ROCmSMI

Generated by Doxygen 1.8.11

Contents

1	ROC	Cm Syst	em Mana	gement Interface (ROCm SMI) Library	1
2	Mod	lule Inde	ex		5
	2.1	Module	es		5
3	Data	Struct	ure Index		7
	3.1	Data S	Structures		7
4	File	Index			9
	4.1	File Lis	st		9
5	Mod	lule Dod	cumentati	on	11
	5.1	Initializ	ation and	Shutdown	11
		5.1.1	Detailed	Description	11
		5.1.2	Function	Documentation	11
			5.1.2.1	rsmi_init(uint64_t init_flags)	11
			5.1.2.2	rsmi_shut_down(void)	12
	5.2	Identifi	er Queries	S	13
		5.2.1	Detailed	Description	13
		5.2.2	Function	Documentation	13
			5.2.2.1	rsmi_num_monitor_devices(uint32_t *num_devices)	13
			5.2.2.2	rsmi_dev_id_get(uint32_t dv_ind, uint16_t *id)	14
			5.2.2.3	rsmi_dev_vendor_id_get(uint32_t dv_ind, uint16_t *id)	14
			5.2.2.4	rsmi_dev_name_get(uint32_t dv_ind, char *name, size_t len)	14
			5.2.2.5	rsmi_dev_vendor_name_get(uint32_t id, char *name, size_t len)	15

iv CONTENTS

		5.2.2.6	rsmi_dev_subsystem_id_get(uint32_t dv_ind, uint16_t *id)	15
		5.2.2.7	rsmi_dev_subsystem_name_get(uint32_t dv_ind, char *name, size_t len)	16
		5.2.2.8	rsmi_dev_subsystem_vendor_id_get(uint32_t dv_ind, uint16_t *id)	16
		5.2.2.9	rsmi_dev_unique_id_get(uint32_t dv_ind, uint64_t *id)	16
5.3	PCIe C	Queries .		18
	5.3.1	Detailed	Description	18
	5.3.2	Function	Documentation	18
		5.3.2.1	rsmi_dev_pci_bandwidth_get(uint32_t dv_ind, rsmi_pcie_bandwidth_t *bandwidth)	18
		5.3.2.2	rsmi_dev_pci_id_get(uint32_t dv_ind, uint64_t *bdfid)	18
		5.3.2.3	rsmi_dev_pci_throughput_get(uint32_t dv_ind, uint64_t *sent, uint64_↔ t *received, uint64_t *max_pkt_sz)	19
		5.3.2.4	rsmi_dev_pci_replay_counter_get(uint32_t dv_ind, uint64_t *counter)	19
5.4	PCIe C	Control		20
	5.4.1	Detailed	Description	20
	5.4.2	Function	Documentation	20
		5.4.2.1	rsmi_dev_pci_bandwidth_set(uint32_t dv_ind, uint64_t bw_bitmask)	20
5.5	Power	Queries .		21
	5.5.1	Detailed	Description	21
	5.5.2	Function	Documentation	21
		5.5.2.1	rsmi_dev_power_ave_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_t *power)	21
		5.5.2.2	rsmi_dev_power_cap_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_t *cap) .	21
		5.5.2.3	rsmi_dev_power_cap_range_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_ ← t *max, uint64_t *min)	22
5.6	Power	Control .		23
	5.6.1	Detailed	Description	23
	5.6.2	Function	Documentation	23
		5.6.2.1	rsmi_dev_power_cap_set(uint32_t dv_ind, uint32_t sensor_ind, uint64_t cap)	23
		5.6.2.2	rsmi_dev_power_profile_set(uint32_t dv_ind, uint32_t sensor_ind, rsmi_power ← profile_preset_masks_t profile)	23
5.7	Memoi	ry Queries		25
	5.7.1	Detailed	Description	25
	5.7.2	Function	Documentation	25

CONTENTS

		5.7.2.1	rsmi_dev_memory_total_get(uint32_t dv_ind, rsmi_memory_type_t mem_type,	
			uint64_t *total)	25
		5.7.2.2	rsmi_dev_memory_usage_get(uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_t *used)	25
5.8	Physica	al State Qu	ueries	27
	5.8.1	Detailed	Description	27
	5.8.2	Function	Documentation	27
		5.8.2.1	rsmi_dev_fan_rpms_get(uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed) .	27
		5.8.2.2	rsmi_dev_fan_speed_get(uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed) .	27
		5.8.2.3	rsmi_dev_fan_speed_max_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_← t *max_speed)	28
		5.8.2.4	rsmi_dev_temp_metric_get(uint32_t dv_ind, uint32_t sensor_type, rsmi_ temperature_metric_t metric, int64_t *temperature)	28
5.9	Physica	al State Co	ontrol	30
	5.9.1	Detailed	Description	30
	5.9.2	Function	Documentation	30
		5.9.2.1	rsmi_dev_fan_reset(uint32_t dv_ind, uint32_t sensor_ind)	30
		5.9.2.2	rsmi_dev_fan_speed_set(uint32_t dv_ind, uint32_t sensor_ind, uint64_t speed) .	30
5.10	Clock,	Power and	Performance Queries	32
	5.10.1	Detailed	Description	32
	5.10.2	Function	Documentation	32
		5.10.2.1	rsmi_dev_busy_percent_get(uint32_t dv_ind, uint32_t *busy_percent)	32
		5.10.2.2	rsmi_dev_perf_level_get(uint32_t dv_ind, rsmi_dev_perf_level_t *perf)	32
		5.10.2.3	rsmi_dev_overdrive_level_get(uint32_t dv_ind, uint32_t *od)	33
		5.10.2.4	$ rsmi_dev_gpu_clk_freq_get(uint32_t \ dv_ind, \ rsmi_clk_type_t \ clk_type, \ rsmi_ \leftrightarrow frequencies_t \ *f) \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	33
		5.10.2.5	rsmi_dev_od_volt_info_get(uint32_t dv_ind, rsmi_od_volt_freq_data_t *odv)	34
		5.10.2.6	rsmi_dev_od_volt_curve_regions_get(uint32_t dv_ind, uint32_t *num_regions, rsmi_freq_volt_region_t *buffer)	34
		5.10.2.7	rsmi_dev_power_profile_presets_get(uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_profile_status_t *status)	34
5.11	Clock,	Power and	Performance Control	36
	5.11.1	Detailed	Description	36

vi

	5.11.2	Function	Documentation	36
		5.11.2.1	rsmi_dev_perf_level_set(int32_t dv_ind, rsmi_dev_perf_level_t perf_lvl)	36
		5.11.2.2	rsmi_dev_overdrive_level_set(int32_t dv_ind, uint32_t od)	36
		5.11.2.3	rsmi_dev_gpu_clk_freq_set(uint32_t dv_ind, rsmi_clk_type_t clk_type, uint64_← t freq_bitmask)	37
		5.11.2.4	rsmi_dev_od_freq_range_set(uint32_t dv_ind, rsmi_clk_type_t clk, rsmi_range ← _t *range)	37
5.12	2 Version	Queries		39
	5.12.1	Detailed	Description	39
	5.12.2	Function	Documentation	39
		5.12.2.1	rsmi_version_get(rsmi_version_t *version)	39
		5.12.2.2	rsmi_version_str_get(rsmi_sw_component_t component, char *ver_str, uint32_t len)	39
		5.12.2.3	rsmi_dev_vbios_version_get(uint32_t dv_ind, char *vbios, uint32_t len)	40
5.13	B Error C	ueries .		41
	5.13.1	Detailed	Description	41
	5.13.2	Function	Documentation	41
		5.13.2.1	rsmi_dev_ecc_count_get(uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_error_← count_t *ec)	41
		5.13.2.2	rsmi_dev_ecc_enabled_get(uint32_t dv_ind, uint64_t *enabled_mask)	41
		5.13.2.3	rsmi_dev_ecc_status_get(uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_ras_← err_state_t *state)	42
		5.13.2.4	rsmi_status_string(rsmi_status_t status, const char **status_string)	42
5.14	Perforn	nance Cou	unter Functions	43
	5.14.1	Detailed	Description	43
	5.14.2	Function	Documentation	43
		5.14.2.1	rsmi_dev_counter_group_supported(uint32_t dv_ind, rsmi_event_group_t group)	43
		5.14.2.2	rsmi_dev_counter_create(uint32_t dv_ind, rsmi_event_type_t type, rsmi_event ← _handle_t *evnt_handle)	44
		5.14.2.3	rsmi_dev_counter_destroy(rsmi_event_handle_t evnt_handle)	44
		5.14.2.4	rsmi_counter_control(rsmi_event_handle_t evt_handle, rsmi_counter_← command_t cmd, void *cmd_args)	44
		5.14.2.5	rsmi_counter_read(rsmi_event_handle_t evt_handle, rsmi_counter_value_t *value)	45
		5.14.2.6	rsmi_counter_available_counters_get(uint32_t dv_ind, rsmi_event_group_t grp, uint32_t *available)	45

CONTENTS vii

6	Data	Structi	ure Documentation	47
	6.1	rsmi_c	ounter_value_t Struct Reference	47
		6.1.1	Detailed Description	47
	6.2	rsmi_e	rror_count_t Struct Reference	47
		6.2.1	Detailed Description	48
	6.3	rsmi_fr	eq_volt_region_t Struct Reference	48
		6.3.1	Detailed Description	48
	6.4	rsmi_fr	equencies_t Struct Reference	48
		6.4.1	Detailed Description	49
		6.4.2	Field Documentation	49
			6.4.2.1 num_supported	49
			6.4.2.2 current	49
			6.4.2.3 frequency	49
	6.5	rsmi_o	d_vddc_point_t Struct Reference	49
		6.5.1	Detailed Description	49
	6.6	rsmi_o	d_volt_curve_t Struct Reference	50
		6.6.1	Detailed Description	50
		6.6.2	Field Documentation	50
			6.6.2.1 vc_points	50
	6.7	rsmi_o	d_volt_freq_data_t Struct Reference	50
		6.7.1	Detailed Description	51
		6.7.2	Field Documentation	51
			6.7.2.1 curr_mclk_range	51
	6.8	rsmi_p	cie_bandwidth_t Struct Reference	51
		6.8.1	Detailed Description	51
		6.8.2	Field Documentation	51
			6.8.2.1 transfer_rate	51
			6.8.2.2 lanes	51
	6.9	rsmi_p	ower_profile_status_t Struct Reference	52
		6.9.1	Detailed Description	52
		6.9.2	Field Documentation	52
			6.9.2.1 available_profiles	52
			6.9.2.2 current	52
			6.9.2.3 num_profiles	52
	6.10	rsmi_ra	ange_t Struct Reference	52
		6.10.1	Detailed Description	53
	6.11	rsmi_v	ersion_t Struct Reference	53
		6.11.1	Detailed Description	53

viii CONTENTS

7	File	Docum	entation		55
	7.1	rocm_s	smi.h File f	Reference	55
		7.1.1	Detailed	Description	60
		7.1.2	Macro De	efinition Documentation	60
			7.1.2.1	RSMI_MAX_FAN_SPEED	60
		7.1.3	Typedef [Documentation	60
			7.1.3.1	rsmi_event_handle_t	60
		7.1.4	Enumera	tion Type Documentation	61
			7.1.4.1	rsmi_status_t	61
			7.1.4.2	rsmi_init_flags_t	61
			7.1.4.3	rsmi_dev_perf_level_t	61
			7.1.4.4	rsmi_sw_component_t	62
			7.1.4.5	rsmi_event_group_t	62
			7.1.4.6	rsmi_event_type_t	62
			7.1.4.7	rsmi_counter_command_t	62
			7.1.4.8	rsmi_clk_type_t	63
			7.1.4.9	rsmi_temperature_metric_t	63
			7.1.4.10	rsmi_temperature_type_t	63
			7.1.4.11	rsmi_power_profile_preset_masks_t	64
			7.1.4.12	rsmi_gpu_block_t	64
			7.1.4.13	rsmi_ras_err_state_t	64
			7.1.4.14	rsmi_memory_type_t	64
			7.1.4.15	rsmi_freq_ind_t	65
Inc	lex				67

Chapter 1

ROCm System Management Interface (ROCm SMI) Library

The ROCm System Management Interface Library, or ROCm SMI library, is part of the Radeon Open Compute ROCm software stack. It is a C library for Linux that provides a user space interface for applications to monitor and control GPU applications.

Important note about Versioning and Backward Compatibility

The ROCm SMI library is currently under development, and therefore subject to change either at the ABI or API level. The intention is to keep the API as stable as possible even while in development, but in some cases we may need to break backwards compatibility in order to ensure future stability and usability. Following Semantic Versioning rules, while the ROCm SMI library is in high state of change, the major version will remain 0, and backward compatibility is not ensured.

Once new development has leveled off, the major version will become greater than 0, and backward compatibility will be enforced between major versions.

Building ROCm SMI

Additional Required software for building

In order to build the ROCm SMI library, the following components are required. Note that the software versions listed are what was used in development. Earlier versions are not guaranteed to work:

- CMake (v3.5.0)
- g++ (5.4.0)

In order to build the latest documentation, the following are required:

- DOxygen (1.8.11)
- latex (pdfTeX 3.14159265-2.6-1.40.16)

The source code for ROCm SMI is available on Github.

After the the ROCm SMI library git repository has been cloned to a local Linux machine, building the library is achieved by following the typical CMake build sequence. Specifically,

```
$ mk -p build
$ cd build
$ cmake <location of root of ROCm SMI library CMakeLists.txt>
$ make
```

The built library will appear in the build folder.

Building the Documentation

The documentation PDF file can be built with the following steps (continued from the steps above):

```
$ make doc
$ cd latex
$ make
```

The reference manual, refman.pdf will be in the latex directory upon a successful build.

Building the Tests

In order to verify the build and capability of ROCm SMI on your system and to see an example of how ROCm SMI can be used, you may build and run the tests that are available in the repo. To build the tests, follow these steps:

```
# Set environment variables used in CMakeLists.txt file
$ ROCM_DIR=<location of ROCm SMI library>
$ mkdir <location for test build>
$ cd <location for test build>
$ cmake -DROCM_DIR=<location of ROCM SMI library .so> <ROCm SMI source root>/tests/rocm_smi_test
```

To run the test, execute the program rsmitst that is built from the steps above. Make sure ROCm SMI library is in your library search path when executing the test program.

Usage Basics

Device Indices

Many of the functions in the library take a "device index". The device index is a number greater than or equal to 0, and less than the number of devices detected, as determined by rsmi_num_monitor_devices(). The index is used to distinguish the detected devices from one another. It is important to note that a device may end up with a different index after a reboot, so an index should not be relied upon to be constant over reboots.

Hello ROCm SMI

The only required ROCm-SMI call for any program that wants to use ROCm-SMI is the rsmi_init() call. This call initializes some internal data structures that will be used by subsequent ROCm-SMI calls.

When ROCm-SMI is no longer being used, <code>rsmi_shut_down()</code> should be called. This provides a way to do any releasing of resources that ROCm-SMI may have held. In many cases, this may have no effect, but may be necessary in future versions of the library.

A simple "Hello World" type program that displays the device ID of detected devices would look like this:

```
1 #include <stdint.h>
2 #include "rocm_smi/rocm_smi.h"
3 int main() {
   rsmi_status_t ret;
   uint32_t num_devices;
uint64_t dev_id;
    // We will skip return code checks for this example, but it
    // is recommended to always check this as some calls may not
10
     // apply for some devices or ROCm releases
11
     ret = rsmi_init(0);
12
13
     ret = rsmi_num_monitor_devices(&num_devices);
14
     for (int i=0; i < num_devices; ++i) {
16
      ret = rsmi_dev_id_get(i, &dev_id);
       // dev_id holds the device ID of device i, upon a
17
18
       // successful call
19
    ret = rsmi_shut_down();
20
21
    return 0;
22 }
```

ROCm System Management Interface (ROCm SMI) Library

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Initialization and Shutdown	11
Identifier Queries	13
PCIe Queries	
PCIe Control	20
Power Queries	21
Power Control	23
Memory Queries	
Physical State Queries	
Physical State Control	30
Clock, Power and Performance Queries	32
Clock, Power and Performance Control	
Version Queries	
Error Queries	41
Performance Counter Functions	43

6 Module Index

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

rsmi_counter_value_t	47
rsmi_error_count_t	
This structure holds error counts	47
rsmi_freq_volt_region_t	
This structure holds 2 rsmi_range_t's, one for frequency and one for voltage. These 2 ranges	
indicate the range of possible values for the corresponding rsmi_od_vddc_point_t	48
rsmi_frequencies_t	
This structure holds information about clock frequencies	48
rsmi_od_vddc_point_t	
This structure represents a point on the frequency-voltage plane	49
rsmi_od_volt_curve_t	50
rsmi_od_volt_freq_data_t	
This structure holds the frequency-voltage values for a device	50
rsmi_pcie_bandwidth_t	
This structure holds information about the possible PCIe bandwidths. Specifically, the possible	
transfer rates and their associated numbers of lanes are stored here	51
rsmi_power_profile_status_t	
This structure contains information about which power profiles are supported by the system for	
a given device, and which power profile is currently active	52
rsmi_range_t	
This structure represents a range (e.g., frequencies or voltages)	52
rsmi_version_t	
This structure holds version information	53

8 Data Structure Index

Chapter 4

File Index

4.1 File List

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

rocm smi.h

55

10 File Index

Chapter 5

Module Documentation

5.1 Initialization and Shutdown

Functions

- rsmi_status_t rsmi_init (uint64_t init_flags)
 - Initialize ROCm SMI.
- rsmi_status_t rsmi_shut_down (void)

Shutdown ROCm SMI.

5.1.1 Detailed Description

These functions are used for initialization of ROCm SMI and clean up when done.

5.1.2 Function Documentation

5.1.2.1 rsmi_status_t rsmi_init (uint64_t init_flags)

Initialize ROCm SMI.

When called, this initializes internal data structures, including those corresponding to sources of information that SMI provides.

Parameters

in	init_flags	Bit flags that tell SMI how to initialze. Values of rsmi_init_flags_t may be OR'd together and			
		passed through init_flags to modify how RSMI initializes.			

Return values

RSMI_STATUS_SUCCESS | is returned upon successful call.

5.1.2.2 rsmi_status_t rsmi_shut_down (void)

Shutdown ROCm SMI.

Do any necessary clean up.

5.2 Identifier Queries 13

5.2 Identifier Queries

Functions

rsmi status t rsmi num monitor devices (uint32 t *num devices)

Get the number of devices that have monitor information.

• rsmi_status_t rsmi_dev_id_get (uint32_t dv_ind, uint16_t *id)

Get the device id associated with the device with provided device index.

rsmi_status_t rsmi_dev_vendor_id_get (uint32_t dv_ind, uint16_t *id)

Get the device vendor id associated with the device with provided device index.

rsmi_status_t rsmi_dev_name_get (uint32_t dv_ind, char *name, size_t len)

Get the name string of a gpu device.

• rsmi_status_t rsmi_dev_vendor_name_get (uint32_t id, char *name, size_t len)

Get the name string for a give vendor ID.

rsmi_status_t rsmi_dev_subsystem_id_get (uint32_t dv_ind, uint16_t *id)

Get the subsystem device id associated with the device with provided device index.

rsmi_status_t rsmi_dev_subsystem_name_get (uint32_t dv_ind, char *name, size_t len)

Get the name string for the device subsytem.

rsmi_status_t rsmi_dev_subsystem_vendor_id_get (uint32_t dv_ind, uint16_t *id)

Get the device subsystem vendor id associated with the device with provided device index.

rsmi_status_t rsmi_dev_unique_id_get (uint32_t dv_ind, uint64_t *id)
 Get Unique ID.

5.2.1 Detailed Description

These functions provide identification information.

5.2.2 Function Documentation

5.2.2.1 rsmi_status_t rsmi_num_monitor_devices (uint32_t * num_devices)

Get the number of devices that have monitor information.

The number of devices which have monitors is returned. Monitors are referenced by the index which can be between 0 and num_devices - 1.

Parameters

in,out	num_devices	Caller provided pointer to uint32_t. Upon successful call, the value num_devices
		will contain the number of monitor devices.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.2.2.2 rsmi_status_t rsmi_dev_id_get (uint32_t dv_ind, uint16_t * id)

Get the device id associated with the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t id, this function will write the device id value to the uint64_t pointed to by id. This ID is an identification of the type of device, so calling this function for different devices will give the same value if they are kind of device. Consequently, this function should not be used to distinguish one device from another. rsmi_dev_pci_id_get() should be used to get a unique identifier.

Parameters

in	dv_ind	a device index
in,out	id	a pointer to uint64_t to which the device id will be written

Return values

RSMI STATUS SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.2.2.3 rsmi_status_t rsmi_dev_vendor_id_get (uint32_t dv_ind, uint16_t * id)

Get the device vendor id associated with the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t id, this function will write the device vendor id value to the uint64_t pointed to by id.

Parameters

in	dv_ind	a device index
in,out	id	a pointer to uint64_t to which the device vendor id will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.2.2.4 rsmi_status_t rsmi_dev_name_get (uint32_t dv_ind, char * name, size_t len)

Get the name string of a gpu device.

Given a device index dv_ind, a pointer to a caller provided char buffer name, and a length of this buffer len, this function will write the name of the device (up to len characters) to the buffer name.

If the integer ID associated with the device is not found in one of the system files containing device name information (e.g. /usr/share/misc/pci.ids), then this function will return the hex device ID as a string. Updating the system name files can be accompplished with "sudo update-pciids".

Parameters

in, out name a pointer to a caller provided char buffer to which the name will be written in len the length of the caller provided buffer name.	in	dv_ind	a device index	
	in,out	name	a pointer to a caller provided char buffer to which the name will be written	
	in	len		enerated by Doxygen

5.2 Identifier Queries 15

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
RSMI_STATUS_INSUFFICIENT_SIZE	is returned if len bytes is not large enough to hold the entire name.
	In this case, only len bytes will be written.

5.2.2.5 rsmi status t rsmi_dev_vendor_name_get (uint32_t id, char * name, size_t len)

Get the name string for a give vendor ID.

Given vendor ID id, a pointer to a caller provided char buffer name, and a length of this buffer len, this function will write the name of the vendor (up to len characters) buffer name. The id may be a device vendor or subsystem vendor ID.

If the integer ID associated with the vendor is not found in one of the system files containing device name information (e.g. /usr/share/misc/pci.ids), then this function will return the hex vendor ID as a string. Updating the system name files can be accompplished with "sudo update-pciids".

Parameters

in	id	a vendor ID
in,out	name	a pointer to a caller provided char buffer to which the name will be written
in	len	the length of the caller provided buffer name.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
RSMI_STATUS_INSUFFICIENT_SIZE	is returned if len bytes is not large enough to hold the entire name.
	In this case, only len bytes will be written.

5.2.2.6 rsmi_status_t rsmi_dev_subsystem_id_get (uint32_t dv_ind, uint16_t * id)

Get the subsystem device id associated with the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t id, this function will write the subsystem device id value to the uint64_t pointed to by id.

Parameters

in	dv_ind	a device index]
in,out	id	a pointer to uint64_t to which the subsystem device id will be written]

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.2.2.7 rsmi_status_t rsmi_dev_subsystem_name_get (uint32_t dv_ind, char * name, size_t len)

Get the name string for the device subsytem.

Given a device index dv_ind, a pointer to a caller provided char buffer name, and a length of this buffer len, this function will write the name of the device subsystem (up to len characters) to the buffer name.

If the integer ID associated with the sub-system is not found in one of the system files containing device name information (e.g. /usr/share/misc/pci.ids), then this function will return the hex sub-system ID as a string. Updating the system name files can be accompplished with "sudo update-pciids".

Parameters

in	dv_ind	a device index
in,out	name	a pointer to a caller provided char buffer to which the name will be written
in	len	the length of the caller provided buffer name.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
RSMI_STATUS_INSUFFICIENT_SIZE	is returned if len bytes is not large enough to hold the entire name.
	In this case, only len bytes will be written.

5.2.2.8 rsmi_status_t rsmi_dev_subsystem_vendor_id_get (uint32_t dv_ind, uint16_t * id)

Get the device subsystem vendor id associated with the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t id, this function will write the device subsystem vendor id value to the uint64_t pointed to by id.

Parameters

in	dv_ind	a device index
in,out	id	a pointer to uint64_t to which the device subsystem vendor id will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.2.2.9 rsmi status t rsmi_dev_unique_id_get (uint32_t dv_ind, uint64_t * id)

Get Unique ID.

Given a device index dv_ind and a pointer to a uint64_t id, this function will write the unique ID of the GPU pointed to id.

5.2 Identifier Queries 17

Parameters

in	dv_ind	a device index
in,out	id	a pointer to uint64_t to which the unique ID of the GPU is written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.3 PCle Queries

Functions

• rsmi_status_t rsmi_dev_pci_bandwidth_get (uint32_t dv_ind, rsmi_pcie_bandwidth_t *bandwidth)

Get the list of possible PCIe bandwidths that are available.

• rsmi_status_t rsmi_dev_pci_id_get (uint32_t dv_ind, uint64_t *bdfid)

Get the unique PCI device identifier associated for a device.

rsmi_status_t rsmi_dev_pci_throughput_get (uint32_t dv_ind, uint64_t *sent, uint64_t *received, uint64_←
t *max_pkt_sz)

Get PCIe traffic information.

rsmi_status_t rsmi_dev_pci_replay_counter_get (uint32_t dv_ind, uint64_t *counter)
 Get PCle replay counter.

5.3.1 Detailed Description

These functions provide information about PCIe.

5.3.2 Function Documentation

5.3.2.1 rsmi_status_t rsmi_dev_pci_bandwidth_get (uint32_t dv_ind, rsmi_pcie_bandwidth_t * bandwidth)

Get the list of possible PCIe bandwidths that are available.

Given a device index dv_ind and a pointer to a to an rsmi_pcie_bandwidth_t structure bandwidth, this function will fill in bandwidth with the possible T/s values and associated number of lanes, and indication of the current selection.

Parameters

in	dv_ind	a device index	
in,out	bandwidth	a pointer to a caller provided rsmi_pcie_bandwidth_t structure to which the frequency	
		information will be written	

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.3.2.2 rsmi_status_t rsmi_dev_pci_id_get (uint32_t dv_ind, uint64_t * bdfid)

Get the unique PCI device identifier associated for a device.

Give a device index dv_ind and a pointer to a uint64_t bdfid, this function will write the Bus/Device/Function PCI identifier (BDFID) associated with device dv_ind to the value pointed to by bdfid.

5.3 PCIe Queries

Parameters

in	dv_ind	a device index
in,out	bdfid	a pointer to uint64_t to which the device bdfid value will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.3.2.3 $rsmi_status_t rsmi_dev_pci_throughput_get (uint32_t dv_ind, uint64_t * sent, uint64_t * received, uint64_t * max_pkt_sz)$

Get PCIe traffic information.

Give a device index dv_ind and pointers to a uint64_t's, sent, received and max_pkt_sz , this function will write the number of bytes sent and received in 1 second to sent and received, respectively. The maximum possible packet size will be written to max_pkt_sz .

Parameters

in	dv_ind	a device index
in,out	sent	a pointer to uint64_t to which the number of bytes sent will be written in 1 second. If pointer is NULL, it will be ignored.
in,out	received	a pointer to uint64_t to which the number of bytes received will be written. If pointer is NULL, it will be ignored.
in,out	max_pkt_sz	a pointer to uint64_t to which the maximum packet size will be written. If pointer is NULL, it will be ignored.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.3.2.4 rsmi_status_t rsmi_dev_pci_replay_counter_get (uint32_t dv_ind, uint64_t * counter)

Get PCIe replay counter.

Given a device index dv_ind and a pointer to a uint64_t counter, this function will write the sum of the number of NAK's received by the GPU and the NAK's generated by the GPU to memory pointed to by counter.

Parameters

in	dv_ind	a device index
in,out	counter	a pointer to uint64_t to which the sum of the NAK's received and generated by the GPU
		is written

Return values

RSMI STATUS SUCCESS	is returned upon successful call.
	is retained apon daggeran cam

5.4 PCle Control

Functions

rsmi_status_t rsmi_dev_pci_bandwidth_set (uint32_t dv_ind, uint64_t bw_bitmask)
 Control the set of allowed PCle bandwidths that can be used.

5.4.1 Detailed Description

These functions provide some control over PCIe.

5.4.2 Function Documentation

5.4.2.1 rsmi status t rsmi dev pci bandwidth set (uint32 t dv ind, uint64 t bw bitmask)

Control the set of allowed PCIe bandwidths that can be used.

Given a device index dv_ind and a 64 bit bitmask bw_bitmask, this function will limit the set of allowable bandwidths. If a bit in bw_bitmask has a value of 1, then the frequency (as ordered in an rsmi_frequencies_t returned by rsmi_dev_gpu_clk_freq_get()) corresponding to that bit index will be allowed.

This function will change the performance level to RSMI_DEV_PERF_LEVEL_MANUAL in order to modify the set of allowable band_widths. Caller will need to set to RSMI_DEV_PERF_LEVEL_AUTO in order to get back to default state.

All bits with indices greater than or equal to the value of the rsmi_frequencies_t::num_supported field of rsmi_\top pcie_bandwidth_t will be ignored.

Parameters

in	dv_ind	a device index
in	bw_bitmask	j (,)
		(0). Only the lowest rsmi_frequencies_t::num_supported (of rsmi_pcie_bandwidth_t) bits
		of this mask are relevant.

5.5 Power Queries 21

5.5 Power Queries

Functions

• rsmi_status_t rsmi_dev_power_ave_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *power)

Get the average power consumption of the device with provided device index.

• rsmi_status_t rsmi_dev_power_cap_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *cap)

Get the cap on power which, when reached, causes the system to take action to reduce power.

• rsmi_status_t rsmi_dev_power_cap_range_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *max, uint64_t *min)

Get the range of valid values for the power cap.

5.5.1 Detailed Description

These functions provide information about power usage.

5.5.2 Function Documentation

```
5.5.2.1 rsmi_status_trsmi_dev_power_ave_get( uint32_t dv_ind, uint32_t sensor_ind, uint64_t * power)
```

Get the average power consumption of the device with provided device index.

Given a device index dv_ind and a pointer to a uint64_t power, this function will write the current average power consumption (in microwatts) to the uint64_t pointed to by power.

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one
		sensor, it could be greater than 0.
in,out	power	a pointer to uint64_t to which the average power consumption will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.5.2.2 rsmi_status_t rsmi_dev_power_cap_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t * cap)

Get the cap on power which, when reached, causes the system to take action to reduce power.

When power use rises above the value power, the system will take action to reduce power use. The power level returned through power will be in microWatts.

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one
		sensor, it could be greater than 0.
Generated by Doxygan		a pointer to a uint64_t that indicates the power cap, in microwatts

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.5.2.3 rsmi_status_t rsmi_dev_power_cap_range_get (uint32_t dv_i uint32_t $sensor_i$ uint64_t * max, uint64_t * min)

Get the range of valid values for the power cap.

This function will return the maximum possible valid power cap max and the minimum possible valid power cap min

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0.
in,out	max	a pointer to a uint64_t that indicates the maximum possible power cap, in microwatts
in,out	min	a pointer to a uint64_t that indicates the minimum possible power cap, in microwatts

Return values

RSMI STATUS SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.6 Power Control 23

5.6 Power Control

Functions

• rsmi_status_t rsmi_dev_power_cap_set (uint32_t dv_ind, uint32_t sensor_ind, uint64_t cap)

Set the power cap value.

rsmi_status_t rsmi_dev_power_profile_set (uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_profile_← preset_masks_t profile)

Set the power profile.

5.6.1 Detailed Description

These functions provide ways to control power usage.

5.6.2 Function Documentation

5.6.2.1 rsmi_status_t rsmi_dev_power_cap_set (uint32_t dv_ind, uint32_t sensor_ind, uint64_t cap)

Set the power cap value.

This function will set the power cap to the provided value cap. cap must be between the minimum and maximum power cap values set by the system, which can be obtained from rsmi_dev_power_cap_range_get.

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one
		sensor, it could be greater than 0.
in,out	cap	a uint64_t that indicates the desired power cap, in microwatts

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.6.2.2 rsmi_status_t rsmi_dev_power_profile_set (uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_profile_preset_masks_t profile)

Set the power profile.

Given a device index dv_{ind} , a sensor index sensor_ind, and a profile, this function will attempt to set the current profile to the provided profile. The provided profile must be one of the currently supported profiles, as indicated by a call to rsmi_dev_power_profile_presets_get()

Parameters

in	dv_ind	a device index	
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it	
		could be greater than 0.	
Generat	ed <i>NyOfilR</i> ygen	a rsmi_power_profile_preset_masks_t that hold the mask of the desired new power profile	

Return values

RSMI_STATUS_SUCCESS is returned upon successful call.

5.7 Memory Queries 25

5.7 Memory Queries

Functions

rsmi_status_t rsmi_dev_memory_total_get (uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_← t *total)

Get the total amount of memory that exists.

rsmi_status_t rsmi_dev_memory_usage_get (uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_
 t *used)

Get the current memory usage.

5.7.1 Detailed Description

These functions provide information about memory systems.

5.7.2 Function Documentation

5.7.2.1 rsmi_status_t rsmi_dev_memory_total_get (uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_t * total)

Get the total amount of memory that exists.

Given a device index dv_ind, a type of memory mem_type, and a pointer to a uint64_t total, this function will write the total amount of mem_type memory that exists to the location pointed to by total.

Parameters

in	dv_ind	a device index
in	mem_type	The type of memory for which the total amount will be found
in, out	total	a pointer to uint64_t to which the total amount of memory will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.7.2.2 rsmi_status_t rsmi_dev_memory_usage_get (uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_t * used)

Get the current memory usage.

Given a device index dv_ind, a type of memory mem_type, and a pointer to a uint64_t usage, this function will write the amount of mem_type memory that that is currently being used to the location pointed to by total.

Parameters

in	dv_ind	a device index
in	mem_type	The type of memory for which the amount being used will be found
in,out	used	a pointer to uint64_t to which the amount of memory currently being used will be written

Return values

RSMI_STATUS_SUCCESS is returned upon successful call.

5.8 Physical State Queries

Functions

- rsmi_status_t rsmi_dev_fan_rpms_get (uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed)

 Get the fan speed in RPMs of the device with the specified device index and 0-based sensor index.
- rsmi_status_t rsmi_dev_fan_speed_get (uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed)

 Get the fan speed for the specified device as a value relative to RSMI_MAX_FAN_SPEED.
- rsmi_status_t rsmi_dev_fan_speed_max_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *max_speed)

 Get the max. fan speed of the device with provided device index.
- rsmi_status_t rsmi_dev_temp_metric_get (uint32_t dv_ind, uint32_t sensor_type, rsmi_temperature_metric
 — t metric, int64_t *temperature)

Get the temperature metric value for the specified metric, from the specified temperature sensor on the specified device.

5.8.1 Detailed Description

These functions provide information about the physical characteristics of the device.

5.8.2 Function Documentation

5.8.2.1 rsmi_status_t rsmi_dev_fan_rpms_get (uint32_t dv_ind, uint32_t sensor_ind, int64_t * speed)

Get the fan speed in RPMs of the device with the specified device index and 0-based sensor index.

Given a device index dv_ind and a pointer to a uint32_t speed, this function will write the current fan speed in RPMs to the uint32_t pointed to by speed

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one
		sensor, it could be greater than 0.
in,out	speed	a pointer to uint32_t to which the speed will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.8.2.2 rsmi_status_trsmi_dev_fan_speed_get(_uint32_t dv_ind, uint32_t sensor_ind, int64_t * speed_)

Get the fan speed for the specified device as a value relative to RSMI_MAX_FAN_SPEED.

Given a device index dv_ind and a pointer to a uint32_t speed, this function will write the current fan speed (a value between 0 and the maximum fan speed, RSMI_MAX_FAN_SPEED) to the uint32_t pointed to by speed

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one
		sensor, it could be greater than 0.
in,out	speed	a pointer to uint32_t to which the speed will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.8.2.3 rsmi status t rsmi_dev_fan_speed_max_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t * max_speed)

Get the max. fan speed of the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t max_speed, this function will write the maximum fan speed possible to the uint32_t pointed to by max_speed

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one
		sensor, it could be greater than 0.
in,out	max_speed	a pointer to uint32_t to which the maximum speed will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.8.2.4 rsmi_status_t rsmi_dev_temp_metric_get (uint32_t dv_ind, uint32_t sensor_type, rsmi_temperature_metric_t metric, int64_t * temperature)

Get the temperature metric value for the specified metric, from the specified temperature sensor on the specified device.

Given a device index dv_ind, a sensor type sensor_type, a rsmi_temperature_metric_t metric and a pointer to an int64_t temperature, this function will write the value of the metric indicated by metric and sensor—type to the memory location temperature.

Parameters

in	dv_ind	a device index
in	sensor_type	part of device from which temperature should be obtained. This should come from
		the enum rsmi_temperature_type_t
in	metric	enum indicated which temperature value should be retrieved
in,out	temperature	a pointer to int64_t to which the temperature will be written, in millidegrees Celcius.

Return values

RSMI_STATUS_SUCCESS | is returned upon successful call.

5.9 Physical State Control

Functions

• rsmi_status_t rsmi_dev_fan_reset (uint32_t dv_ind, uint32_t sensor_ind)

Reset the fan to automatic driver control.

• rsmi_status_t rsmi_dev_fan_speed_set (uint32_t dv_ind, uint32_t sensor_ind, uint64_t speed)

Set the fan speed for the specified device with the provided speed, in RPMs.

5.9.1 Detailed Description

These functions provide control over the physical state of a device.

5.9.2 Function Documentation

5.9.2.1 rsmi_status_t rsmi_dev_fan_reset (uint32_t dv_ind, uint32_t sensor_ind)

Reset the fan to automatic driver control.

This function returns control of the fan to the system

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it
		could be greater than 0.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.9.2.2 rsmi_status_t rsmi_dev_fan_speed_set (uint32_t dv_ind, uint32_t sensor_ind, uint64_t speed)

Set the fan speed for the specified device with the provided speed, in RPMs.

Given a device index dv_ind and a integer value indicating speed speed, this function will attempt to set the fan speed to speed. An error will be returned if the specified speed is outside the allowable range for the device. The maximum value is 255 and the minimum is 0.

Parameters

in	dv_ind	a device index	
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0.	
in	speed the speed to which the function will attempt to set the fan		

Return values

RSMI_STATUS_SUCCESS is returned upon successful call.

5.10 Clock, Power and Performance Queries

Functions

rsmi_status_t rsmi_dev_busy_percent_get (uint32_t dv_ind, uint32_t *busy_percent)

Get percentage of time device is busy doing any processing.

• rsmi_status_t rsmi_dev_perf_level_get (uint32_t dv_ind, rsmi_dev_perf_level_t *perf)

Get the performance level of the device with provided device index.

rsmi status t rsmi dev overdrive level get (uint32 t dv ind, uint32 t *od)

Get the overdrive percent associated with the device with provided device index.

• rsmi_status_t rsmi_dev_gpu_clk_freq_get (uint32_t dv_ind, rsmi_clk_type_t clk_type, rsmi_frequencies_t *f)

Get the list of possible system clock speeds of device for a specified clock type.

rsmi_status_t rsmi_dev_od_volt_info_get (uint32_t dv_ind, rsmi_od_volt_freq_data_t *odv)

This function retrieves the voltage/frequency curve information.

rsmi_status_t rsmi_dev_od_volt_curve_regions_get (uint32_t dv_ind, uint32_t *num_regions, rsmi_freq_
volt region t *buffer)

This function will retrieve the current valid regions in the frequency/voltage space.

rsmi_status_t rsmi_dev_power_profile_presets_get (uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_
 profile_status_t *status)

Get the list of available preset power profiles and an indication of which profile is currently active.

5.10.1 Detailed Description

These functions provide information about clock frequencies and performance.

5.10.2 Function Documentation

5.10.2.1 rsmi_status_t rsmi_dev_busy_percent_get (uint32_t dv_ind, uint32_t * busy_percent)

Get percentage of time device is busy doing any processing.

Given a device index dv_ind, this function returns the percentage of time that the specified device is busy. The device is considered busy if any one or more of its sub-blocks are working, and idle if none of the sub-blocks are working.

Parameters

in	dv_ind	a device index
in,out	busy_percent	a pointer to the uint32_t to which the busy percent will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call

5.10.2.2 rsmi status trsmi_dev_perf_level_get (uint32_t dv_ind, rsmi dev perf_level_t * perf)

Get the performance level of the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t perf, this function will write the rsmi_dev_perf_level_t to the uint32_t pointed to by perf

Parameters

in	dv_ind	a device index
in,out	perf	a pointer to rsmi_dev_perf_level_t to which the performance level will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.10.2.3 rsmi_status_t rsmi_dev_overdrive_level_get (uint32_t dv_ind, uint32_t * od)

Get the overdrive percent associated with the device with provided device index.

Given a device index dv_ind and a pointer to a uint32_t od, this function will write the overdrive percentage to the uint32_t pointed to by od

Parameters

in	dv_ind	a device index
in,o	ut od	a pointer to uint32_t to which the overdrive percentage will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.

5.10.2.4 rsmi_status_t rsmi_dev_gpu_clk_freq_get (uint32_t dv_i nd, rsmi_clk_type_t clk_t ype, rsmi_frequencies_t *f)

Get the list of possible system clock speeds of device for a specified clock type.

Given a device index dv_ind, a clock type clk_type, and a pointer to a to an rsmi_frequencies_t structure f, this function will fill in f with the possible clock speeds, and indication of the current clock speed selection.

Parameters

in	dv_ind	a device index
in	clk_type	the type of clock for which the frequency is desired
in,out	f	a pointer to a caller provided rsmi_frequencies_t structure to which the frequency
		information will be written. Frequency values are in Hz.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.

5.10.2.5 rsmi_status_t rsmi_dev_od_volt_info_get (uint32_t dv_ind, rsmi_od_volt_freq_data_t * odv)

This function retrieves the voltage/frequency curve information.

Given a device index dv_ind and a pointer to a rsmi_od_volt_freq_data_t structure odv, this function will populate odv. See rsmi_od_volt_freq_data_t for more details.

Parameters

in	dv_ind	a device index
in	odv	a pointer to an rsmi_od_volt_freq_data_t structure

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

```
5.10.2.6 rsmi_status_t rsmi_dev_od_volt_curve_regions_get ( uint32_t dv_ind, uint32_t * num_iregions, rsmi_freq_volt_region_t * buffer)
```

This function will retrieve the current valid regions in the frequency/voltage space.

Given a device index dv_ind, a pointer to an unsigned integer num_regions and a buffer of rsmi_freq_volt caller structures, buffer, this function will populate buffer with the current frequency-volt space regions. The caller should assign buffer to memory that can be written to by this function. The caller should also indicate the number of rsmi_freq_volt_region_t structures that can safely be written to buffer in num_regions.

The number of regions to expect this function provide (num_regions) can be obtained by calling rsmi_dev_od—volt_info_get().

Parameters

in	dv_ind	a device index
in,out	num_regions	As input, this is the number of rsmi_freq_volt_region_t structures that can be written to buffer. As output, this is the number of rsmi_freq_volt_region_t structures that were actually written.
in,out	buffer	a caller provided buffer to which rsmi_freq_volt_region_t structures will be written

Return values

```
RSMI_STATUS_SUCCESS | is returned upon successful call.
```

```
5.10.2.7 rsmi_status_t rsmi_dev_power_profile_presets_get ( uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_profile_status_t * status_)
```

Get the list of available preset power profiles and an indication of which profile is currently active.

Given a device index dv_ind and a pointer to a rsmi_power_profile_status_t status, this function will set the bits of the rsmi_power_profile_status_t.available_profiles bit field of status to 1 if the profile corresponding to the

respective rsmi_power_profile_preset_masks_t profiles are enabled. For example, if both the VIDEO and VR power profiles are available selections, then RSMI_PWR_PROF_PRST_VIDEO_MASK AND'ed with rsmi_power_profile status_t.available_profiles will be non-zero as will RSMI_PWR_PROF_PRST_VR_MASK AND'ed with rsmi_cower_profile_status_t.available_profiles. Additionally, rsmi_power_profile_status_t.current will be set to the rsmicower_profile_preset_masks_t of the profile that is currently active.

Parameters

in	dv_ind	a device index
in	sensor_ind	a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0.
in,out	status	a pointer to rsmi_power_profile_status_t that will be populated by a call to this function

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.11 Clock, Power and Performance Control

Functions

- rsmi_status_t rsmi_dev_perf_level_set (int32_t dv_ind, rsmi_dev_perf_level_t perf_lvl)
 - Set the PowerPlay performance level associated with the device with provided device index with the provided value.
- rsmi_status_t rsmi_dev_overdrive_level_set (int32_t dv_ind, uint32_t od)
 - Set the overdrive percent associated with the device with provided device index with the provided value. See details for WARNING.
- rsmi_status_t rsmi_dev_gpu_clk_freq_set (uint32_t dv_ind, rsmi_clk_type_t clk_type, uint64_t freq_bitmask)

 Control the set of allowed frequencies that can be used for the specified clock.
- rsmi_status_t rsmi_dev_od_freq_range_set (uint32_t dv_ind, rsmi_clk_type_t clk, rsmi_range_t *range)

 Set the frequency limits for the specified clock.

5.11.1 Detailed Description

These functions provide control over clock frequencies, power and performance.

5.11.2 Function Documentation

5.11.2.1 rsmi_status_t rsmi_dev_perf_level_set (int32_t dv_ind, rsmi_dev_perf_level_t perf_lvl)

Set the PowerPlay performance level associated with the device with provided device index with the provided value.

Given a device index dv_ind and an rsmi_dev_perf_level_t perf_level, this function will set the PowerPlay performance level for the device to the value perf_lvl.

Parameters

in	dv_ind	a device index
in	perf←	the value to which the performance level should be set
	_lvl	

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.11.2.2 rsmi_status_t rsmi_dev_overdrive_level_set (int32_t dv_ind, uint32_t od)

Set the overdrive percent associated with the device with provided device index with the provided value. See details for WARNING.

Given a device index dv_ind and an overdrive level od, this function will set the overdrive level for the device to the value od. The overdrive level is an integer value between 0 and 20, inclusive, which represents the overdrive percentage; e.g., a value of 5 specifies an overclocking of 5%.

The overdrive level is specific to the gpu system clock.

The overdrive level is the percentage above the maximum Performance Level to which overclocking will be limited. The overclocking percentage does not apply to clock speeds other than the maximum. This percentage is limited to 20%.

******WARNING****** Operating your AMD GPU outside of official AMD specifications or outside of factory settings, including but not limited to the conducting of overclocking (including use of this overclocking software, even if such software has been directly or indirectly provided by AMD or otherwise affiliated in any way with AMD), may cause damage to your AMD GPU, system components and/or result in system failure, as well as cause other problems. DAMAGES CAUSED BY USE OF YOUR AMD GPU OUTSIDE OF OFFICIAL AMD SPECIFICATIONS OR OUTSIDE OF FACTORY SETTINGS ARE NOT COVERED UNDER ANY AMD PRODUCT WARRANTY ACHOND MAY NOT BE COVERED BY YOUR BOARD OR SYSTEM MANUFACTURER'S WARRANTY. Please use this utility with caution.

Parameters

in	dv_ind	a device index
in	od	the value to which the overdrive level should be set

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.11.2.3 rsmi_status_t rsmi_dev_gpu_clk_freq_set (uint32_t dv_ind, rsmi_clk_type_t clk_type, uint64_t freq_bitmask)

Control the set of allowed frequencies that can be used for the specified clock.

Given a device index dv_ind, a clock type clk_type, and a 64 bit bitmask freq_bitmask, this function will limit the set of allowable frequencies. If a bit in freq_bitmask has a value of 1, then the frequency (as ordered in an rsmi_frequencies_t returned by rsmi_dev_gpu_clk_freq_get()) corresponding to that bit index will be allowed.

This function will change the performance level to RSMI_DEV_PERF_LEVEL_MANUAL in order to modify the set of allowable frequencies. Caller will need to set to RSMI_DEV_PERF_LEVEL_AUTO in order to get back to default state.

All bits with indices greater than or equal to rsmi frequencies t::num supported will be ignored.

Parameters

in	dv_ind	a device index
in	clk_type	the type of clock for which the set of frequencies will be modified
in	freq_bitmask	A bitmask indicating the indices of the frequencies that are to be enabled (1) and disabled (0). Only the lowest rsmi_frequencies_t.num_supported bits of this mask are relevant.

5.11.2.4 rsmi_status_t rsmi_dev_od_freq_range_set (uint32_t dv_ind, rsmi_clk_type_t clk, rsmi_range_t * range)

Set the frequency limits for the specified clock.

Given a device index dv_{ind} , a clock type (rsmi_clk_type_t) clk, and a pointer to a rsmi_range_t range containing the desired upper and lower frequency limits, this function will attempt to set the frequency limits to those specified in range.

Parameters

in	dv_ind	a device index
in	clk	The clock type for which the limits should be imposed.
in	range	A pointer to the rsmi_range_t containing the desired limits

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.12 Version Queries 39

5.12 Version Queries

Functions

rsmi_status_t rsmi_version_get (rsmi_version_t *version)

Get the build version information for the currently running build of RSMI.

- rsmi_status_t rsmi_version_str_get (rsmi_sw_component_t component, char *ver_str, uint32_t len)

 Get the driver version string for the current system.
- rsmi_status_t rsmi_dev_vbios_version_get (uint32_t dv_ind, char *vbios, uint32_t len)
 Get the VBIOS identifer string.

5.12.1 Detailed Description

These functions provide version information about various subsystems.

5.12.2 Function Documentation

5.12.2.1 rsmi_status_trsmi_version_get(rsmi_version_t * version)

Get the build version information for the currently running build of RSMI.

Get the major, minor, patch and build string for RSMI build currently in use through version

Parameters

in,out	version	A pointer to an rsmi_version_t structure that will be updated with the version information
		upon return.

Return values

RSMI STATUS SUCCESS	is returned upon successful call
	is retained apon eacestain can

5.12.2.2 rsmi_status_t rsmi_version_str_get (rsmi_sw_component_t component, char * ver_str, uint32_t len)

Get the driver version string for the current system.

Given a software component component, a pointer to a char buffer, <code>ver_str</code>, this function will write the driver version string (up to <code>len</code> characters) for the current system to <code>ver_str</code>. The caller must ensure that it is safe to write at least <code>len</code> characters to <code>ver_str</code>.

Parameters

in	component	The component for which the version string is being requested
in,out	ver_str	A pointer to a buffer of char's to which the VBIOS name will be written
in	len	The number of char's pointed to by ver_str which can safely be written to by this function.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.12.2.3 rsmi_status_t rsmi_dev_vbios_version_get (uint32_t dv_ind, char * vbios, uint32_t len)

Get the VBIOS identifer string.

Given a device ID dv_{ind} , and a pointer to a char buffer, vbios, this function will write the VBIOS string (up to len characters) for device dv_{ind} to vbios. The caller must ensure that it is safe to write at least len characters to vbios.

Parameters

in	dv_ind	a device index
in, out	vbios	A pointer to a buffer of char's to which the VBIOS name will be written
in	len	The number of char's pointed to by <code>vbios</code> which can safely be written to by this function.

Return values

DOME CTATUS SUSSESS	in waterway along a consequent of a life
HOWI STATUS SUCCESS	is returned upon successful call.

5.13 Error Queries 41

5.13 Error Queries

Functions

• rsmi_status_t rsmi_dev_ecc_count_get (uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_error_count_t *ec)

Retrieve the error counts for a GPU block.

rsmi_status_t rsmi_dev_ecc_enabled_get (uint32_t dv_ind, uint64_t *enabled_mask)
 Retrieve the enabled ECC bit-mask.

rsmi_status_t rsmi_dev_ecc_status_get (uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_ras_err_state_← t *state)

Retrieve the ECC status for a GPU block.

• rsmi_status_t rsmi_status_string (rsmi_status_t status, const char **status_string)

Get a description of a provided RSMI error status.

5.13.1 Detailed Description

These functions provide error information about RSMI calls as well as device errors.

5.13.2 Function Documentation

5.13.2.1 rsmi_status_t rsmi_dev_ecc_count_get (uint32_t dv_ind , rsmi_gpu_block_t block, rsmi_error_count_t * ec)

Retrieve the error counts for a GPU block.

Given a device index dv_ind, an rsmi_gpu_block_t block and a pointer to an rsmi_error_count_tec, this function will write the error count values for the GPU block indicated by block to memory pointed to by ec.

Parameters

in	dv_ind	a device index
in	block	The block for which error counts should be retrieved
in,out	ec	A pointer to an rsmi_error_count_t to which the error counts should be written

Return values

RSMI STATUS SUCCESS	is returned upon successful call.
	io rotairioù apori oacocorai cam

5.13.2.2 rsmi_status_trsmi_dev_ecc_enabled_get(_uint32_t dv_ind, uint64_t * enabled_mask_)

Retrieve the enabled ECC bit-mask.

Given a device index dv_ind, and a pointer to a uint64_t enabled_mask, this function will write a bit_mask to memory pointed to by enabled_mask. Upon a successful call, the bitmask can then be AND'd with elements of the rsmi_gpu_block_t ennumeration to determine if the corresponding block has ECC enabled. Note that the bits above RSMI_GPU_BLOCK_LAST correspond to blocks that do not yet have ECC support.

Parameters

in	dv_ind	a device index
in,out	enabled_mask	A pointer to a uint64_t to which the enabled mask will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.
---------------------	-----------------------------------

5.13.2.3 rsmi_status_t rsmi_dev_ecc_status_get (uint32_t dv_i nd, rsmi_gpu_block_t block, rsmi_ras_err_state_t * state)

Retrieve the ECC status for a GPU block.

Given a device index dv_ind, an rsmi_gpu_block_t block and a pointer to an rsmi_ras_err_state_t state, this function will write the current state for the GPU block indicated by block to memory pointed to by state.

Parameters

in	dv_ind	a device index
in	block	The block for which error counts should be retrieved
in,out	state	A pointer to an rsmi_ras_err_state_t to which the ECC state should be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call.

5.13.2.4 rsmi_status_t rsmi_status_string (rsmi_status_t status, const char ** status_string)

Get a description of a provided RSMI error status.

Set the provided pointer to a const char *, status_string, to a string containing a description of the provided error code status.

Parameters

in	status	The error status for which a description is desired
in,out	status_string	A pointer to a const char * which will be made to point to a description of the provided error code

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call
---------------------	----------------------------------

5.14 Performance Counter Functions

Functions

- rsmi_status_t rsmi_dev_counter_group_supported (uint32_t dv_ind, rsmi_event_group_t group)

 Tell if an event group is supported by a given device.
- rsmi_status_t rsmi_dev_counter_create (uint32_t dv_ind, rsmi_event_type_t type, rsmi_event_handle_← t *evnt_handle)

Create a performance counter object.

• rsmi_status_t rsmi_dev_counter_destroy (rsmi_event_handle_t evnt_handle)

Deallocate a performance counter object.

 rsmi_status_t rsmi_counter_control (rsmi_event_handle_t evt_handle, rsmi_counter_command_t cmd, void *cmd_args)

Issue performance counter control commands.

- rsmi_status_t rsmi_counter_read (rsmi_event_handle_t evt_handle, rsmi_counter_value_t *value)
 - Read the current value of a performance counter.
- rsmi_status_t rsmi_counter_available_counters_get (uint32_t dv_ind, rsmi_event_group_t grp, uint32_
 t *available)

Get the number of currently available counters.

5.14.1 Detailed Description

These functions are used to configure, query and control performance counting.

5.14.2 Function Documentation

5.14.2.1 rsmi_status_t rsmi_dev_counter_group_supported (uint32_t dv_ind, rsmi_event_group_t group)

Tell if an event group is supported by a given device.

Given a device index dv_ind and an event group specifier group, tell if group type events are supported by the device associated with dv_ind

Parameters

in	dv_ind	device index of device being queried
in	group	rsmi_event_group_t identifier of group for which support is being queried

Return values

RSMI_STATUS_SUCCESS	if the device associatee with dv_ind support counting events of the type
	indicated by group.

RSMI_STATUS_NOT_SUPPORTED If the device does not support event group group

5.14.2.2 rsmi_status_t rsmi_dev_counter_create (uint32_t dv_ind, rsmi_event_type_t type, rsmi_event_handle_t * evnt_handle)

Create a performance counter object.

Create a performance counter object of type type for the device with a device index of dv_ind , and write a handle to the object to the memory location pointed to by $evnt_handle$. $evnt_handle$ can be used with other performance event operations. The handle should be deallocated with $rsmi_dev_counter_destroy()$ when no longer needed.

Parameters

in	dv_ind	a device index
in	type	the type of performance event to create
in,out	evnt_handle	A pointer to a rsmi_event_handle_t which will be associated with a newly allocated counter

Return values

RSMI STATUS SUCCESS	is returned upon successful call
---------------------	----------------------------------

5.14.2.3 rsmi_status_t rsmi_dev_counter_destroy (rsmi_event_handle_t evnt_handle)

Deallocate a performance counter object.

Deallocate the performance counter object with the provided rsmi_event_handle_t evnt_handle

Parameters

in	evnt_handle	handle to event object to be deallocated
----	-------------	--

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call
---------------------	----------------------------------

5.14.2.4 rsmi_status_t rsmi_counter_control (rsmi_event_handle_t evt_handle, rsmi_counter_command_t cmd, void * cmd_args)

Issue performance counter control commands.

 $\label{lssue} \textbf{lssue a command cmd on the event counter associated with the provided handle \verb|evt_handle|.}$

Parameters

in	evt_handle	an event handle
in	cmd	The event counter command to be issued
in,out	cmd_args	Currently not used. Should be set to NULL.

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call
---------------------	----------------------------------

 $5.14.2.5 \quad rsmi_status_t \ rsmi_counter_read \ (\ rsmi_event_handle_t \ \textit{evt_handle}, \ rsmi_counter_value_t * \textit{value} \)$

Read the current value of a performance counter.

Read the current counter value of the counter associated with the provided handle evt_handle and write the value to the location pointed to by value.

Parameters

in	evt_handle	an event handle
in,out	value	pointer to memory of size of rsmi_counter_value_t to which the counter value will be
		written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call
---------------------	----------------------------------

5.14.2.6 rsmi_status_t rsmi_counter_available_counters_get (uint32_t dv_ind, rsmi_event_group_t grp, uint32_t * available)

Get the number of currently available counters.

Given a device index dv_{ind} , a performance event group grp, and a pointer to a uint32_t available, this function will write the number of grp type counters that are available on the device with index dv_{ind} to the memory that available points to.

Parameters

in	dv_ind	a device index
in	grp	an event device group
in,out	available	A pointer to a uint32_t to which the number of available counters will be written

Return values

RSMI_STATUS_SUCCESS	is returned upon successful call
---------------------	----------------------------------

Chapter 6

Data Structure Documentation

6.1 rsmi_counter_value_t Struct Reference

```
#include <rocm_smi.h>
```

Data Fields

• uint64_t value

Counter value.

uint64_t time_enabled

Time that the counter was enabled.

• uint64_t time_running

Time that che counter was running.

6.1.1 Detailed Description

Counter value

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

• rocm_smi.h

6.2 rsmi_error_count_t Struct Reference

This structure holds error counts.

```
#include <rocm_smi.h>
```

Data Fields

• uint64_t correctable_err

Accumulated correctable errors.

• uint64_t uncorrectable_err

Accumulated uncorrectable errors.

6.2.1 Detailed Description

This structure holds error counts.

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

· rocm smi.h

6.3 rsmi_freq_volt_region_t Struct Reference

This structure holds 2 rsmi_range_t's, one for frequency and one for voltage. These 2 ranges indicate the range of possible values for the corresponding rsmi_od_vddc_point_t.

```
#include <room_smi.h>
```

Data Fields

• rsmi_range_t freq_range

The frequency range for this VDDC Curve point.

rsmi_range_t volt_range

The voltage range for this VDDC Curve point.

6.3.1 Detailed Description

This structure holds 2 rsmi_range_t's, one for frequency and one for voltage. These 2 ranges indicate the range of possible values for the corresponding rsmi_od_vddc_point_t.

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

· rocm_smi.h

6.4 rsmi_frequencies_t Struct Reference

This structure holds information about clock frequencies.

```
#include <room_smi.h>
```

Data Fields

- uint32_t num_supported
- uint32 t current
- uint64_t frequency [RSMI_MAX_NUM_FREQUENCIES]

6.4.1 Detailed Description

This structure holds information about clock frequencies.

6.4.2 Field Documentation

6.4.2.1 uint32_t rsmi_frequencies_t::num_supported

The number of supported frequencies

6.4.2.2 uint32_t rsmi_frequencies_t::current

The current frequency index

6.4.2.3 uint64_t rsmi_frequencies_t::frequency[RSMI_MAX_NUM_FREQUENCIES]

List of frequencies. Only the first num_supported frequencies are valid.

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

• rocm_smi.h

6.5 rsmi_od_vddc_point_t Struct Reference

This structure represents a point on the frequency-voltage plane.

```
#include <rocm_smi.h>
```

Data Fields

uint64_t frequency

Frequency coordinate (in Hz)

• uint64_t voltage

Voltage coordinate (in mV)

6.5.1 Detailed Description

This structure represents a point on the frequency-voltage plane.

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

rocm_smi.h

6.6 rsmi_od_volt_curve_t Struct Reference

```
#include <rocm_smi.h>
```

Data Fields

rsmi_od_vddc_point_t vc_points [RSMI_NUM_VOLTAGE_CURVE_POINTS]

6.6.1 Detailed Description

RSMI_NUM_VOLTAGE_CURVE_POINTS number of rsmi_od_vddc_point_t's

6.6.2 Field Documentation

6.6.2.1 rsmi_od_vddc_point_t rsmi_od_volt_curve_t::vc_points[RSMI_NUM_VOLTAGE_CURVE_POINTS]

Array of RSMI_NUM_VOLTAGE_CURVE_POINTS rsmi_od_vddc_point_t's that make up the voltage frequency curve points.

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

· rocm smi.h

6.7 rsmi_od_volt_freq_data_t Struct Reference

This structure holds the frequency-voltage values for a device.

```
#include <rocm_smi.h>
```

Data Fields

• rsmi_range_t curr_sclk_range

The current SCLK frequency range.

- rsmi_range_t curr_mclk_range
- rsmi_range_t sclk_freq_limits

The range possible of SCLK values.

• rsmi_range_t mclk_freq_limits

The range possible of MCLK values.

• rsmi_od_volt_curve_t curve

The current voltage curve.

• uint32_t num_regions

The number of voltage curve regions.

6.7.1 Detailed Description

This structure holds the frequency-voltage values for a device.

6.7.2 Field Documentation

6.7.2.1 rsmi_range_t rsmi_od_volt_freq_data_t::curr_mclk_range

The current MCLK frequency range; (upper bound only)

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

rocm_smi.h

6.8 rsmi_pcie_bandwidth_t Struct Reference

This structure holds information about the possible PCIe bandwidths. Specifically, the possible transfer rates and their associated numbers of lanes are stored here.

```
#include <rocm_smi.h>
```

Data Fields

- · rsmi frequencies t transfer rate
- uint32_t lanes [RSMI_MAX_NUM_FREQUENCIES]

6.8.1 Detailed Description

This structure holds information about the possible PCIe bandwidths. Specifically, the possible transfer rates and their associated numbers of lanes are stored here.

6.8.2 Field Documentation

6.8.2.1 rsmi_frequencies_t rsmi_pcie_bandwidth_t::transfer_rate

Transfer rates (T/s) that are possible

6.8.2.2 uint32_t rsmi_pcie_bandwidth_t::lanes[RSMI_MAX_NUM_FREQUENCIES]

List of lanes for corresponding transfer rate. Only the first num_supported bandwidths are valid.

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

• rocm_smi.h

6.9 rsmi_power_profile_status_t Struct Reference

This structure contains information about which power profiles are supported by the system for a given device, and which power profile is currently active.

```
#include <rocm_smi.h>
```

Data Fields

- rsmi_bit_field_t available_profiles
- rsmi_power_profile_preset_masks_t current
- uint32 t num profiles

6.9.1 Detailed Description

This structure contains information about which power profiles are supported by the system for a given device, and which power profile is currently active.

6.9.2 Field Documentation

```
6.9.2.1 rsmi_bit_field_t rsmi_power_profile_status_t::available_profiles
```

Which profiles are supported by this system

 $6.9.2.2 \quad rsmi_power_profile_preset_masks_t \ rsmi_power_profile_status_t::current$

Which power profile is currently active

6.9.2.3 uint32_t rsmi_power_profile_status_t::num_profiles

How many power profiles are available

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

· rocm smi.h

6.10 rsmi_range_t Struct Reference

This structure represents a range (e.g., frequencies or voltages).

```
#include <rocm_smi.h>
```

Data Fields

```
    uint64_t lower_bound
```

Lower bound of range.

• uint64_t upper_bound

Upper bound of range.

6.10.1 Detailed Description

This structure represents a range (e.g., frequencies or voltages).

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

• rocm_smi.h

6.11 rsmi_version_t Struct Reference

This structure holds version information.

```
#include <rocm_smi.h>
```

Data Fields

• uint32_t major

Major version.

uint32_t minor

Minor version.

uint32_t patch

Patch, build or stepping version.

· const char * build

Build string.

6.11.1 Detailed Description

This structure holds version information.

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

· rocm_smi.h

Chapter 7

File Documentation

7.1 rocm_smi.h File Reference

The rocm_smi library api is new, and therefore subject to change either at the ABI or API level. Instead of marking every function prototype as "unstable", we are instead saying the API is unstable (i.e., changes are possible) while the major version remains 0. This means that if the API/ABI changes, we will not increment the major version to 1. Once the ABI stabilizes, we will increment the major version to 1, and thereafter increment it on all ABI breaks.

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

Data Structures

- · struct rsmi_counter_value_t
- · struct rsmi_power_profile_status_t

This structure contains information about which power profiles are supported by the system for a given device, and which power profile is currently active.

· struct rsmi_frequencies_t

This structure holds information about clock frequencies.

· struct rsmi pcie bandwidth t

This structure holds information about the possible PCle bandwidths. Specifically, the possible transfer rates and their associated numbers of lanes are stored here.

· struct rsmi_version_t

This structure holds version information.

· struct rsmi range t

This structure represents a range (e.g., frequencies or voltages).

struct rsmi_od_vddc_point_t

This structure represents a point on the frequency-voltage plane.

struct rsmi_freq_volt_region_t

This structure holds 2 rsmi_range_t's, one for frequency and one for voltage. These 2 ranges indicate the range of possible values for the corresponding rsmi_od_vddc_point_t.

- struct rsmi_od_volt_curve_t
- struct rsmi_od_volt_freq_data_t

This structure holds the frequency-voltage values for a device.

· struct rsmi_error_count_t

This structure holds error counts.

56 File Documentation

Macros

• #define RSMI_MAX_NUM_FREQUENCIES 32

Guaranteed maximum possible number of supported frequencies.

- #define RSMI MAX FAN SPEED 255
- #define RSMI_NUM_VOLTAGE_CURVE_POINTS 3

The number of points that make up a voltage-frequency curve definition.

• #define RSMI MAX NUM POWER PROFILES (sizeof(rsmi bit field t) * 8)

Number of possible power profiles that a system could support.

Typedefs

· typedef uintptr t rsmi event handle t

Handle to performance event counter.

typedef uint64_t rsmi_bit_field_t

Bitfield used in various RSMI calls.

Enumerations

enum rsmi_status_t {
 RSMI_STATUS_SUCCESS = 0x0, RSMI_STATUS_INVALID_ARGS, RSMI_STATUS_NOT_SUPPORTED,
 RSMI_STATUS_FILE_ERROR,
 RSMI_STATUS_PERMISSION, RSMI_STATUS_OUT_OF_RESOURCES, RSMI_STATUS_INTERNAL_

EXCEPTION, RSMI_STATUS_INPUT_OUT_OF_BOUNDS,
RSMI_STATUS_INIT_ERROR, RSMI_INITIALIZATION_ERROR = RSMI_STATUS_INIT_ERROR, RSMI

STATUS_NOT_YET_IMPLEMENTED, RSMI_STATUS_NOT_FOUND,

RSMI_STATUS_INSUFFICIENT_SIZE, RSMI_STATUS_INTERRUPT, RSMI_STATUS_UNEXPECTED_← SIZE, RSMI_STATUS_UNKNOWN_ERROR = 0xFFFFFFFF }

Error codes retured by rocm_smi_lib functions.

• enum rsmi init flags t { RSMI INIT FLAG ALL GPUS = 0x1 }

Initialization flags.

enum rsmi_dev_perf_level_t {

$$\label{eq:rsmldev_perf_level_auto} \begin{split} & \text{RSMI_DEV_PERF_LEVEL_AUTO} = 0, & \text{RSMI_DEV_PERF_LEVEL_FIRST} = & \text{RSMI_DEV_PERF_LEVEL_} \\ & \text{AUTO, RSMI_DEV_PERF_LEVEL_LOW, RSMI_DEV_PERF_LEVEL_HIGH,} \end{split}$$

 $RSMI_DEV_PERF_LEVEL_MANUAL, RSMI_DEV_PERF_LEVEL_STABLE_STD, RSMI_DEV_PERF_LE \leftrightarrow VEL_STABLE_PEAK, RSMI_DEV_PERF_LEVEL_STABLE_MIN_MCLK,$

RSMI_DEV_PERF_LEVEL_STABLE_MIN_SCLK, RSMI_DEV_PERF_LEVEL_LAST = RSMI_DEV_PER ← F LEVEL STABLE MIN SCLK, RSMI DEV PERF LEVEL UNKNOWN = 0x100 }

PowerPlay performance levels.

• enum rsmi_sw_component_t { RSMI_SW_COMP_FIRST = 0x0, RSMI_SW_COMP_DRIVER = RSMI_SW ← COMP_FIRST, RSMI_SW_COMP_LAST = RSMI_SW_COMP_DRIVER }

Available clock types.

enum rsmi_event_group_t { RSMI_EVNT_GRP_XGMI = 0, RSMI_EVNT_GRP_INVALID = 0xFFFFFFFF }

Enum denoting an event group. The value of the enum is the base value for all the event enums in the group.

enum rsmi_event_type_t {

RSMI_EVNT_FIRST = RSMI_EVNT_GRP_XGMI, RSMI_EVNT_XGMI_FIRST = RSMI_EVNT_GRP_XGMI, RSMI_EVNT_XGMI_0_NOP_TX = RSMI_EVNT_XGMI_FIRST, RSMI_EVNT_XGMI_0_REQUEST_TX, RSMI_EVNT_XGMI_0_RESPONSE_TX, RSMI_EVNT_XGMI_0_BEATS_TX, RSMI_EVNT_XGMI_1_NO↔ P_TX, RSMI_EVNT_XGMI_1_REQUEST_TX, RSMI_EVNT_XGMI_1_RESPONSE_TX, RSMI_EVNT_XGMI_1_BEATS_TX, RSMI_EVNT_XGMI_1_BEATS_TX, RSMI_EVNT_XGMI_LAST = RSMI_EVNT_XGMI_1_BEATS_TX, RSMI_EVNT_LAST = RSMI_EVNT_XGMI_LAST }

Event type enum. Events belonging to a particular event group rsmi_event_group_t should begin ennumerating at the rsmi_event_group_t value for that group.

```
• enum rsmi_counter_command_t { RSMI_CNTR_CMD_START = 0, RSMI_CNTR_CMD_STOP }
```

enum rsmi clk type t {

RSMI_CLK_TYPE_SYS = 0x0, **RSMI_CLK_TYPE_FIRST** = RSMI_CLK_TYPE_SYS, RSMI_CLK_TYPE_↔ DF, RSMI_CLK_TYPE_DCEF,

RSMI_CLK_TYPE_SOC, RSMI_CLK_TYPE_MEM, RSMI_CLK_TYPE_LAST = RSMI_CLK_TYPE_MEM, RSMI_CLK_INVALID = 0xFFFFFFFF }

enum rsmi temperature metric t {

RSMI_TEMP_CURRENT = 0x0, RSMI_TEMP_FIRST = RSMI_TEMP_CURRENT, RSMI_TEMP_MAX, R↔ SMI_TEMP_MIN.

RSMI_TEMP_MAX_HYST, RSMI_TEMP_MIN_HYST, RSMI_TEMP_CRITICAL, RSMI_TEMP_CRITICAL \leftrightarrow HYST,

RSMI_TEMP_EMERGENCY, RSMI_TEMP_EMERGENCY_HYST, RSMI_TEMP_CRIT_MIN, RSMI_TEM→ P CRIT_MIN_HYST,

 $RSMI_TEMP_OFFSET, RSMI_TEMP_LOWEST, RSMI_TEMP_HIGHEST, \textbf{RSMI_TEMP_LAST} = RSMI_ \leftarrow TEMP_HIGHEST\}$

Temperature Metrics. This enum is used to identify various temperature metrics. Corresponding values will be in millidegress Celcius.

enum rsmi_temperature_type_t {

RSMI_TEMP_TYPE_FIRST = 0, RSMI_TEMP_TYPE_EDGE = RSMI_TEMP_TYPE_FIRST, RSMI_TEMP ← _ TYPE_JUNCTION, RSMI_TEMP_TYPE_MEMORY,

RSMI_TEMP_TYPE_LAST = RSMI_TEMP_TYPE_MEMORY }

This ennumeration is used to indicate from which part of the device a temperature reading should be obtained.

enum rsmi_power_profile_preset_masks_t {

RSMI_PWR_PROF_PRST_CUSTOM_MASK = 0x1, RSMI_PWR_PROF_PRST_VIDEO_MASK = 0x2, R⇔ SMI_PWR_PROF_PRST_POWER_SAVING_MASK = 0x4, RSMI_PWR_PROF_PRST_COMPUTE_MASK = 0x8.

RSMI_PWR_PROF_PRST_VR_MASK = 0x10, RSMI_PWR_PROF_PRST_3D_FULL_SCR_MASK = 0x20, RSMI_PWR_PROF_PRST_BOOTUP_DEFAULT = 0x40, RSMI_PWR_PROF_PRST_LAST = RSMI_PW⇔ R PROF_PRST_BOOTUP_DEFAULT.

Pre-set Profile Selections. These bitmasks can be AND'd with the rsmi_power_profile_status_t.available_profiles returned from rsmi_dev_power_profile_presets_get() to determine which power profiles are supported by the system.

enum rsmi_gpu_block_t {

This enum is used to identify different GPU blocks.

• enum rsmi ras err state t {

RSMI_RAS_ERR_STATE_NONE = 0, RSMI_RAS_ERR_STATE_DISABLED, RSMI_RAS_ERR_STATE ← _ PARITY, RSMI_RAS_ERR_STATE_SING_C,

RSMI_RAS_ERR_STATE_MULT_UC, RSMI_RAS_ERR_STATE_POISON, RSMI_RAS_ERR_STATE_L ← AST = RSMI_RAS_ERR_STATE_POISON, RSMI_RAS_ERR_STATE_INVALID = 0xFFFFFFFF }

The current ECC state.

enum rsmi_memory_type_t {

RSMI_MEM_TYPE_FIRST = 0, RSMI_MEM_TYPE_VRAM = RSMI_MEM_TYPE_FIRST, RSMI_MEM_T↔ YPE VIS VRAM, RSMI_MEM_TYPE_GTT,

RSMI MEM TYPE LAST = RSMI MEM TYPE GTT }

Types of memory.

enum rsmi_freq_ind_t { RSMI_FREQ_IND_MIN = 0, RSMI_FREQ_IND_MAX = 1, RSMI_FREQ_IND_INV
 ALID = 0xFFFFFFFF}

This values of this enum are used as frequency identifiers.

58 **File Documentation**

Functions

rsmi_status_t rsmi_init (uint64_t init_flags)

Initialize ROCm SMI.

· rsmi status t rsmi shut down (void)

Shutdown ROCm SMI.

rsmi_status_t rsmi_num_monitor_devices (uint32_t *num_devices)

Get the number of devices that have monitor information.

rsmi status t rsmi dev id get (uint32 t dv ind, uint16 t *id)

Get the device id associated with the device with provided device index.

rsmi_status_t rsmi_dev_vendor_id_get (uint32_t dv_ind, uint16_t *id)

Get the device vendor id associated with the device with provided device index.

• rsmi_status_t rsmi_dev_name_get (uint32_t dv_ind, char *name, size_t len)

Get the name string of a gpu device.

• rsmi_status_t rsmi_dev_vendor_name_get (uint32_t id, char *name, size_t len)

Get the name string for a give vendor ID.

rsmi_status_t rsmi_dev_subsystem_id_get (uint32_t dv_ind, uint16_t *id)

Get the subsystem device id associated with the device with provided device index.

rsmi_status_t rsmi_dev_subsystem_name_get (uint32_t dv_ind, char *name, size_t len)

Get the name string for the device subsytem.

rsmi_status_t rsmi_dev_subsystem_vendor_id_get (uint32_t dv_ind, uint16_t *id)

Get the device subsystem vendor id associated with the device with provided device index.

• rsmi_status_t rsmi_dev_unique_id_get (uint32_t dv_ind, uint64_t *id)

Get Unique ID.

rsmi_status_t rsmi_dev_pci_bandwidth_get (uint32_t dv_ind, rsmi_pcie_bandwidth_t *bandwidth)

Get the list of possible PCIe bandwidths that are available.

rsmi_status_t rsmi_dev_pci_id_get (uint32_t dv_ind, uint64_t *bdfid)

Get the unique PCI device identifier associated for a device.

 rsmi status t rsmi dev pci throughput get (uint32 t dv ind, uint64 t *sent, uint64 t *received, uint64 ← t *max_pkt_sz)

Get PCIe traffic information.

rsmi status t rsmi dev pci replay counter get (uint32 t dv ind, uint64 t *counter)

Get PCIe replay counter.

rsmi_status_t rsmi_dev_pci_bandwidth_set (uint32_t dv_ind, uint64_t bw_bitmask)

Control the set of allowed PCIe bandwidths that can be used.

rsmi_status_t rsmi_dev_power_ave_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *power)

Get the average power consumption of the device with provided device index.

rsmi_status_t rsmi_dev_power_cap_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *cap)

Get the cap on power which, when reached, causes the system to take action to reduce power.

• rsmi_status_t rsmi_dev_power_cap_range_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *max, uint64 t *min)

Get the range of valid values for the power cap.

• rsmi_status_t rsmi_dev_power_cap_set (uint32_t dv_ind, uint32_t sensor_ind, uint64_t cap)

Set the power cap value.

 rsmi status t rsmi dev power profile set (uint32 t dv ind, uint32 t sensor ind, rsmi power profile ← preset_masks_t profile)

Set the power profile.

 rsmi_status_t rsmi_dev_memory_total_get (uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_← t *total)

Get the total amount of memory that exists.

 rsmi_status_t rsmi_dev_memory_usage_get (uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_← t *used)

Get the current memory usage.

• rsmi_status_t rsmi_dev_fan_rpms_get (uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed)

Get the fan speed in RPMs of the device with the specified device index and 0-based sensor index.

rsmi_status_t rsmi_dev_fan_speed_get (uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed)

Get the fan speed for the specified device as a value relative to RSMI_MAX_FAN_SPEED.

• rsmi_status_t rsmi_dev_fan_speed_max_get (uint32_t dv_ind, uint32_t sensor_ind, uint64_t *max_speed)

Get the max. fan speed of the device with provided device index.

rsmi_status_t rsmi_dev_temp_metric_get (uint32_t dv_ind, uint32_t sensor_type, rsmi_temperature_metric t metric, int64_t *temperature)

Get the temperature metric value for the specified metric, from the specified temperature sensor on the specified device.

rsmi_status_t rsmi_dev_fan_reset (uint32_t dv_ind, uint32_t sensor_ind)

Reset the fan to automatic driver control.

rsmi_status_t rsmi_dev_fan_speed_set (uint32_t dv_ind, uint32_t sensor_ind, uint64_t speed)

Set the fan speed for the specified device with the provided speed, in RPMs.

rsmi_status_t rsmi_dev_busy_percent_get (uint32_t dv_ind, uint32_t *busy_percent)

Get percentage of time device is busy doing any processing.

rsmi_status_t rsmi_dev_perf_level_get (uint32_t dv_ind, rsmi_dev_perf_level_t *perf)

Get the performance level of the device with provided device index.

rsmi_status_t rsmi_dev_overdrive_level_get (uint32_t dv_ind, uint32_t *od)

Get the overdrive percent associated with the device with provided device index.

- rsmi_status_t rsmi_dev_gpu_clk_freq_get (uint32_t dv_ind, rsmi_clk_type_t clk_type, rsmi_frequencies_t *f)
- Get the list of possible system clock speeds of device for a specified clock type.

 rsmi status t rsmi dev od volt info get (uint32 t dv ind, rsmi od volt freg data t *odv)

This function retrieves the voltage/frequency curve information.

rsmi_status_t rsmi_dev_od_volt_curve_regions_get (uint32_t dv_ind, uint32_t *num_regions, rsmi_freq_← volt region t *buffer)

This function will retrieve the current valid regions in the frequency/voltage space.

• rsmi_status_t rsmi_dev_power_profile_presets_get (uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_ profile status t *status)

Get the list of available preset power profiles and an indication of which profile is currently active.

rsmi_status_t rsmi_dev_perf_level_set (int32_t dv_ind, rsmi_dev_perf_level_t perf_lvl)

Set the PowerPlay performance level associated with the device with provided device index with the provided value.

• rsmi_status_t rsmi_dev_overdrive_level_set (int32_t dv_ind, uint32_t od)

Set the overdrive percent associated with the device with provided device index with the provided value. See details for WARNING.

- rsmi_status_t rsmi_dev_gpu_clk_freq_set (uint32_t dv_ind, rsmi_clk_type_t clk_type, uint64_t freq_bitmask)
- Control the set of allowed frequencies that can be used for the specified clock.

 rsmi_status_t rsmi_dev_od_freq_range_set (uint32_t dv_ind, rsmi_clk_type_t clk, rsmi_range_t *range)

Set the frequency limits for the specified clock.

rsmi_status_t rsmi_version_get (rsmi_version_t *version)

Get the build version information for the currently running build of RSMI.

• rsmi_status_t rsmi_version_str_get (rsmi_sw_component_t component, char *ver_str, uint32_t len)

Get the driver version string for the current system.

• rsmi_status_t rsmi_dev_vbios_version_get (uint32_t dv_ind, char *vbios, uint32_t len)

Get the VBIOS identifer string.

- rsmi_status_t rsmi_dev_ecc_count_get (uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_error_count_t *ec)

 Retrieve the error counts for a GPU block.
- rsmi status t rsmi dev ecc enabled get (uint32 t dv ind, uint64 t *enabled mask)

Retrieve the enabled ECC bit-mask.

rsmi_status_t rsmi_dev_ecc_status_get (uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_ras_err_state_
 t *state)

60 File Documentation

Retrieve the ECC status for a GPU block.

rsmi_status_t rsmi_status_string (rsmi_status_t status, const char **status_string)

Get a description of a provided RSMI error status.

rsmi_status_t rsmi_dev_counter_group_supported (uint32_t dv_ind, rsmi_event_group_t group)

Tell if an event group is supported by a given device.

 rsmi_status_t rsmi_dev_counter_create (uint32_t dv_ind, rsmi_event_type_t type, rsmi_event_handle_← t *evnt handle)

Create a performance counter object.

rsmi status t rsmi dev counter destroy (rsmi event handle t evnt handle)

Deallocate a performance counter object.

 rsmi_status_t rsmi_counter_control (rsmi_event_handle_t evt_handle, rsmi_counter_command_t cmd, void *cmd_args)

Issue performance counter control commands.

rsmi_status_t rsmi_counter_read (rsmi_event_handle_t evt_handle, rsmi_counter_value_t *value)

Read the current value of a performance counter.

rsmi_status_t rsmi_counter_available_counters_get (uint32_t dv_ind, rsmi_event_group_t grp, uint32_
 t *available)

Get the number of currently available counters.

7.1.1 Detailed Description

The rocm_smi library api is new, and therefore subject to change either at the ABI or API level. Instead of marking every function prototype as "unstable", we are instead saying the API is unstable (i.e., changes are possible) while the major version remains 0. This means that if the API/ABI changes, we will not increment the major version to 1. Once the ABI stabilizes, we will increment the major version to 1, and thereafter increment it on all ABI breaks.

Main header file for the ROCm SMI library. All required function, structure, enum, etc. definitions should be defined in this file.

7.1.2 Macro Definition Documentation

7.1.2.1 #define RSMI_MAX_FAN_SPEED 255

Maximum possible value for fan speed. Should be used as the denominator when determining fan speed percentage.

7.1.3 Typedef Documentation

7.1.3.1 typedef uintptr_t rsmi_event_handle_t

Handle to performance event counter.

Event counter types

7.1.4 Enumeration Type Documentation

7.1.4.1 enum rsmi_status_t

Error codes retured by rocm smi lib functions.

Enumerator

RSMI_STATUS_SUCCESS Operation was successful.

RSMI_STATUS_INVALID_ARGS Passed in arguments are not valid.

RSMI_STATUS_NOT_SUPPORTED The requested information or action is not available for the given input, on the given system

RSMI_STATUS_FILE_ERROR Problem accessing a file. This may because the operation is not supported by the Linux kernel version running on the executing machine

RSMI_STATUS_PERMISSION Permission denied/EACCESS file error. Many functions require root access to run.

RSMI_STATUS_OUT_OF_RESOURCES Unable to acquire memory or other resource

RSMI_STATUS_INTERNAL_EXCEPTION An internal exception was caught.

RSMI_STATUS_INPUT_OUT_OF_BOUNDS The provided input is out of allowable or safe range

RSMI STATUS INIT ERROR An error occurred when rsmi initializing internal data structures

RSMI_STATUS_NOT_YET_IMPLEMENTED The requested function has not yet been implemented in the current system for the current devices

RSMI_STATUS_NOT_FOUND An item was searched for but not found

RSMI_STATUS_INSUFFICIENT_SIZE Not enough resources were available for the operation

RSMI STATUS INTERRUPT An interrupt occurred during execution of function

RSMI_STATUS_UNEXPECTED_SIZE An unexpected amount of data was read

RSMI_STATUS_UNKNOWN_ERROR An unknown error occurred.

7.1.4.2 enum rsmi_init_flags_t

Initialization flags.

Initialization flags may be OR'd together and passed to rsmi init().

Enumerator

RSMI_INIT_FLAG_ALL_GPUS Attempt to add all GPUs found (including non-AMD) to the list of devices from which SMI information can be retrieved. By default, only AMD devices are ennumerated by RSMI.

7.1.4.3 enum rsmi_dev_perf_level_t

PowerPlay performance levels.

Enumerator

RSMI_DEV_PERF_LEVEL_AUTO Performance level is "auto".

RSMI_DEV_PERF_LEVEL_LOW Keep PowerPlay levels "low", regardless of workload

RSMI_DEV_PERF_LEVEL_HIGH Keep PowerPlay levels "high", regardless of workload

RSMI_DEV_PERF_LEVEL_MANUAL Only use values defined by manually setting the RSMI_CLK_TYP↔ E_SYS speed

RSMI_DEV_PERF_LEVEL_STABLE_STD Stable power state with profiling clocks

RSMI DEV_PERF_LEVEL_STABLE_PEAK Stable power state with peak clocks.

RSMI_DEV_PERF_LEVEL_STABLE_MIN_MCLK Stable power state with minimum memory clock

RSMI_DEV_PERF_LEVEL_STABLE_MIN_SCLK Stable power state with minimum system clock

RSMI_DEV_PERF_LEVEL_UNKNOWN Unknown performance level.

62 File Documentation

7.1.4.4 enum rsmi_sw_component_t

Available clock types.

Software components

Enumerator

RSMI_SW_COMP_DRIVER Driver.

7.1.4.5 enum rsmi_event_group_t

Enum denoting an event group. The value of the enum is the base value for all the event enums in the group.

Event Groups

Enumerator

RSMI EVNT_GRP_XGMI Data Fabric (XGMI) related events.

7.1.4.6 enum rsmi_event_type_t

Event type enum. Events belonging to a particular event group rsmi_event_group_t should begin ennumerating at the rsmi_event_group_t value for that group.

Event types

Enumerator

RSMI_EVNT_XGMI_0_NOP_TX NOPs sent to neighbor 0.

RSMI_EVNT_XGMI_0_REQUEST_TX Outgoing requests to neighbor 0

RSMI_EVNT_XGMI_0_RESPONSE_TX Outgoing responses to neighbor 0

RSMI_EVNT_XGMI_0_BEATS_TX Data beats sent to neighbor 0

 ${\it RSMI_EVNT_XGMI_1_NOP_TX}$ NOPs sent to neighbor 1.

RSMI_EVNT_XGMI_1_REQUEST_TX neighbor 1 Outgoing requests to

RSMI_EVNT_XGMI_1_RESPONSE_TX Outgoing responses to neighbor 1

RSMI_EVNT_XGMI_1_BEATS_TX Data beats sent to neighbor 1

7.1.4.7 enum rsmi_counter_command_t

Event counter commands

Enumerator

RSMI_CNTR_CMD_START Start the counter.

RSMI_CNTR_CMD_STOP Stop the counter.

7.1.4.8 enum rsmi_clk_type_t

Clock types

Enumerator

RSMI_CLK_TYPE_SYS System clock.

RSMI CLK TYPE DF Data Fabric clock (for ASICs running on a separate clock)

RSMI_CLK_TYPE_DCEF Display Controller Engine clock.

RSMI_CLK_TYPE_SOC SOC clock.

RSMI_CLK_TYPE_MEM Memory clock.

7.1.4.9 enum rsmi_temperature_metric_t

Temperature Metrics. This enum is used to identify various temperature metrics. Corresponding values will be in millidegress Celcius.

Enumerator

RSMI_TEMP_CURRENT Temperature current value.

RSMI_TEMP_MAX Temperature max value.

RSMI_TEMP_MIN Temperature min value.

RSMI_TEMP_MAX_HYST Temperature hysteresis value for max limit. (This is an absolute temperature, not a delta).

RSMI_TEMP_MIN_HYST Temperature hysteresis value for min limit. (This is an absolute temperature, not a delta).

RSMI_TEMP_CRITICAL Temperature critical max value, typically greater than corresponding temp_max values.

RSMI_TEMP_CRITICAL_HYST Temperature hysteresis value for critical limit. (This is an absolute temperature, not a delta).

RSMI_TEMP_EMERGENCY Temperature emergency max value, for chips supporting more than two upper temperature limits. Must be equal or greater than corresponding temp_crit values.

RSMI_TEMP_EMERGENCY_HYST Temperature hysteresis value for emergency limit. (This is an absolute temperature, not a delta).

RSMI_TEMP_CRIT_MIN Temperature critical min value, typically lower than corresponding temperature minimum values.

RSMI_TEMP_CRIT_MIN_HYST Temperature hysteresis value for critical minimum limit. (This is an absolute temperature, not a delta).

RSMI_TEMP_OFFSET Temperature offset which is added to the temperature reading by the chip.

RSMI_TEMP_LOWEST Historical minimum temperature.

RSMI_TEMP_HIGHEST Historical maximum temperature.

7.1.4.10 enum rsmi_temperature_type_t

This ennumeration is used to indicate from which part of the device a temperature reading should be obtained.

Enumerator

RSMI_TEMP_TYPE_EDGE Edge GPU temperature.

RSMI_TEMP_TYPE_JUNCTION Junction/hotspot temperature

RSMI_TEMP_TYPE_MEMORY VRAM temperature.

64 File Documentation

7.1.4.11 enum rsmi_power_profile_preset_masks_t

Pre-set Profile Selections. These bitmasks can be AND'd with the rsmi_power_profile_status_t.available_profiles returned from rsmi_dev_power_profile_presets_get() to determine which power profiles are supported by the system.

Enumerator

```
RSMI_PWR_PROF_PRST_CUSTOM_MASK Custom Power Profile.

RSMI_PWR_PROF_PRST_VIDEO_MASK Video Power Profile.

RSMI_PWR_PROF_PRST_POWER_SAVING_MASK Power Saving Profile.

RSMI_PWR_PROF_PRST_COMPUTE_MASK Compute Saving Profile.

RSMI_PWR_PROF_PRST_VR_MASK VR Power Profile. 3D Full Screen Power Profile

RSMI_PWR_PROF_PRST_BOOTUP_DEFAULT Default Boot Up Profile.

RSMI_PWR_PROF_PRST_LAST Invalid power profile.
```

7.1.4.12 enum rsmi gpu block t

This enum is used to identify different GPU blocks.

Enumerator

```
RSMI_GPU_BLOCK_INVALID Used to indicate an invalid block
RSMI_GPU_BLOCK_UMC UMC block.
RSMI_GPU_BLOCK_SDMA SDMA block.
RSMI_GPU_BLOCK_GFX GFX block.
RSMI_GPU_BLOCK_LAST The highest bit position for supported blocks
```

7.1.4.13 enum rsmi_ras_err_state_t

The current ECC state.

Enumerator

```
RSMI_RAS_ERR_STATE_NONE No current errors.

RSMI_RAS_ERR_STATE_DISABLED ECC is disabled.

RSMI_RAS_ERR_STATE_PARITY ECC errors present, but type unknown.

RSMI_RAS_ERR_STATE_SING_C Single correctable error.

RSMI_RAS_ERR_STATE_MULT_UC Multiple uncorrectable errors.

RSMI_RAS_ERR_STATE_POISON Firmware detected error and isolated page. Treat as uncorrectable.
```

7.1.4.14 enum rsmi_memory_type_t

Types of memory.

Enumerator

```
RSMI_MEM_TYPE_VRAM VRAM memory.

RSMI_MEM_TYPE_VIS_VRAM VRAM memory that is visible.

RSMI_MEM_TYPE_GTT GTT memory.
```

7.1.4.15 enum rsmi_freq_ind_t

This values of this enum are used as frequency identifiers.

Enumerator

RSMI_FREQ_IND_MIN Index used for the minimum frequency value. **RSMI_FREQ_IND_MAX** Index used for the maximum frequency value. **RSMI_FREQ_IND_INVALID** An invalid frequency index.

File Documentation

Index

available_profiles	num_profiles
rsmi_power_profile_status_t, 52	rsmi_power_profile_status_t, 52
	num_supported
Clock, Power and Performance Control, 36	rsmi_frequencies_t, 49
rsmi_dev_gpu_clk_freq_set, 37	_ ' ′
rsmi_dev_od_freq_range_set, 37	PCIe Control, 20
rsmi_dev_overdrive_level_set, 36	rsmi_dev_pci_bandwidth_set, 20
rsmi_dev_perf_level_set, 36	PCIe Queries, 18
Clock, Power and Performance Queries, 32	rsmi_dev_pci_bandwidth_get, 18
rsmi_dev_busy_percent_get, 32	rsmi_dev_pci_id_get, 18
rsmi_dev_gpu_clk_freq_get, 33	rsmi_dev_pci_replay_counter_get, 19
rsmi_dev_od_volt_curve_regions_get, 34	rsmi_dev_pci_throughput_get, 19
rsmi dev od volt info get, 33	Performance Counter Functions, 43
rsmi_dev_overdrive_level_get, 33	
rsmi_dev_perf_level_get, 32	rsmi_counter_available_counters_get, 45
	rsmi_counter_control, 44
rsmi_dev_power_profile_presets_get, 34	rsmi_counter_read, 45
curr_mclk_range	rsmi_dev_counter_create, 43
rsmi_od_volt_freq_data_t, 51	rsmi_dev_counter_destroy, 44
current	rsmi_dev_counter_group_supported, 43
rsmi_frequencies_t, 49	Physical State Control, 30
rsmi_power_profile_status_t, 52	rsmi_dev_fan_reset, 30
5 0 1 1	rsmi_dev_fan_speed_set, 30
Error Queries, 41	Physical State Queries, 27
rsmi_dev_ecc_count_get, 41	rsmi_dev_fan_rpms_get, 27
rsmi_dev_ecc_enabled_get, 41	rsmi_dev_fan_speed_get, 27
rsmi_dev_ecc_status_get, 42	rsmi_dev_fan_speed_max_get, 28
rsmi_status_string, 42	rsmi_dev_temp_metric_get, 28
	Power Control, 23
frequency	rsmi_dev_power_cap_set, 23
rsmi_frequencies_t, 49	rsmi_dev_power_profile_set, 23
11 18 0 1 10	Power Queries, 21
Identifier Queries, 13	rsmi_dev_power_ave_get, 21
rsmi_dev_id_get, 13	rsmi_dev_power_cap_get, 21
rsmi_dev_name_get, 14	rsmi_dev_power_cap_range_get, 22
rsmi_dev_subsystem_id_get, 15	rsmi_dev_power_cap_range_get, 22
rsmi_dev_subsystem_name_get, 15	RSMI CLK TYPE DCEF
rsmi_dev_subsystem_vendor_id_get, 16	rocm_smi.h, 63
rsmi_dev_unique_id_get, 16	RSMI CLK TYPE DF
rsmi_dev_vendor_id_get, 14	
rsmi_dev_vendor_name_get, 15	rocm_smi.h, 63
rsmi_num_monitor_devices, 13	RSMI_CLK_TYPE_MEM
Initialization and Shutdown, 11	rocm_smi.h, 63
rsmi_init, 11	RSMI_CLK_TYPE_SOC
rsmi_shut_down, 11	rocm_smi.h, 63
,	RSMI_CLK_TYPE_SYS
lanes	rocm_smi.h, 63
rsmi_pcie_bandwidth_t, 51	RSMI_CNTR_CMD_START
<u> </u>	rocm_smi.h, 62
Memory Queries, 25	RSMI_CNTR_CMD_STOP
rsmi_dev_memory_total_get, 25	rocm_smi.h, 62
rsmi_dev_memory_usage_get, 25	RSMI_DEV_PERF_LEVEL_AUTO

rocm smi.h, 61	rocm smi.h, 64
- · · ·	RSMI_MEM_TYPE_VRAM
RSMI_DEV_PERF_LEVEL_HIGH	
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_LOW	RSMI_PWR_PROF_PRST_BOOTUP_DEFAULT
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_MANUAL	RSMI_PWR_PROF_PRST_COMPUTE_MASK
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_STABLE_MIN_MCLK	RSMI_PWR_PROF_PRST_CUSTOM_MASK
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_STABLE_MIN_SCLK	RSMI PWR PROF PRST LAST
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_STABLE_PEAK	RSMI_PWR_PROF_PRST_POWER_SAVING_MASK
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_STABLE_STD	RSMI_PWR_PROF_PRST_VIDEO_MASK
rocm_smi.h, 61	rocm_smi.h, 64
RSMI_DEV_PERF_LEVEL_UNKNOWN	RSMI_PWR_PROF_PRST_VR_MASK
rocm_smi.h, 61	rocm_smi.h, 64
RSMI EVNT GRP XGMI	RSMI_RAS_ERR_STATE_DISABLED
rocm_smi.h, 62	rocm_smi.h, 64
RSMI_EVNT_XGMI_0_BEATS_TX	RSMI_RAS_ERR_STATE_MULT_UC
rocm_smi.h, 62	rocm_smi.h, 64
RSMI_EVNT_XGMI_0_NOP_TX	RSMI_RAS_ERR_STATE_NONE
rocm_smi.h, 62	rocm_smi.h, 64
RSMI_EVNT_XGMI_0_REQUEST_TX	RSMI_RAS_ERR_STATE_PARITY
rocm_smi.h, 62	rocm_smi.h, 64
RSMI_EVNT_XGMI_0_RESPONSE_TX	RSMI_RAS_ERR_STATE_POISON
rocm smi.h, 62	rocm_smi.h, 64
- · · ·	RSMI_RAS_ERR_STATE_SING_C
RSMI_EVNT_XGMI_1_BEATS_TX	
rocm_smi.h, 62	rocm_smi.h, 64
RSMI_EVNT_XGMI_1_NOP_TX	RSMI_STATUS_FILE_ERROR
rocm_smi.h, 62	rocm_smi.h, 61
RSMI_EVNT_XGMI_1_REQUEST_TX	RSMI_STATUS_INIT_ERROR
rocm_smi.h, 62	rocm_smi.h, 61
RSMI_EVNT_XGMI_1_RESPONSE_TX	RSMI_STATUS_INPUT_OUT_OF_BOUNDS
rocm_smi.h, 62	rocm smi.h, 61
RSMI_FREQ_IND_INVALID	RSMI STATUS INSUFFICIENT SIZE
rocm_smi.h, 65	rocm_smi.h, 61
RSMI_FREQ_IND_MAX	RSMI_STATUS_INTERNAL_EXCEPTION
rocm_smi.h, 65	rocm_smi.h, 61
RSMI_FREQ_IND_MIN	RSMI_STATUS_INTERRUPT
rocm_smi.h, 65	rocm_smi.h, 61
RSMI_GPU_BLOCK_GFX	RSMI_STATUS_INVALID_ARGS
rocm_smi.h, 64	rocm_smi.h, 61
RSMI_GPU_BLOCK_INVALID	RSMI_STATUS_NOT_FOUND
rocm_smi.h, 64	rocm smi.h, 61
RSMI_GPU_BLOCK_LAST	RSMI_STATUS_NOT_SUPPORTED
rocm_smi.h, 64	rocm_smi.h, 61
RSMI_GPU_BLOCK_SDMA	RSMI_STATUS_NOT_YET_IMPLEMENTED
rocm_smi.h, 64	rocm_smi.h, 61
RSMI_GPU_BLOCK_UMC	RSMI_STATUS_OUT_OF_RESOURCES
rocm_smi.h, 64	rocm_smi.h, 61
RSMI_INIT_FLAG_ALL_GPUS	RSMI_STATUS_PERMISSION
rocm_smi.h, 61	rocm_smi.h, 61
RSMI_MAX_FAN_SPEED	RSMI_STATUS_SUCCESS
rocm_smi.h, 60	rocm_smi.h, 61
RSMI_MEM_TYPE_GTT	RSMI_STATUS_UNEXPECTED_SIZE
rocm_smi.h, 64	rocm_smi.h, 61
RSMI_MEM_TYPE_VIS_VRAM	RSMI_STATUS_UNKNOWN_ERROR

rocm_smi.h, 61	RSMI_EVNT_XGMI_0_NOP_TX, 62
RSMI_SW_COMP_DRIVER	RSMI_EVNT_XGMI_0_REQUEST_TX, 62
rocm_smi.h, 62	RSMI_EVNT_XGMI_0_RESPONSE_TX, 62
RSMI_TEMP_CRIT_MIN_HYST	RSMI_EVNT_XGMI_1_BEATS_TX, 62
rocm_smi.h, 63	RSMI_EVNT_XGMI_1_NOP_TX, 62
RSMI_TEMP_CRIT_MIN	RSMI_EVNT_XGMI_1_REQUEST_TX, 62
rocm_smi.h, 63	RSMI_EVNT_XGMI_1_RESPONSE_TX, 62
RSMI_TEMP_CRITICAL_HYST	RSMI_FREQ_IND_INVALID, 65
rocm smi.h, 63	RSMI_FREQ_IND_MAX, 65
RSMI_TEMP_CRITICAL	RSMI FREQ IND MIN, 65
rocm_smi.h, 63	RSMI GPU BLOCK GFX, 64
RSMI_TEMP_CURRENT	RSMI_GPU_BLOCK_INVALID, 64
rocm_smi.h, 63	RSMI_GPU_BLOCK_LAST, 64
RSMI_TEMP_EMERGENCY_HYST	RSMI GPU BLOCK SDMA, 64
rocm_smi.h, 63	RSMI_GPU_BLOCK_UMC, 64
RSMI_TEMP_EMERGENCY	RSMI_INIT_FLAG_ALL_GPUS, 61
rocm smi.h, 63	RSMI MAX FAN SPEED, 60
RSMI TEMP HIGHEST	RSMI MEM TYPE GTT, 64
-	
rocm_smi.h, 63	RSMI_MEM_TYPE_VIS_VRAM, 64
RSMI_TEMP_LOWEST	RSMI_MEM_TYPE_VRAM, 64
rocm_smi.h, 63	RSMI_PWR_PROF_PRST_BOOTUP_DEFAULT,
RSMI_TEMP_MAX_HYST	64
rocm_smi.h, 63	RSMI_PWR_PROF_PRST_COMPUTE_MASK, 64
RSMI_TEMP_MAX	RSMI_PWR_PROF_PRST_CUSTOM_MASK, 64
rocm_smi.h, 63	RSMI_PWR_PROF_PRST_LAST, 64
RSMI_TEMP_MIN_HYST	RSMI_PWR_PROF_PRST_POWER_SAVING_←
rocm_smi.h, 63	MASK, 64
RSMI_TEMP_MIN	RSMI_PWR_PROF_PRST_VIDEO_MASK, 64
rocm_smi.h, 63	RSMI_PWR_PROF_PRST_VR_MASK, 64
RSMI_TEMP_OFFSET	RSMI_RAS_ERR_STATE_DISABLED, 64
rocm_smi.h, 63	RSMI_RAS_ERR_STATE_MULT_UC, 64
RSMI_TEMP_TYPE_EDGE	RSMI_RAS_ERR_STATE_NONE, 64
rocm_smi.h, 63	RSMI_RAS_ERR_STATE_PARITY, 64
RSMI_TEMP_TYPE_JUNCTION	RSMI_RAS_ERR_STATE_POISON, 64
rocm_smi.h, 63	RSMI RAS ERR STATE SING C, 64
RSMI_TEMP_TYPE_MEMORY	RSMI_STATUS_FILE_ERROR, 61
rocm_smi.h, 63	RSMI_STATUS_INIT_ERROR, 61
rocm_smi.h, 55	RSMI_STATUS_INPUT_OUT_OF_BOUNDS, 61
RSMI_CLK_TYPE_DCEF, 63	RSMI_STATUS_INSUFFICIENT_SIZE, 61
RSMI CLK TYPE DF, 63	RSMI_STATUS_INTERNAL_EXCEPTION, 61
RSMI_CLK_TYPE_MEM, 63	RSMI_STATUS_INTERRUPT, 61
RSMI CLK TYPE SOC, 63	RSMI STATUS INVALID ARGS, 61
RSMI_CLK_TYPE_SYS, 63	RSMI_STATUS_NOT_FOUND, 61
RSMI_CNTR_CMD_START, 62	RSMI_STATUS_NOT_SUPPORTED, 61
RSMI_CNTR_CMD_STOP, 62	RSMI_STATUS_NOT_YET_IMPLEMENTED, 61
RSMI_DEV_PERF_LEVEL_AUTO, 61	RSMI_STATUS_OUT_OF_RESOURCES, 61
RSMI_DEV_PERF_LEVEL_HIGH, 61	RSMI_STATUS_PERMISSION, 61
RSMI_DEV_PERF_LEVEL_LOW, 61	RSMI_STATUS_SUCCESS, 61
RSMI_DEV_PERF_LEVEL_MANUAL, 61	RSMI_STATUS_UNEXPECTED_SIZE, 61
RSMI_DEV_PERF_LEVEL_STABLE_MIN_MCLK,	RSMI_STATUS_UNKNOWN_ERROR, 61
61	RSMI_SW_COMP_DRIVER, 62
RSMI_DEV_PERF_LEVEL_STABLE_MIN_SCLK,	RSMI_TEMP_CRIT_MIN_HYST, 63
61	RSMI_TEMP_CRIT_MIN, 63
RSMI_DEV_PERF_LEVEL_STABLE_PEAK, 61	RSMI_TEMP_CRITICAL_HYST, 63
RSMI_DEV_PERF_LEVEL_STABLE_STD, 61	RSMI_TEMP_CRITICAL, 63
RSMI_DEV_PERF_LEVEL_UNKNOWN, 61	RSMI_TEMP_CURRENT, 63
RSMI_EVNT_GRP_XGMI, 62	RSMI_TEMP_EMERGENCY_HYST, 63
RSMI_EVNT_XGMI_0_BEATS_TX, 62	RSMI_TEMP_EMERGENCY, 63

RSMI TEMP HIGHEST, 63	Physical State Queries, 28
RSMI TEMP LOWEST, 63	rsmi_dev_fan_speed_set
RSMI TEMP MAX HYST, 63	Physical State Control, 30
RSMI TEMP MAX, 63	rsmi_dev_gpu_clk_freq_get
RSMI_TEMP_MIN_HYST, 63	Clock, Power and Performance Queries, 33
RSMI_TEMP_MIN, 63	rsmi dev gpu clk freq set
RSMI_TEMP_OFFSET, 63	Clock, Power and Performance Control, 37
RSMI_TEMP_TYPE_EDGE, 63	rsmi_dev_id_get
RSMI_TEMP_TYPE_JUNCTION, 63	Identifier Queries, 13
RSMI TEMP TYPE MEMORY, 63	rsmi_dev_memory_total_get
rsmi clk type t, 62	Memory Queries, 25
rsmi_counter_command_t, 62	rsmi_dev_memory_usage_get
rsmi_dev_perf_level_t, 61	Memory Queries, 25
rsmi_event_group_t, 62	rsmi_dev_name_get
rsmi_event_handle_t, 60	Identifier Queries, 14
rsmi_event_type_t, 62	rsmi_dev_od_freq_range_set
rsmi_freq_ind_t, 64	Clock, Power and Performance Control, 37
rsmi gpu block t, 64	rsmi_dev_od_volt_curve_regions_get
rsmi_init_flags_t, 61	Clock, Power and Performance Queries, 34
rsmi_memory_type_t, 64	rsmi_dev_od_volt_info_get
rsmi_power_profile_preset_masks_t, 63	Clock, Power and Performance Queries, 33
rsmi_ras_err_state_t, 64	rsmi_dev_overdrive_level_get
rsmi status t, 61	Clock, Power and Performance Queries, 33
rsmi sw component t, 61	rsmi_dev_overdrive_level_set
rsmi_temperature_metric_t, 63	Clock, Power and Performance Control, 36
rsmi_temperature_type_t, 63	rsmi_dev_pci_bandwidth_get
smi_clk_type_t	PCIe Queries, 18
rocm_smi.h, 62	rsmi_dev_pci_bandwidth_set
smi_counter_available_counters_get	PCIe Control, 20
Performance Counter Functions, 45	rsmi_dev_pci_id_get
smi_counter_command_t	PCIe Queries, 18
rocm_smi.h, 62	rsmi_dev_pci_replay_counter_get
smi_counter_control	PCIe Queries, 19
Performance Counter Functions, 44	waret alast mat themselvelens to mat
i dilailiana adamai i anonono, i i	rsmi dev pci throughput get
smi_counter_read	PCIe Queries, 19
	,,
smi_counter_read	PCIe Queries, 19
smi_counter_read Performance Counter Functions, 45	PCIe Queries, 19 rsmi_dev_perf_level_get
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get
smi_counter_read Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21 rsmi_dev_power_cap_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43	PCIe Queries, 19 rsmi_dev_perf_level_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21 rsmi_dev_power_cap_get Power Queries, 21 rsmi_dev_power_cap_range_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get	PCIe Queries, 19 rsmi_dev_perf_level_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get	PCIe Queries, 19 rsmi_dev_perf_level_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41	PCIe Queries, 19 rsmi_dev_perf_level_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41 smi_dev_ecc_status_get Error Queries, 42	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21 rsmi_dev_power_cap_get Power Queries, 21 rsmi_dev_power_cap_range_get Power Queries, 22 rsmi_dev_power_cap_set Power Control, 23 rsmi_dev_power_profile_presets_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41 smi_dev_ecc_status_get Error Queries, 42	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21 rsmi_dev_power_cap_get Power Queries, 21 rsmi_dev_power_cap_get Power Queries, 22 rsmi_dev_power_cap_range_get Power Queries, 22 rsmi_dev_power_cap_set Power Control, 23 rsmi_dev_power_profile_presets_get Clock, Power and Performance Queries, 34
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41 smi_dev_ecc_status_get Error Queries, 42 smi_dev_fan_reset	PCIe Queries, 19 rsmi_dev_perf_level_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41 smi_dev_ecc_status_get Error Queries, 42 smi_dev_fan_reset Physical State Control, 30	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21 rsmi_dev_power_cap_get Power Queries, 21 rsmi_dev_power_cap_range_get Power Queries, 22 rsmi_dev_power_cap_set Power Control, 23 rsmi_dev_power_profile_presets_get Clock, Power and Performance Queries, 34 rsmi_dev_power_profile_set Power Control, 23
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41 smi_dev_ecc_status_get Error Queries, 42 smi_dev_fan_reset Physical State Control, 30 smi_dev_fan_rpms_get	PCIe Queries, 19 rsmi_dev_perf_level_get
Performance Counter Functions, 45 smi_counter_value_t, 47 smi_dev_busy_percent_get Clock, Power and Performance Queries, 32 smi_dev_counter_create Performance Counter Functions, 43 smi_dev_counter_destroy Performance Counter Functions, 44 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_counter_group_supported Performance Counter Functions, 43 smi_dev_ecc_count_get Error Queries, 41 smi_dev_ecc_enabled_get Error Queries, 41 smi_dev_ecc_status_get Error Queries, 42 smi_dev_fan_reset Physical State Control, 30 smi_dev_fan_rpms_get Physical State Queries, 27	PCIe Queries, 19 rsmi_dev_perf_level_get Clock, Power and Performance Queries, 32 rsmi_dev_perf_level_set Clock, Power and Performance Control, 36 rsmi_dev_perf_level_t rocm_smi.h, 61 rsmi_dev_power_ave_get Power Queries, 21 rsmi_dev_power_cap_get Power Queries, 21 rsmi_dev_power_cap_range_get Power Queries, 22 rsmi_dev_power_cap_set Power Control, 23 rsmi_dev_power_profile_presets_get Clock, Power and Performance Queries, 34 rsmi_dev_power_profile_set Power Control, 23 rsmi_dev_power_profile_set Power Control, 23 rsmi_dev_subsystem_id_get Identifier Queries, 15

Identifier Queries, 16	rsmi_sw_component_t
rsmi dev temp metric get	rocm_smi.h, 61
Physical State Queries, 28	rsmi_temperature_metric_t
rsmi_dev_unique_id_get	rocm_smi.h, 63
Identifier Queries, 16	rsmi temperature type t
rsmi dev vbios version get	rocm_smi.h, 63
Version Queries, 40	rsmi_version_get
rsmi_dev_vendor_id_get	Version Queries, 39
Identifier Queries, 14	rsmi_version_str_get
rsmi dev vendor name get	Version Queries, 39
Identifier Queries, 15	rsmi version t, 53
rsmi_error_count_t, 47	70ro, 00
	transfer rate
rsmi_event_group_t	rsmi_pcie_bandwidth_t, 51
rocm_smi.h, 62	<u> </u>
rsmi_event_handle_t	vc_points
rocm_smi.h, 60	rsmi_od_volt_curve_t, 50
rsmi_event_type_t	Version Queries, 39
rocm_smi.h, 62	rsmi_dev_vbios_version_get, 40
rsmi_freq_ind_t	rsmi_version_get, 39
rocm_smi.h, 64	rsmi_version_str_get, 39
rsmi_freq_volt_region_t, 48	
rsmi_frequencies_t, 48	
current, 49	
frequency, 49	
num_supported, 49	
rsmi_gpu_block_t	
rocm_smi.h, 64	
rsmi_init	
Initialization and Shutdown, 11	
rsmi_init_flags_t	
rocm_smi.h, 61	
rsmi_memory_type_t	
rocm_smi.h, 64	
rsmi_num_monitor_devices	
Identifier Queries, 13	
rsmi_od_vddc_point_t, 49	
rsmi_od_volt_curve_t, 50	
vc_points, 50	
rsmi_od_volt_freq_data_t, 50	
curr_mclk_range, 51	
rsmi_pcie_bandwidth_t, 51	
lanes, 51	
transfer_rate, 51	
rsmi_power_profile_preset_masks_t	
rocm_smi.h, 63	
rsmi_power_profile_status_t, 52	
available profiles, 52	
current, 52	
num_profiles, 52	
rsmi_range_t, 52	
rsmi_ras_err_state_t	
rocm_smi.h, 64	
rsmi_shut_down	
Initialization and Shutdown, 11	
rsmi_status_string Error Quaries 42	
Error Queries, 42	
rsmi_status_t	
rocm_smi.h, 61	