18  
 intr\_enable, 21  
 is\_stalling, 21  
 md\_entry\_num, 17  
 md\_num, 17  
 mdcfg\_fmt, 18  
 mdcfglck\_f, 17  
 mdcfglck\_lock, 17  
 mdlck\_lock, 17  
 mdlck\_md, 17  
 mfr\_en, 20  
 msi\_en, 21  
 msi\_sel, 21  
 msiaddr64, 18  
 msidata, 18  
 no\_err\_rec, 18  
 no\_w, 20  
 no\_x, 19  
 non\_prio\_en, 19  
 ops\_generic, 15  
 ops\_specific, 15  
 ops\_sps, 16  
 pees, 20  
 peis, 20  
 prio\_ent\_prog, 19  
 prio\_entry\_num, 16  
 rrid\_num, 16  
 rrid\_transl, 16  
 rrid\_transl\_en, 19  
 rrid\_transl\_prog, 19  
 specver, 17  
 sps\_en, 19  
 srcmd\_fmt, 18  
 stall\_en, 20  
 stallViolation\_en, 21  
 support\_stall\_by\_md, 21  
 support\_stall\_by\_rrid, 21  
 tor\_en, 19  
 vendor, 16  
 xinr, 19  
 iopmp\_invalidate\_error  
     libiopmp.h, 65  
 iopmp\_is\_entrylck\_locked  
     libiopmp.h, 47  
 iopmp\_is\_err\_cfg\_locked  
     libiopmp.h, 45  
 iopmp\_is\_initialized  
     libiopmp.h, 37  
 iopmp\_is\_mdcfglck\_locked  
     libiopmp.h, 61  
 iopmp\_is\_mdlck\_locked  
     libiopmp.h, 47  
 iopmp\_is\_srcmd\_table\_fmt\_0\_locked  
     libiopmp.h, 67  
 iopmp\_is\_srcmd\_table\_fmt\_2\_locked  
     libiopmp.h, 68  
 iopmp\_lock\_entries  
     libiopmp.h, 61  
 iopmp\_lock\_err\_cfg  
     libiopmp.h, 62  
 iopmp\_lock\_md  
     libiopmp.h, 60  
 iopmp\_lock\_mdcfg  
     libiopmp.h, 60  
 iopmp\_lock\_prio\_entry\_num  
     libiopmp.h, 54  
 iopmp\_lock\_rrid\_transl  
     libiopmp.h, 54  
 iopmp\_lock\_srcmd\_table\_fmt\_0  
     libiopmp.h, 66  
 iopmp\_lock\_srcmd\_table\_fmt\_2  
     libiopmp.h, 67  
 IOPMP\_MAX\_RRID\_SRCMD\_FMT\_2  
     libiopmp.h, 29  
 iopmp\_mdcfg\_fmt  
     libiopmp.h, 34  
 IOPMP\_MDCFG\_FMT\_0  
     libiopmp.h, 34  
 IOPMP\_MDCFG\_FMT\_1  
     libiopmp.h, 34  
 IOPMP\_MDCFG\_FMT\_2  
     libiopmp.h, 34  
 IOPMP\_MDCFG\_FMT\_MAX  
     libiopmp.h, 34  
 IOPMP\_MDCFG\_FMT\_RESERVED  
     libiopmp.h, 34  
 iopmp\_mfr\_get\_sv\_window  
     libiopmp.h, 66  
 iopmp\_model  
     libiopmp.h, 34  
 IOPMP\_MODEL\_6  
     libiopmp.h, 34  
 IOPMP\_MODEL\_8  
     libiopmp.h, 34  
 IOPMP\_MODEL\_9  
     libiopmp.h, 34  
 IOPMP\_MODEL\_COMPACT\_K  
     libiopmp.h, 34  
 IOPMP\_MODEL\_DYNAMIC\_K  
     libiopmp.h, 34  
 IOPMP\_MODEL\_FULL  
     libiopmp.h, 34  
 IOPMP\_MODEL\_ISOLATION  
     libiopmp.h, 34  
 IOPMP\_MODEL\_RAPID\_K  
     libiopmp.h, 34  
 IOPMP\_MODEL\_RESERVED\_10  
     libiopmp.h, 34  
 IOPMP\_MODEL\_RESERVED\_11  
     libiopmp.h, 34

IOPMP\_MODEL\_RESERVED\_12  
    libiopmp.h, 34  
IOPMP\_MODEL\_RESERVED\_13  
    libiopmp.h, 34  
IOPMP\_MODEL\_RESERVED\_14  
    libiopmp.h, 34  
IOPMP\_MODEL\_RESERVED\_15  
    libiopmp.h, 34  
IOPMP\_MODEL\_RESERVED\_3  
    libiopmp.h, 34  
IOPMP\_MODEL\_RESERVED\_7  
    libiopmp.h, 34  
IOPMP\_OK  
    libiopmp.h, 36  
IOPMP\_PRIENT\_ANY  
    libiopmp.h, 33  
iomp\_prient\_flags  
    libiopmp.h, 32  
IOPMP\_PRIENT\_NON\_PRIORITY  
    libiopmp.h, 33  
IOPMP\_PRIENT\_PRIORITY  
    libiopmp.h, 33  
iomp\_query\_stall\_stat\_by\_rrid  
    libiopmp.h, 59  
iomp\_resume\_transactions  
    libiopmp.h, 57  
iomp\_rridscp\_op  
    libiopmp.h, 34  
IOPMP\_RRIDSCP\_OP\_DONT\_STALL  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_OP\_QUERY  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_OP\_RESERVED  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_OP\_STALL  
    libiopmp.h, 35  
iomp\_rridscp\_stat  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_STAT\_ERR\_RRID  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_STAT\_NOT\_IMPL  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_STAT\_NOT\_STALLED  
    libiopmp.h, 35  
IOPMP\_RRIDSCP\_STAT\_STALLED  
    libiopmp.h, 35  
iomp\_set\_enable  
    libiopmp.h, 55  
iomp\_set\_entries  
    libiopmp.h, 83  
iomp\_set\_entries\_to\_md  
    libiopmp.h, 79  
iomp\_set\_entry  
    libiopmp.h, 83  
iomp\_set\_entry\_to\_md  
    libiopmp.h, 79  
iomp\_set\_global\_err\_resp  
    libiopmp.h, 62  
iomp\_set\_global\_intr  
    libiopmp.h, 62  
iomp\_set\_md\_entry\_association  
    libiopmp.h, 76  
iomp\_set\_md\_entry\_association\_multi  
    libiopmp.h, 75  
iomp\_set\_md\_entry\_num  
    libiopmp.h, 77  
iomp\_set\_md\_permission  
    libiopmp.h, 69  
iomp\_set\_md\_permission\_multi  
    libiopmp.h, 70  
iomp\_set\_msi\_info  
    libiopmp.h, 64  
iomp\_set\_msi\_sel  
    libiopmp.h, 63  
iomp\_set\_prio\_entry\_num  
    libiopmp.h, 55  
iomp\_set\_rrid\_md\_association  
    libiopmp.h, 69  
iomp\_set\_rrid\_transl  
    libiopmp.h, 56  
iomp\_set\_srcmd\_perm\_cfg  
    libiopmp.h, 70  
iomp\_set\_srcmd\_perm\_cfg\_nocheck  
    libiopmp.h, 71  
iomp\_set\_stallViolation\_en  
    libiopmp.h, 65  
iomp\_sps\_get\_rrid\_md\_insn\_fetch  
    libiopmp.h, 73  
iomp\_sps\_get\_rrid\_md\_read  
    libiopmp.h, 71  
iomp\_sps\_get\_rrid\_md\_rwx  
    libiopmp.h, 74  
iomp\_sps\_get\_rrid\_md\_write  
    libiopmp.h, 72  
iomp\_sps\_set\_rrid\_insn\_fetch  
    libiopmp.h, 73  
iomp\_sps\_set\_rrid\_md\_read  
    libiopmp.h, 71  
iomp\_sps\_set\_rrid\_md\_rwx  
    libiopmp.h, 74  
iomp\_sps\_set\_rrid\_md\_write  
    libiopmp.h, 72  
iomp\_srcmd\_fmt  
    libiopmp.h, 34  
IOPMP\_SRCMD\_FMT\_0  
    libiopmp.h, 34  
IOPMP\_SRCMD\_FMT\_1  
    libiopmp.h, 34  
IOPMP\_SRCMD\_FMT\_2  
    libiopmp.h, 34  
IOPMP\_SRCMD\_FMT\_MAX  
    libiopmp.h, 34  
IOPMP\_SRCMD\_FMT\_RESERVED  
    libiopmp.h, 34  
IOPMP\_SRCMD\_PERM\_CFG\_SET\_DIRECT  
    libiopmp.h, 30

IOPMP\_SRCMD\_PERM\_CFG\_t  
*libiopmp.h*, 32

iopmp\_srcmd\_perm\_config, 22  
*srcmd\_perm\_mask*, 22  
*srcmd\_perm\_val*, 22

IOPMP\_SRCMD\_PERM\_MASK  
*libiopmp.h*, 30

IOPMP\_SRCMD\_PERM\_R  
*libiopmp.h*, 29

IOPMP\_SRCMD\_PERM\_W  
*libiopmp.h*, 29

iopmp\_stall\_cherry\_pick\_rrid  
*libiopmp.h*, 58

iopmp\_stall\_transactions\_by\_mds  
*libiopmp.h*, 56

IOPMP\_t  
*libiopmp.h*, 32

iopmp\_transactions\_are\_resumed  
*libiopmp.h*, 58

iopmp\_transactions\_are\_stalled  
*libiopmp.h*, 57

is\_stalling  
*iopmp\_instance*, 21

**libiopmp - A Library to Program RISC-V IOPMP**, 1  
*libiopmp.h*, 23, 86

iopmp\_capture\_error, 65

iopmp\_clear\_entries, 84

iopmp\_clear\_entries\_in\_md, 84

iopmp\_clear\_entry, 85

iopmp\_encode\_entry, 78

iopmp\_entries\_get\_belong\_md, 86

IOPMP\_ENTRY\_A\_MASK, 35

IOPMP\_ENTRY\_A\_NA4, 35

IOPMP\_ENTRY\_A\_NAPOT, 35

IOPMP\_ENTRY\_A\_OFF, 35

IOPMP\_ENTRY\_A\_TOR, 35

IOPMP\_ENTRY\_FIRST\_TOR, 36

iopmp\_entry\_flags, 35

IOPMP\_ENTRY\_FORCE\_OFF, 36

IOPMP\_ENTRY\_FORCE\_TOR, 36

iopmp\_entry\_get\_addr, 51

iopmp\_entry\_get\_cfg, 51

IOPMP\_ENTRY\_NON\_PRIO, 36

IOPMP\_ENTRY\_PRIO, 36

IOPMP\_ENTRY\_R, 35

IOPMP\_ENTRY\_RW, 35

IOPMP\_ENTRY\_RXW, 35

IOPMP\_ENTRY\_RX, 35

IOPMP\_ENTRY\_SEE\_MASK, 35

IOPMP\_ENTRY\_SERE, 35

IOPMP\_ENTRY\_SEWE, 35

IOPMP\_ENTRY\_SEXE, 35

IOPMP\_ENTRY\_SIE\_MASK, 35

IOPMP\_ENTRY\_SIRE, 35

IOPMP\_ENTRY\_SIWE, 35

IOPMP\_ENTRY\_SIXE, 35

IOPMP\_ENTRY\_SW\_FLAGS\_MASK, 36

IOPMP\_Entry\_t, 32

IOPMP\_ENTRY\_W, 35

IOPMP\_ENTRY\_X, 35

IOPMP\_ERR\_ILLEGAL\_VALUE, 36

IOPMP\_ERR\_INVALID\_PARAMETER, 36

IOPMP\_ERR\_INVALID\_PRIORITY, 36

IOPMP\_ERR\_NOT\_ALLOWED, 36

IOPMP\_ERR\_NOT\_AVAILABLE, 36

IOPMP\_ERR\_NOT\_EXIST, 36

IOPMP\_ERR\_NOT\_SUPPORTED, 36

IOPMP\_ERR\_OUT\_OF\_BOUNDS, 36

IOPMP\_ERR\_REG\_IS\_LOCKED, 36

iopmp\_err\_report\_get\_addr, 48

iopmp\_err\_report\_get\_eid, 49

iopmp\_err\_report\_get\_etype, 50

iopmp\_err\_report\_get\_msi\_werr, 50

iopmp\_err\_report\_get\_rrid, 48

iopmp\_err\_report\_get\_svc, 51

iopmp\_err\_report\_get\_ttype, 50

iopmp\_err\_report\_is\_no\_hit, 49

iopmp\_err\_report\_is\_part\_hit, 49

IOPMP\_ERR\_REPORT\_t, 32

iopmp\_errinfo\_etype, 33

IOPMP\_ERRINFO\_ETYPE\_INST\_FETCH, 33

IOPMP\_ERRINFO\_ETYPE\_NONE, 33

IOPMP\_ERRINFO\_ETYPE\_NOT\_HIT, 33

IOPMP\_ERRINFO\_ETYPE\_PART\_HIT, 33

IOPMP\_ERRINFO\_ETYPE\_READ, 33

IOPMP\_ERRINFO\_ETYPE\_RESERVED\_0, 33

IOPMP\_ERRINFO\_ETYPE\_RESERVED\_1, 33

IOPMP\_ERRINFO\_ETYPE\_RESERVED\_2, 33

IOPMP\_ERRINFO\_ETYPE\_RESERVED\_3, 33

IOPMP\_ERRINFO\_ETYPE\_RESERVED\_4, 33

IOPMP\_ERRINFO\_ETYPE\_RESERVED\_5, 33

IOPMP\_ERRINFO\_ETYPE\_STALL, 33

IOPMP\_ERRINFO\_ETYPE\_UNKNOWN\_RRID, 33

IOPMP\_ERRINFO\_ETYPE\_USER\_DEF\_0, 33

IOPMP\_ERRINFO\_ETYPE\_USER\_DEF\_1, 33

IOPMP\_ERRINFO\_ETYPE\_WRITE, 33

iopmp\_errinfo\_ttype, 33

IOPMP\_ERRINFO\_TTYPE\_INST\_FETCH, 33

IOPMP\_ERRINFO\_TTYPE\_READ, 33

IOPMP\_ERRINFO\_TTYPE\_RSVD, 33

IOPMP\_ERRINFO\_TTYPE\_WRITE, 33

iopmp\_error, 36

iopmp\_get\_addr\_en, 43

iopmp\_get\_and\_clear\_msi\_werr, 64

iopmp\_get\_base\_addr, 37

iopmp\_get\_base\_addr\_entry\_array, 38

iopmp\_get\_enable, 43

iopmp\_get\_entries, 81

iopmp\_get\_entries\_from\_md, 80

iopmp\_get\_entry, 82

iopmp\_get\_entry\_from\_md, 81

iopmp\_get\_entry\_num, 44

iopmp\_get\_global\_err\_resp, 46

iopmp\_get\_global\_intr, 46

iopmp\_get\_granularity, 38

iopmp\_get\_impid, 54  
iopmp\_get\_locked\_entry\_num, 48  
iopmp\_get\_locked\_md, 59  
iopmp\_get\_locked\_mdcfg\_num, 61  
iopmp\_get\_md\_entry\_association, 75  
iopmp\_get\_md\_entry\_num, 77  
iopmp\_get\_md\_num, 43  
iopmp\_get\_mdcfg\_fmt, 38  
iopmp\_get\_msi\_addr, 63  
iopmp\_get\_msi\_data, 63  
iopmp\_get\_msi\_sel, 47  
iopmp\_get\_no\_w, 41  
iopmp\_get\_no\_x, 41  
iopmp\_get\_prio\_entry\_num, 44  
iopmp\_get\_rrid\_md\_association, 68  
iopmp\_get\_rrid\_num, 44  
iopmp\_get\_rrid\_transl, 56  
iopmp\_get\_rrid\_transl\_prog, 55  
iopmp\_get\_specver, 52  
iopmp\_get\_srcmd\_fmt, 39  
iopmp\_get\_stallViolation\_en, 46  
iopmp\_get\_support\_chk\_x, 40  
iopmp\_get\_support\_mfr, 42  
iopmp\_get\_support\_pees, 42  
iopmp\_get\_support\_peis, 42  
iopmp\_get\_support\_programmable\_prio\_entry, 40  
iopmp\_get\_support\_rrid\_transl, 40  
iopmp\_get\_support\_sps, 39  
iopmp\_get\_support\_stall, 41  
iopmp\_get\_support\_stall\_by\_md, 45  
iopmp\_get\_support\_stall\_by\_rrid, 45  
iopmp\_get\_support\_tor, 39  
iopmp\_get\_vendor\_id, 52  
iopmp\_impid, 33  
IOPMP\_IMPID\_NOT\_SPECIFIED, 34  
iopmp\_init, 52  
iopmp\_invalidate\_error, 65  
iopmp\_is\_entrylck\_locked, 47  
iopmp\_is\_err\_cfg\_locked, 45  
iopmp\_is\_initialized, 37  
iopmp\_is\_mdcfglck\_locked, 61  
iopmp\_is\_mdlck\_locked, 47  
iopmp\_is\_srcmd\_table\_fmt\_0\_locked, 67  
iopmp\_is\_srcmd\_table\_fmt\_2\_locked, 68  
iopmp\_lock\_entries, 61  
iopmp\_lock\_err\_cfg, 62  
iopmp\_lock\_md, 60  
iopmp\_lock\_mdcfg, 60  
iopmp\_lock\_prio\_entry\_num, 54  
iopmp\_lock\_rrid\_transl, 54  
iopmp\_lock\_srcmd\_table\_fmt\_0, 66  
iopmp\_lock\_srcmd\_table\_fmt\_2, 67  
IOPMP\_MAX\_RRID\_SRCMD\_FMT\_2, 29  
iopmp\_mdcfg\_fmt, 34  
IOPMP\_MDCFG\_FMT\_0, 34  
IOPMP\_MDCFG\_FMT\_1, 34  
IOPMP\_MDCFG\_FMT\_2, 34  
IOPMP\_MDCFG\_FMT\_MAX, 34  
IOPMP\_MDCFG\_FMT\_RESERVED, 34  
iopmp\_mfr\_get\_sv\_window, 66  
iopmp\_model, 34  
IOPMP\_MODEL\_6, 34  
IOPMP\_MODEL\_8, 34  
IOPMP\_MODEL\_9, 34  
IOPMP\_MODEL\_COMPACT\_K, 34  
IOPMP\_MODEL\_DYNAMIC\_K, 34  
IOPMP\_MODEL\_FULL, 34  
IOPMP\_MODEL\_ISOLATION, 34  
IOPMP\_MODEL\_RAPID\_K, 34  
IOPMP\_MODEL\_RESERVED\_10, 34  
IOPMP\_MODEL\_RESERVED\_11, 34  
IOPMP\_MODEL\_RESERVED\_12, 34  
IOPMP\_MODEL\_RESERVED\_13, 34  
IOPMP\_MODEL\_RESERVED\_14, 34  
IOPMP\_MODEL\_RESERVED\_15, 34  
IOPMP\_MODEL\_RESERVED\_3, 34  
IOPMP\_MODEL\_RESERVED\_7, 34  
IOPMP\_OK, 36  
IOPMP\_PRIENT\_ANY, 33  
iopmp\_prient\_flags, 32  
IOPMP\_PRIENT\_NON\_PRIORITY, 33  
IOPMP\_PRIENT\_PRIORITY, 33  
iopmp\_query\_stall\_stat\_by\_rrid, 59  
iopmp\_resume\_transactions, 57  
iopmp\_rridscp\_op, 34  
IOPMP\_RRIDSCP\_OP\_DONT\_STALL, 35  
IOPMP\_RRIDSCP\_OP\_QUERY, 35  
IOPMP\_RRIDSCP\_OP\_RESERVED, 35  
IOPMP\_RRIDSCP\_OP\_STALL, 35  
iopmp\_rridscp\_stat, 35  
IOPMP\_RRIDSCP\_STAT\_ERR\_RRID, 35  
IOPMP\_RRIDSCP\_STAT\_NOT\_IMPL, 35  
IOPMP\_RRIDSCP\_STAT\_NOT\_STALLED, 35  
IOPMP\_RRIDSCP\_STAT\_STALLED, 35  
iopmp\_set\_enable, 55  
iopmp\_set\_entries, 83  
iopmp\_set\_entries\_to\_md, 79  
iopmp\_set\_entry, 83  
iopmp\_set\_entry\_to\_md, 79  
iopmp\_set\_global\_err\_resp, 62  
iopmp\_set\_global\_intr, 62  
iopmp\_set\_md\_entry\_association, 76  
iopmp\_set\_md\_entry\_association\_multi, 75  
iopmp\_set\_md\_entry\_num, 77  
iopmp\_set\_md\_permission, 69  
iopmp\_set\_md\_permission\_multi, 70  
iopmp\_set\_msi\_info, 64  
iopmp\_set\_msi\_sel, 63  
iopmp\_set\_prio\_entry\_num, 55  
iopmp\_set\_rrid\_md\_association, 69  
iopmp\_set\_rrid\_transl, 56  
iopmp\_set\_srcmd\_perm\_cfg, 70  
iopmp\_set\_srcmd\_perm\_cfg\_nocheck, 71  
iopmp\_set\_stallViolation\_en, 65  
iopmp\_sps\_get\_rrid\_md\_insn\_fetch, 73  
iopmp\_sps\_get\_rrid\_md\_read, 71



iopmp\_instance, 16  
private\_data  
    iopmp\_entry, 12

r  
    iopmp\_entry, 10

README.md, 96

rrid  
    iopmp\_err\_report, 13

rrid\_num  
    iopmp\_instance, 16

rrid\_transl  
    iopmp\_instance, 16

rrid\_transl\_en  
    iopmp\_instance, 19

rrid\_transl\_prog  
    iopmp\_instance, 19

rsv  
    iopmp\_entry, 11

sere  
    iopmp\_entry, 11

sewe  
    iopmp\_entry, 11

sexe  
    iopmp\_entry, 11

sire  
    iopmp\_entry, 11

siwe  
    iopmp\_entry, 11

sixe  
    iopmp\_entry, 11

specver  
    iopmp\_instance, 17

sps\_en  
    iopmp\_instance, 19

srcmd\_fmt  
    iopmp\_instance, 18

srcmd\_perm\_mask  
    iopmp\_srcmd\_perm\_config, 22

srcmd\_perm\_val  
    iopmp\_srcmd\_perm\_config, 22

stall\_en  
    iopmp\_instance, 20

stallViolation\_en  
    iopmp\_instance, 21

support\_stall\_by\_md  
    iopmp\_instance, 21

support\_stall\_by\_rrid  
    iopmp\_instance, 21

svc  
    iopmp\_err\_report, 13

tor\_en  
    iopmp\_instance, 19

ttype  
    iopmp\_err\_report, 13

vendor