### 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	a 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	ier Queries	3	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)	13
			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 tid. 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)	15
		5.2.2.8	rsmi_dev_subsystem_vendor_id_get(uint32_t dv_ind, uint16_t *id)	16
5.3	PCIe C	Queries .		17
	5.3.1	Detailed	Description	17
	5.3.2	Function	Documentation	17
		5.3.2.1	rsmi_dev_pci_bandwidth_get(uint32_t dv_ind, rsmi_pcie_bandwidth_t *bandwidth)	17
		5.3.2.2	rsmi_dev_pci_id_get(uint32_t dv_ind, uint64_t *bdfid)	17
		5.3.2.3	rsmi_dev_pci_throughput_get(uint32_t dv_ind, uint64_t *sent, uint64_← t *received, uint64_t *max_pkt_sz)	18
5.4	PCIe C	Control		19
	5.4.1	Detailed	Description	19
	5.4.2	Function	Documentation	19
		5.4.2.1	rsmi_dev_pci_bandwidth_set(uint32_t dv_ind, uint64_t bw_bitmask)	19
5.5	Power	Queries .		20
	5.5.1	Detailed	Description	20
	5.5.2	Function	Documentation	20
		5.5.2.1	rsmi_dev_power_ave_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_t *power)	20
		5.5.2.2	rsmi_dev_power_cap_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_t *cap) .	20
		5.5.2.3	rsmi_dev_power_cap_range_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_← t *max, uint64_t *min)	21
5.6	Power	Control .		22
	5.6.1	Detailed	Description	22
	5.6.2	Function	Documentation	22
		5.6.2.1	rsmi_dev_power_cap_set(uint32_t dv_ind, uint32_t sensor_ind, uint64_t cap)	22
		5.6.2.2	rsmi_dev_power_profile_set(uint32_t dv_ind, uint32_t sensor_ind, rsmi_power ← profile_preset_masks_t profile)	22
5.7	Memor	y Queries		24
	5.7.1	Detailed	Description	24
	5.7.2	Function	Documentation	24
		5.7.2.1	rsmi_dev_memory_total_get(uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_t *total)	24

CONTENTS

		5.7.2.2	rsmi_dev_memory_usage_get(uint32_t dv_ind, rsmi_memory_type_t mem_type, uint64_t *used)	24
5.8	Physica	al State Qu	ueries	26
	5.8.1	Detailed	Description	26
	5.8.2	Function	Documentation	26
		5.8.2.1	rsmi_dev_fan_rpms_get(uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed) .	26
		5.8.2.2	rsmi_dev_fan_speed_get(uint32_t dv_ind, uint32_t sensor_ind, int64_t *speed) .	26
		5.8.2.3	rsmi_dev_fan_speed_max_get(uint32_t dv_ind, uint32_t sensor_ind, uint64_ t *max_speed)	27
		5.8.2.4	rsmi_dev_temp_metric_get(uint32_t dv_ind, uint32_t sensor_ind, rsmi_← temperature_metric_t metric, int64_t *temperature)	27
5.9	Physica	al State Co	ontrol	29
	5.9.1	Detailed	Description	29
	5.9.2	Function	Documentation	29
		5.9.2.1	rsmi_dev_fan_reset(uint32_t dv_ind, uint32_t sensor_ind)	29
		5.9.2.2	rsmi_dev_fan_speed_set(uint32_t dv_ind, uint32_t sensor_ind, uint64_t speed) .	29
5.10	Clock,	Power and	Performance Queries	31
	5.10.1	Detailed	Description	31
	5.10.2	Function	Documentation	31
		5.10.2.1	rsmi_dev_busy_percent_get(uint32_t dv_ind, uint32_t *busy_percent)	31
		5.10.2.2	rsmi_dev_perf_level_get(uint32_t dv_ind, rsmi_dev_perf_level_t *perf)	31
		5.10.2.3	rsmi_dev_overdrive_level_get(uint32_t dv_ind, uint32_t *od)	32
		5.10.2.4	$ \begin{array}{llllllllllllllllllllllllllllllllllll$	32
		5.10.2.5	rsmi_dev_od_volt_info_get(uint32_t dv_ind, rsmi_od_volt_freq_data_t *odv)	33
		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)	33
		5.10.2.7	rsmi_dev_power_profile_presets_get(uint32_t dv_ind, uint32_t sensor_ind, rsmi_power_profile_status_t *status)	33
5.11	Clock,	Power and	Performance Control	35
	5.11.1	Detailed	Description	35
	5.11.2	Function	Documentation	35
		5.11.2.1	rsmi_dev_perf_level_set(int32_t dv_ind, rsmi_dev_perf_level_t perf_lvl)	35

vi

			5.11.2.2	rsmi_dev_overdrive_level_set(int32_t dv_ind, uint32_t od)	35
			5.11.2.3	rsmi_dev_gpu_clk_freq_set(uint32_t dv_ind, rsmi_clk_type_t clk_type, uint64_ ← t freq_bitmask)	36
			5.11.2.4	rsmi_dev_od_freq_range_set(uint32_t dv_ind, rsmi_clk_type_t clk, rsmi_range ↔ _t *range)	36
	5.12	Version	Queries		38
		5.12.1	Detailed [	Description	38
		5.12.2	Function	Documentation	38
			5.12.2.1	rsmi_version_get(rsmi_version_t *version)	38
			5.12.2.2	rsmi_dev_vbios_version_get(uint32_t dv_ind, char *vbios, uint32_t len)	38
	5.13	Error C	ueries .		40
		5.13.1	Detailed [	Description	40
		5.13.2	Function	Documentation	40
			5.13.2.1	rsmi_dev_error_count_get(uint32_t dv_ind, rsmi_gpu_block_t block, rsmi_error ← _count_t *ec)	40
			5.13.2.2	rsmi_status_string(rsmi_status_t status, const char **status_string)	40
6	Data	Structu	ire Docum	entation	43
6	Data 6.1			_t Struct Reference	<b>43</b>
6			rror_count_		
6		rsmi_e 6.1.1	rror_count_ Detailed [	_t Struct Reference	43
6	6.1	rsmi_e 6.1.1	rror_count_ Detailed [ eq_volt_re	_t Struct Reference	43 43
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1	rror_count_ Detailed [ eq_volt_repaired [	_t Struct Reference	43 43 43
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1	Detailed [ eq_volt_re  Detailed [ equencies_	_t Struct Reference	43 43 43 44
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr	Detailed [ Detailed [ Detailed [ Detailed I Detailed [ Detailed I	_t Struct Reference	43 43 43 44 44
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1	Detailed [ Detailed [ Detailed [ Detailed I Detailed [ Detailed I	_t Struct Reference	43 43 44 44 44
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1	Detailed [ eq_volt_re  Detailed [ equencies]  Detailed [ Field Doc	_t Struct Reference	43 43 44 44 44
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1	Detailed I eq_volt_red Detailed I equencies_ Detailed I Field Docc 6.3.2.1	_t Struct Reference	43 43 44 44 44 44
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1 6.3.2	Detailed I eq_volt_re Detailed I equencies Detailed I Field Doc 6.3.2.1 6.3.2.2 6.3.2.3	_t Struct Reference	43 43 44 44 44 44 44
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1 6.3.2	Detailed I eq_volt_re Detailed I equencies Detailed I Field Doc 6.3.2.1 6.3.2.2 6.3.2.3 d_vddc_po	_t Struct Reference	43 43 44 44 44 44 45
6	6.1	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1 6.3.2	Detailed I eq_volt_red Detailed I equencies_ Detailed I Field Docc 6.3.2.1 6.3.2.2 6.3.2.3 d_vddc_pocc Detailed I	_t Struct Reference	43 43 44 44 44 44 45 45
6	<ul><li>6.1</li><li>6.2</li><li>6.3</li><li>6.4</li></ul>	rsmi_e 6.1.1 rsmi_fr 6.2.1 rsmi_fr 6.3.1 6.3.2	Detailed I eq_volt_red Detailed I equencies Detailed I Field Doc 6.3.2.1 6.3.2.2 6.3.2.3 d_vddc_po Detailed I d_volt_curv	_t Struct Reference Description  gion_t Struct Reference Description _t Struct Reference Description  umentation  num_supported  current  frequency  int_t Struct Reference  Description	43 43 44 44 44 44 45 45

CONTENTS vii

		6.5.2	Field Documentati	on		 	 	 46
			6.5.2.1 vc_poin	ts		 	 	 46
	6.6	rsmi_o	d_volt_freq_data_t	Struct Reference		 	 	 46
		6.6.1	Detailed Description	on		 	 	 46
		6.6.2	Field Documentati	on		 	 	 46
			6.6.2.1 curr_mc	lk_range		 	 	 46
	6.7	rsmi_p	cie_bandwidth_t St	ruct Reference		 	 	 47
		6.7.1	Detailed Description	on		 	 	 47
		6.7.2	Field Documentati	on		 	 	 47
			6.7.2.1 transfer	_rate		 	 	 47
			6.7.2.2 lanes .			 	 	 47
	6.8	rsmi_p	ower_profile_status	_t Struct Reference .		 	 	 47
		6.8.1	Detailed Description	on		 	 	 48
		6.8.2	Field Documentati	on		 	 	 48
			6.8.2.1 available	e_profiles		 	 	 48
			6.8.2.2 current			 	 	 48
			6.8.2.3 num_pr	ofiles		 	 	 48
	6.9	rsmi_ra	nge_t Struct Refer	ence		 	 	 48
		6.9.1	Detailed Description	on		 	 	 48
	6.10	rsmi_ve	ersion_t Struct Refe	rence		 	 	 49
		6.10.1	Detailed Description	on		 	 	 49
7	File I	Docume	ntation					51
	7.1	rocm s	mi.h File Reference			 	 	 51
		7.1.1		on				55
		7.1.2	•	ocumentation				55
			7.1.2.1 RSMI N	MAX_FAN_SPEED		 	 	 55
		7.1.3		Documentation				55
			7.1.3.1 rsmi_sta	atus_t		 	 	 55
			7.1.3.2 rsmi_de	v_perf_level_t		 	 	 56
			7.1.3.3 rsmi_clk	typet		 	 	 56
			7.1.3.4 rsmi_tei	mperature_metric_t .		 	 	 56
			7.1.3.5 rsmi_po	wer_profile_preset_m	asks_t .	 	 	 57
			_	emory_type_t				57
				q_ind_t				57
	_							
Inc	lex							59

### **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:

itialization and Shutdown	-11
entifier Queries	13
Cle Queries	17
Cle Control	19
ower Queries	20
ower Control	22
emory Queries	24
hysical State Queries	26
hysical State Control	29
lock, Power and Performance Queries	31
lock, Power and Performance Control	35
ersion Queries	38
rror Queries	40

6 Module Index

# **Chapter 3**

# **Data Structure Index**

#### 3.1 Data Structures

Here are the data structures with brief descriptions:

rsmi_error_count_t	
This structure holds error counts	43
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	43
rsmi_frequencies_t	
This structure holds information about clock frequencies	44
rsmi_od_vddc_point_t	
This structure represents a point on the frequency-voltage plane	45
rsmi_od_volt_curve_t	45
rsmi_od_volt_freq_data_t	
This structure holds the frequency-voltage values for a device	46
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	47
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	47
rsmi_range_t	
This structure represents a range (e.g., frequencies or voltages)	48
rsmi_version_t	
This structure holds version information	49

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

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 . . . . . . . . . .

51

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. Not currently used.
----	------------	---

#### 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.

#### 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.

#### **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 Identifier Queries 15

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.

#### **Parameters**

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

#### **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.
---

5.3 PCIe Queries 17

#### 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.

#### 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.

 $5.3.2.2 \quad rsmi\_status\_t \ rsmi\_dev\_pci\_id\_get \ ( \ uint32\_t \ \textit{dv\_ind,} \ uint64\_t * \textit{bdfid} \ )$ 

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.

#### **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  $dv_i$ nd, uint64\_t \* sent, uint64\_t \* received, uint64\_t \* received, uint64\_t \* received, uint64\_t \*

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.
---------------------	-----------------------------------

5.4 PCIe Control 19

#### 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 PCIe 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	A bitmask indicating the indices of the bandwidths that are to be enabled (1) and disabled
		(0). Only the lowest rsmi_frequencies_t::num_supported (of rsmi_pcie_bandwidth_t) bits of this mask are relevant.

#### 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 to the uint64\_t in microwatts pointed to by power. This function requires root privilege.

#### **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

MI_STATUS_SUCCESS	is returned upon successful call.
-------------------	-----------------------------------

```
\textbf{5.5.2.2} \quad \textbf{rsmi\_status\_t rsmi\_dev\_power\_cap\_get ( uint32\_t \textit{dv\_ind, uint32\_t sensor\_ind, uint64\_t} * \textit{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.	
in,out	сар	a pointer to a uint64_t that indicates the power cap, in microwatts  Generated by Doxygen	

5.5 Power Queries 21

#### 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\_ind, uint32\_t sensor\_ind, 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 i	returned upon successful call.
----------------------------	--------------------------------

#### 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.
in	profile	a rsmi_power_profile_preset_masks_t that hold the mask of the desired new-power profilegen

5.6 Power Control 23

#### Return values

RSMI\_STATUS\_SUCCESS is returned upon successful call.

#### 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
Ī	in,out	used	a pointer to uint64_t to which the amount of memory currently being used will be written

5.7 Memory Queries 25

#### 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 in RPMs.
- 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\_ind, 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 in RPMs.

Given a device index dv ind this function will get the fan speed.

#### **Parameters**

in   dv ind   a device index
------------------------------

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 255) to the uint32\_t 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\_ind, 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 0-based sensor index sensor\_ind, 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 to the memory location temperature.

#### **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	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.
		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,out	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

#### 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_regions, 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

RSML STATUS SUCCESS	is returned upon successful call.
110WI_0171100_0000L00	is retarried aport successial 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

#### **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\_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_t rsmi_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
---------------------	----------------------------------

5.12.2.2 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 <code>dv\_ind</code>, and a pointer to a char buffer, <code>vbios</code>, this function will write the VBIOS string (up to <code>len</code> characters) for device <code>dv\_ind</code> to <code>vbios</code>. The caller must ensure that it is safe to write at least <code>len</code> characters to <code>vbios</code>.

#### **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 vbios which can safely be written to by this function.

5.12 Version Queries 39

## Return values

RSMI\_STATUS\_SUCCESS is returned upon successful call.

#### 5.13 Error Queries

#### **Functions**

• rsmi\_status\_t rsmi\_dev\_error\_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\_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\_error\_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\_t ec, 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.

5.13.2.2 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

5.13 Error Queries 41

## Return values

RSMI\_STATUS\_SUCCESS | is returned upon successful call

## **Chapter 6**

## **Data Structure Documentation**

## 6.1 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.

## 6.1.1 Detailed Description

This structure holds error counts.

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

• rocm\_smi.h

## 6.2 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 <rocm_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.

## 6.2.1 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

## 6.3 rsmi\_frequencies\_t Struct Reference

This structure holds information about clock frequencies.

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

#### **Data Fields**

- uint32\_t num\_supported
- uint32\_t current
- uint64\_t frequency [RSMI\_MAX\_NUM\_FREQUENCIES]

## 6.3.1 Detailed Description

This structure holds information about clock frequencies.

#### 6.3.2 Field Documentation

6.3.2.1 uint32\_t rsmi\_frequencies\_t::num\_supported

The number of supported frequencies

6.3.2.2 uint32\_t rsmi\_frequencies\_t::current

The current frequency index

6.3.2.3 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

## 6.4 rsmi\_od\_vddc\_point\_t Struct Reference

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

```
#include <room smi.h>
```

#### **Data Fields**

· uint64\_t frequency

Frequency coordinate (in Hz)

• uint64\_t voltage

Voltage coordinate (in mV)

#### 6.4.1 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

## 6.5 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]

#### 6.5.1 Detailed Description

RSMI\_NUM\_VOLTAGE\_CURVE\_POINTS number of rsmi\_od\_vddc\_point\_t's

#### 6.5.2 Field Documentation

6.5.2.1 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

## 6.6 rsmi\_od\_volt\_freq\_data\_t Struct Reference

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

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

### **Data Fields**

· rsmi\_range\_t curr\_sclk\_range

The current SCLK frequency range.

rsmi\_range\_t curr\_mclk\_range

(upper bound only)

• 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.

#### 6.6.1 Detailed Description

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

#### 6.6.2 Field Documentation

6.6.2.1 rsmi\_range\_t rsmi\_od\_volt\_freq\_data\_t::curr\_mclk\_range

(upper bound only)

The current MCLK frequency range;

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

• rocm\_smi.h

## 6.7 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 <room smi.h>
```

#### **Data Fields**

- · rsmi\_frequencies\_t transfer\_rate
- uint32\_t lanes [RSMI\_MAX\_NUM\_FREQUENCIES]

#### 6.7.1 Detailed Description

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

#### 6.7.2 Field Documentation

6.7.2.1 rsmi\_frequencies\_t rsmi\_pcie\_bandwidth\_t::transfer\_rate

Transfer rates (T/s) that are possible

6.7.2.2 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

## 6.8 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

## 6.8.1 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.

#### 6.8.2 Field Documentation

6.8.2.1 rsmi\_bit\_field\_t rsmi\_power\_profile\_status\_t::available\_profiles

Which profiles are supported by this system

6.8.2.2 rsmi\_power\_profile\_preset\_masks\_t rsmi\_power\_profile\_status\_t::current

Which power profile is currently active

6.8.2.3 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

## 6.9 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.

## 6.9.1 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

## 6.10 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.

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

## **Data Structures**

· 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 PCIe 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.

52 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 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\_ $\leftarrow$  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\_UNKNOWN\_ERROR = 0xFFFFFFFF }

Error codes retured by rocm smi lib functions.

enum rsmi\_dev\_perf\_level\_t {

RSMI\_DEV\_PERF\_LEVEL\_AUTO = 0, RSMI\_DEV\_PERF\_LEVEL\_FIRST = RSMI\_DEV\_PERF\_LEVEL\_←
AUTO, RSMI DEV PERF LEVEL LOW, RSMI DEV PERF LEVEL HIGH,

RSMI\_DEV\_PERF\_LEVEL\_MANUAL, RSMI\_DEV\_PERF\_LEVEL\_STABLE\_STD, RSMI\_DEV\_PERF\_LE↔ 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\_clk\_type\_t { RSMI\_CLK\_TYPE\_SYS = 0x0, RSMI\_CLK\_TYPE\_FIRST = RSMI\_CLK\_TYPE\_S ← YS, RSMI\_CLK\_TYPE\_MEM, RSMI\_CLK\_TYPE\_LAST = RSMI\_CLK\_TYPE\_MEM }

Available clock types.

enum rsmi\_temperature\_metric\_t {

RSMI\_TEMP\_CURRENT = 0x0, RSMI\_TEMP\_FIRST = RSMI\_TEMP\_CURRENT, RSMI\_TEMP\_MAX, R↔ SMI\_TEMP\_MIN,

$$\label{eq:rsml_temp_offset} \begin{split} & \text{RSMI\_TEMP\_LOWEST, RSMI\_TEMP\_HIGHEST, } & \text{RSMI\_TEMP\_LAST} = \text{RSMI\_} \\ & \text{TEMP\_HIGHEST} \end{split} \end{split}$$

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

enum rsmi power profile preset masks t {

 $\label{eq:rsm_pwr_prof_prst_custom_mask} $$ = 0x1, RSMI\_PWR\_PROF\_PRST\_VIDEO\_MASK = 0x2, R \hookrightarrow 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,

**RSMI\_PWR\_PROF\_PRST\_INVALID** = 0xFFFFFFFFFFFFFF }

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 {

RSMI\_GPU\_BLOCK\_FIRST = 0, RSMI\_GPU\_BLOCK\_UMC = RSMI\_GPU\_BLOCK\_FIRST, RSMI\_GPU ← BLOCK\_SDMA, RSMI\_GPU\_BLOCK\_GFX,
RSMI\_GPU\_BLOCK\_LAST = RSMI\_GPU\_BLOCK\_GFX }

This enum is used to identify different GPU blocks.

• 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.

#### **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 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 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.

54 File Documentation

• 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 in RPMs.

- 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\_ind, 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\_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\_confile\_presets\_get (uint32\_t dv\_ind, uint32\_t sensor\_ind, uint32\_t s

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\_dev\_vbios\_version\_get (uint32\_t dv\_ind, char \*vbios, uint32\_t len)
   Get the VBIOS identifer string.
- rsmi\_status\_t rsmi\_dev\_error\_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\_status\_string (rsmi\_status\_t status, const char \*\*status\_string)

  Get a description of a provided RSMI error status.

#### 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 Enumeration Type Documentation

7.1.3.1 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

**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

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 for the operation

RSMI\_STATUS\_UNKNOWN\_ERROR An unknown error occurred.

56 File Documentation

7.1.3.2 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.

#### 7.1.3.3 enum rsmi\_clk\_type\_t

Available clock types.

#### Enumerator

**RSMI\_CLK\_TYPE\_SYS** System clock. **RSMI\_CLK\_TYPE\_MEM** Memory clock.

#### 7.1.3.4 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.

**RSMI\_TEMP\_MIN\_HYST** Temperature hysteresis value for min limit.

**RSMI\_TEMP\_CRITICAL** Temperature critical max value, typically greater than corresponding temp\_max values.

**RSMI\_TEMP\_CRITICAL\_HYST** Temperature hysteresis value for critical limit.

**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.

**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.

**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.

7.1.3.5 enum rsmi\_power\_profile\_preset\_masks\_t

Pre-set Profile Selections. These bitmasks can be AND'd with the <a href="mailto:rsmi\_power\_profile\_status\_t.available\_profiles">rsmi\_power\_profile\_status\_t.available\_profiles</a> returned from <a href="mailto:rsmi\_dev\_power\_profile\_presets\_get">rsmi\_dev\_power\_profile\_presets\_get</a>() 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.

7.1.3.6 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.
```

## 7.1.3.7 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.
```

58 File Documentation

# Index

available_profiles	num_supported
rsmi_power_profile_status_t, 48	rsmi_frequencies_t, 44
Clock, Power and Performance Control, 35	PCIe Control, 19
rsmi_dev_gpu_clk_freq_set, 36	rsmi_dev_pci_bandwidth_set, 19
rsmi_dev_od_freq_range_set, 36	PCIe Queries, 17
rsmi_dev_overdrive_level_set, 35	rsmi_dev_pci_bandwidth_get, 17
rsmi_dev_perf_level_set, 35	rsmi_dev_pci_id_get, 17
Clock, Power and Performance Queries, 31	rsmi_dev_pci_throughput_get, 18
rsmi_dev_busy_percent_get, 31	Physical State Control, 29
rsmi dev gpu clk freg get, 32	rsmi dev fan reset, 29
rsmi_dev_od_volt_curve_regions_get, 33	rsmi_dev_fan_speed_set, 29
rsmi_dev_od_volt_info_get, 32	Physical State Queries, 26
rsmi_dev_overdrive_level_get, 32	rsmi_dev_fan_rpms_get, 26
rsmi_dev_perf_level_get, 31	rsmi_dev_fan_speed_get, 26
rsmi_dev_power_profile_presets_get, 33	rsmi_dev_fan_speed_max_get, 27
curr_mclk_range	rsmi_dev_temp_metric_get, 27
rsmi_od_volt_freq_data_t, 46	Power Control, 22
current	
	rsmi_dev_power_cap_set, 22
rsmi_frequencies_t, 44	rsmi_dev_power_profile_set, 22
rsmi_power_profile_status_t, 48	Power Queries, 20
Error Queries, 40	rsmi_dev_power_ave_get, 20
rsmi dev error count get, 40	rsmi_dev_power_cap_get, 20
rsmi_status_string, 40	rsmi_dev_power_cap_range_get, 21
rsm_status_string, 40	RSMI_CLK_TYPE_MEM
frequency	rocm_smi.h, 56
rsmi_frequencies_t, 44	RSMI CLK TYPE SYS
	rocm_smi.h, 56
Identifier Queries, 13	RSMI_DEV_PERF_LEVEL_AUTO
rsmi_dev_id_get, 13	rocm_smi.h, 56
rsmi_dev_name_get, 14	RSMI_DEV_PERF_LEVEL_HIGH
rsmi_dev_subsystem_id_get, 15	rocm_smi.h, 56
rsmi_dev_subsystem_name_get, 15	RSMI DEV PERF LEVEL LOW
rsmi_dev_subsystem_vendor_id_get, 16	
rsmi_dev_vendor_id_get, 14	rocm_smi.h, 56
rsmi_dev_vendor_name_get, 15	RSMI_DEV_PERF_LEVEL_MANUAL
rsmi num monitor devices, 13	rocm_smi.h, 56
Initialization and Shutdown, 11	RSMI_DEV_PERF_LEVEL_STABLE_MIN_MCL
rsmi init, 11	rocm_smi.h, 56
rsmi_shut_down, 11	RSMI_DEV_PERF_LEVEL_STABLE_MIN_SCLK
	rocm_smi.h, 56
lanes	RSMI_DEV_PERF_LEVEL_STABLE_PEAK
rsmi pcie bandwidth t, 47	rocm_smi.h, 56
<b>–</b> – – – ,	RSMI_DEV_PERF_LEVEL_STABLE_STD
Memory Queries, 24	rocm_smi.h, 56
rsmi_dev_memory_total_get, 24	RSMI_DEV_PERF_LEVEL_UNKNOWN
rsmi_dev_memory_usage_get, 24	rocm_smi.h, 56
;_ 0 _0 ,	RSMI_FREQ_IND_INVALID
num_profiles	rocm_smi.h, 57
rsmi_power_profile_status_t, 48	RSMI_FREQ_IND_MAX

60 INDEX

room amily 57	room amily EG
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_FREQ_IND_MIN	RSMI_TEMP_CURRENT
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_MAX_FAN_SPEED	RSMI_TEMP_EMERGENCY_HYST
rocm_smi.h, 55	rocm_smi.h, 56
RSMI_MEM_TYPE_GTT	RSMI_TEMP_EMERGENCY
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_MEM_TYPE_VIS_VRAM	RSMI_TEMP_HIGHEST
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_MEM_TYPE_VRAM	RSMI_TEMP_LOWEST
rocm smi.h, 57	rocm_smi.h, 56
RSMI_PWR_PROF_PRST_BOOTUP_DEFAULT	RSMI_TEMP_MAX_HYST
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_PWR_PROF_PRST_COMPUTE_MASK	RSMI_TEMP_MAX
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_PWR_PROF_PRST_CUSTOM_MASK	RSMI_TEMP_MIN_HYST
rocm_smi.h, 57	
	rocm_smi.h, 56
RSMI_PWR_PROF_PRST_LAST	RSMI_TEMP_MIN
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_PWR_PROF_PRST_POWER_SAVING_MASK	RSMI_TEMP_OFFSET
rocm_smi.h, 57	rocm_smi.h, 56
RSMI_PWR_PROF_PRST_VIDEO_MASK	rocm_smi.h, 51
rocm_smi.h, 57	RSMI_CLK_TYPE_MEM, 56
RSMI_PWR_PROF_PRST_VR_MASK	RSMI_CLK_TYPE_SYS, 56
rocm_smi.h, 57	RSMI_DEV_PERF_LEVEL_AUTO, 56
RSMI_STATUS_FILE_ERROR	RSMI_DEV_PERF_LEVEL_HIGH, 56
rocm_smi.h, 55	RSMI_DEV_PERF_LEVEL_LOW, 56
RSMI_STATUS_INIT_ERROR	RSMI_DEV_PERF_LEVEL_MANUAL, 56
rocm_smi.h, 55	RSMI_DEV_PERF_LEVEL_STABLE_MIN_MCLK,
RSMI_STATUS_INPUT_OUT_OF_BOUNDS	56
rocm_smi.h, 55	RSMI_DEV_PERF_LEVEL_STABLE_MIN_SCLK,
RSMI_STATUS_INSUFFICIENT_SIZE	56
rocm_smi.h, 55	RSMI DEV PERF LEVEL STABLE PEAK, 56
RSMI STATUS INTERNAL EXCEPTION	RSMI_DEV_PERF_LEVEL_STABLE_STD, 56
rocm smi.h, 55	RSMI_DEV_PERF_LEVEL_UNKNOWN, 56
<del>-</del> :	
RSMI_STATUS_INVALID_ARGS	RSMI_FREQ_IND_INVALID, 57
rocm_smi.h, 55	RSMI_FREQ_IND_MAX, 57
RSMI_STATUS_NOT_FOUND	RSMI_FREQ_IND_MIN, 57
rocm_smi.h, 55	RSMI_MAX_FAN_SPEED, 55
RSMI_STATUS_NOT_SUPPORTED	RSMI_MEM_TYPE_GTT, 57
rocm_smi.h, 55	RSMI_MEM_TYPE_VIS_VRAM, 57
RSMI_STATUS_NOT_YET_IMPLEMENTED	RSMI_MEM_TYPE_VRAM, 57
rocm_smi.h, 55	RSMI_PWR_PROF_PRST_BOOTUP_DEFAULT,
RSMI_STATUS_OUT_OF_RESOURCES	57
rocm_smi.h, 55	RSMI_PWR_PROF_PRST_COMPUTE_MASK, 57
RSMI_STATUS_PERMISSION	RSMI_PWR_PROF_PRST_CUSTOM_MASK, 57
rocm_smi.h, 55	RSMI PWR PROF PRST LAST, 57
RSMI_STATUS_SUCCESS	RSMI_PWR_PROF_PRST_POWER_SAVING_
rocm_smi.h, 55	MASK, 57
RSMI_STATUS_UNKNOWN_ERROR	RSMI_PWR_PROF_PRST_VIDEO_MASK, 57
rocm_smi.h, 55	RSMI_PWR_PROF_PRST_VR_MASK, 57
RSMI_TEMP_CRIT_MIN_HYST	RSMI_STATUS_FILE_ERROR, 55
rocm_smi.h, 56	RSMI_STATUS_INIT_ERROR, 55
RSMI_TEMP_CRIT_MIN	RSMI_STATUS_INPUT_OUT_OF_BOUNDS, 55
rocm_smi.h, 56	RSMI_STATUS_INSUFFICIENT_SIZE, 55
RSMI_TEMP_CRITICAL_HYST	RSMI_STATUS_INTERNAL_EXCEPTION, 55
rocm_smi.h, 56	RSMI_STATUS_INVALID_ARGS, 55
RSMI_TEMP_CRITICAL	RSMI_STATUS_NOT_FOUND, 55

INDEX 61

RSMI_STATUS_NOT_SUPPORTED, 55	Clock, Power and Performance Queries, 33
RSMI_STATUS_NOT_YET_IMPLEMENTED, 55	rsmi_dev_od_volt_info_get
RSMI_STATUS_OUT_OF_RESOURCES, 55	Clock, Power and Performance Queries, 32
RSMI_STATUS_PERMISSION, 55	rsmi_dev_overdrive_level_get
RSMI STATUS SUCCESS, 55	Clock, Power and Performance Queries, 32
RSMI_STATUS_UNKNOWN_ERROR, 55	rsmi_dev_overdrive_level_set
RSMI_TEMP_CRIT_MIN_HYST, 56	Clock, Power and Performance Control, 35
RSMI_TEMP_CRIT_MIN, 56	rsmi_dev_pci_bandwidth_get
RSMI_TEMP_CRITICAL_HYST, 56	PCle Queries, 17
RSMI_TEMP_CRITICAL, 56	rsmi_dev_pci_bandwidth_set
RSMI_TEMP_CURRENT, 56	PCIe Control, 19
RSMI TEMP EMERGENCY HYST, 56	rsmi_dev_pci_id_get
RSMI_TEMP_EMERGENCY, 56	PCle Queries, 17
RSMI TEMP HIGHEST, 56	rsmi_dev_pci_throughput_get
RSMI TEMP LOWEST, 56	PCIe Queries, 18
RSMI_TEMP_MAX_HYST, 56	rsmi_dev_perf_level_get
RSMI_TEMP_MAX, 56	Clock, Power and Performance Queries, 31
RSMI_TEMP_MIN_HYST, 56	rsmi_dev_perf_level_set
RSMI_TEMP_MIN, 56	Clock, Power and Performance Control, 35
RSMI TEMP OFFSET, 56	rsmi dev perf level t
rsmi_clk_type_t, 56	rocm smi.h, 55
rsmi_dev_perf_level_t, 55	rsmi dev power ave get
rsmi_freq_ind_t, 57	Power Queries, 20
rsmi memory type t, 57	
	rsmi_dev_power_cap_get Power Queries, 20
rsmi_power_profile_preset_masks_t, 56	
rsmi_status_t, 55	rsmi_dev_power_cap_range_get
rsmi_temperature_metric_t, 56	Power Queries, 21
rsmi_clk_type_t	rsmi_dev_power_cap_set
rocm_smi.h, 56	Power Control, 22
rsmi_dev_busy_percent_get	rsmi_dev_power_profile_presets_get
Clock, Power and Performance Queries, 31	Clock, Power and Performance Queries, 33
rsmi_dev_error_count_get	rsmi_dev_power_profile_set
Error Queries, 40	Power Control, 22
rsmi_dev_fan_reset	rsmi_dev_subsystem_id_get
Physical State Control, 29	Identifier Queries, 15
rsmi_dev_fan_rpms_get	rsmi_dev_subsystem_name_get
Physical State Queries, 26	Identifier Queries, 15
rsmi_dev_fan_speed_get	rsmi_dev_subsystem_vendor_id_get
Physical State Queries, 26	Identifier Queries, 16
rsmi_dev_fan_speed_max_get	rsmi_dev_temp_metric_get
Physical State Queries, 27	Physical State Queries, 27
rsmi_dev_fan_speed_set	rsmi_dev_vbios_version_get
Physical State Control, 29	Version Queries, 38
rsmi_dev_gpu_clk_freq_get	rsmi_dev_vendor_id_get
Clock, Power and Performance Queries, 32	Identifier Queries, 14
rsmi_dev_gpu_clk_freq_set	rsmi_dev_vendor_name_get
Clock, Power and Performance Control, 36	Identifier Queries, 15
rsmi_dev_id_get	rsmi_error_count_t, 43
Identifier Queries, 13	rsmi_freq_ind_t
rsmi_dev_memory_total_get	rocm_smi.h, 57
Memory Queries, 24	rsmi_freq_volt_region_t, 43
rsmi_dev_memory_usage_get	rsmi_frequencies_t, 44
Memory Queries, 24	current, 44
rsmi_dev_name_get	frequency, 44
Identifier Queries, 14	num_supported, 44
rsmi_dev_od_freq_range_set	rsmi init
Clock, Power and Performance Control, 36	Initialization and Shutdown, 11
rsmi_dev_od_volt_curve_regions_get	rsmi_memory_type_t
	_ /-// -

62 INDEX

```
rocm_smi.h, 57
rsmi_num_monitor_devices
     Identifier Queries, 13
rsmi_od_vddc_point_t, 45
rsmi_od_volt_curve_t, 45
    vc points, 46
rsmi_od_volt_freq_data_t, 46
    curr_mclk_range, 46
rsmi pcie bandwidth t, 47
    lanes, 47
    transfer_rate, 47
rsmi_power_profile_preset_masks_t
    rocm_smi.h, 56
rsmi_power_profile_status_t, 47
    available_profiles, 48
    current, 48
     num profiles, 48
rsmi_range_t, 48
rsmi_shut_down
    Initialization and Shutdown, 11
rsmi status string
     Error Queries, 40
rsmi_status_t
     rocm_smi.h, 55
rsmi_temperature_metric_t
    rocm_smi.h, 56
rsmi_version_get
     Version Queries, 38
rsmi_version_t, 49
transfer rate
    rsmi_pcie_bandwidth_t, 47
vc points
    rsmi_od_volt_curve_t, 46
Version Queries, 38
    rsmi_dev_vbios_version_get, 38
     rsmi_version_get, 38
```