

AMDTPowerProfileAPI

Generated by Doxygen 1.6.1

Mon Apr 11 05:56:10 2016



# Contents

<b>1</b>	<b>CodeXL Power Profiler API</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	Power Profiling . . . . .	9
5.1.1	Detailed Description . . . . .	11
5.1.2	Enumeration Type Documentation . . . . .	12
5.1.2.1	AMDTPwrProfileMode . . . . .	12
5.1.2.2	AMDTPwrDeviceType . . . . .	12
5.1.2.3	AMDTPwrCategory . . . . .	12
5.1.2.4	AMDTPwrAggregation . . . . .	13
5.1.2.5	AMDTPwrUnit . . . . .	13
5.1.2.6	AMDTPwrProfileState . . . . .	14
5.1.2.7	AMDTPwrSampleValueOption . . . . .	14
5.1.2.8	AMDTPwrApuPStates . . . . .	14
5.1.3	Function Documentation . . . . .	15
5.1.3.1	AMDTPwrProfileInitialize . . . . .	15
5.1.3.2	AMDTPwrGetSystemTopology . . . . .	16
5.1.3.3	AMDTPwrGetDeviceCounters . . . . .	16

5.1.3.4	AMDTPwrGetCounterDesc . . . . .	17
5.1.3.5	AMDTPwrEnableCounter . . . . .	17
5.1.3.6	AMDTPwrDisableCounter . . . . .	18
5.1.3.7	AMDTPwrEnableAllCounters . . . . .	19
5.1.3.8	AMDTPwrGetMinimalTimerSamplingPeriod . . . . .	19
5.1.3.9	AMDTPwrSetTimerSamplingPeriod . . . . .	20
5.1.3.10	AMDTPwrStartProfiling . . . . .	20
5.1.3.11	AMDTPwrStopProfiling . . . . .	21
5.1.3.12	AMDTPwrPauseProfiling . . . . .	21
5.1.3.13	AMDTPwrResumeProfiling . . . . .	22
5.1.3.14	AMDTPwrGetProfilingState . . . . .	22
5.1.3.15	AMDTPwrProfileClose . . . . .	22
5.1.3.16	AMDTPwrSetSampleValueOption . . . . .	23
5.1.3.17	AMDTPwrGetSampleValueOption . . . . .	23
5.1.3.18	AMDTPwrReadAllEnabledCounters . . . . .	24
5.1.3.19	AMDTPwrReadCounterHistogram . . . . .	24
5.1.3.20	AMDTPwrReadCumulativeCounter . . . . .	25
5.1.3.21	AMDTPwrGetTimerSamplingPeriod . . . . .	26
5.1.3.22	AMDTPwrIsCounterEnabled . . . . .	26
5.1.3.23	AMDTPwrGetNumEnabledCounters . . . . .	27
5.1.3.24	AMDTPwrGetApuPstateInfo . . . . .	27
5.1.3.25	AMDTPwrGetCounterHierarchy . . . . .	28
5.1.3.26	AMDTPwrGetNodeTemperature . . . . .	28
5.1.3.27	AMDTEnableProcessProfiling . . . . .	29
5.1.3.28	AMDTReadProcessProfileData . . . . .	29
<b>6</b>	<b>Data Structure Documentation</b>	<b>31</b>
6.1	AMDTPwrApuPstate Struct Reference . . . . .	31
6.1.1	Detailed Description . . . . .	31
6.1.2	Field Documentation . . . . .	31
6.1.2.1	m_state . . . . .	31
6.1.2.2	m_isBoosted . . . . .	31
6.1.2.3	m_frequency . . . . .	32
6.2	AMDTPwrApuPstateList Struct Reference . . . . .	33

6.2.1	Detailed Description	33
6.2.2	Field Documentation	33
6.2.2.1	m_cnt	33
6.2.2.2	m_stateInfo	33
6.3	AMDTPwrCounterDesc Struct Reference	34
6.3.1	Detailed Description	34
6.3.2	Field Documentation	34
6.3.2.1	m_counterID	34
6.3.2.2	m_deviceId	34
6.3.2.3	m_name	35
6.3.2.4	m_description	35
6.3.2.5	m_category	35
6.3.2.6	m_aggregation	35
6.3.2.7	m_minValue	35
6.3.2.8	m_maxValue	35
6.3.2.9	m_units	35
6.4	AMDTPwrCounterHierarchy Struct Reference	36
6.4.1	Detailed Description	36
6.4.2	Field Documentation	36
6.4.2.1	m_counter	36
6.4.2.2	m_parent	36
6.4.2.3	m_childCnt	36
6.4.2.4	m_pChildList	36
6.5	AMDTPwrCounterValue Struct Reference	37
6.5.1	Detailed Description	37
6.5.2	Field Documentation	37
6.5.2.1	m_counterID	37
6.5.2.2	m_counterValue	37
6.6	AMDTPwrDevice Struct Reference	38
6.6.1	Detailed Description	38
6.6.2	Field Documentation	38
6.6.2.1	m_type	38
6.6.2.2	m_deviceID	38
6.6.2.3	m_pName	38

6.6.2.4	<a href="#">m_pDescription</a>	38
6.6.2.5	<a href="#">m_isAccessible</a>	39
6.6.2.6	<a href="#">m_pFirstChild</a>	39
6.6.2.7	<a href="#">m_pNextDevice</a>	39
6.7	<a href="#">AMDTPwrHistogram Struct Reference</a>	40
6.7.1	<a href="#">Detailed Description</a>	40
6.7.2	<a href="#">Field Documentation</a>	40
6.7.2.1	<a href="#">m_counterId</a>	40
6.7.2.2	<a href="#">m_numOfBins</a>	40
6.7.2.3	<a href="#">m_pRange</a>	40
6.7.2.4	<a href="#">m_pBins</a>	40
6.8	<a href="#">AMDTPwrInstrumentedPowerData Struct Reference</a>	41
6.8.1	<a href="#">Detailed Description</a>	41
6.8.2	<a href="#">Field Documentation</a>	41
6.8.2.1	<a href="#">m_name</a>	41
6.8.2.2	<a href="#">m_userBuffer</a>	41
6.8.2.3	<a href="#">m_systemStartTime</a>	41
6.8.2.4	<a href="#">m_startTs</a>	41
6.8.2.5	<a href="#">m_endTs</a>	42
6.8.2.6	<a href="#">m_pidInfo</a>	42
6.9	<a href="#">AMDTPwrProcessInfo Struct Reference</a>	43
6.9.1	<a href="#">Detailed Description</a>	43
6.9.2	<a href="#">Field Documentation</a>	43
6.9.2.1	<a href="#">m_pid</a>	43
6.9.2.2	<a href="#">m_sampleCnt</a>	43
6.9.2.3	<a href="#">m_power</a>	43
6.9.2.4	<a href="#">m_ipc</a>	43
6.9.2.5	<a href="#">m_name</a>	44
6.9.2.6	<a href="#">m_path</a>	44
6.10	<a href="#">AMDTPwrSample Struct Reference</a>	45
6.10.1	<a href="#">Detailed Description</a>	45
6.10.2	<a href="#">Field Documentation</a>	45
6.10.2.1	<a href="#">m_systemTime</a>	45
6.10.2.2	<a href="#">m_elapsedTimeMs</a>	45

---

6.10.2.3	m_recordId	45
6.10.2.4	m_numOfValues	45
6.10.2.5	m_counterValues	46
6.11	AMDTPwrSystemTime Struct Reference	47
6.11.1	Detailed Description	47
6.11.2	Field Documentation	47
6.11.2.1	m_second	47
6.11.2.2	m_microSecond	47
<b>7</b>	<b>File Documentation</b>	<b>49</b>
7.1	AMDTDefinitions.h File Reference	49
7.1.1	Detailed Description	51
7.1.2	Define Documentation	51
7.1.2.1	AMDT_STATUS_OK	51
7.1.2.2	AMDT_ERROR_FAIL	51
7.1.2.3	AMDT_ERROR_INVALIDARG	51
7.1.2.4	AMDT_ERROR_OUTOFMEMORY	51
7.1.2.5	AMDT_ERROR_UNEXPECTED	51
7.1.2.6	AMDT_ERROR_ACCESSDENIED	52
7.1.2.7	AMDT_ERROR_HANDLE	52
7.1.2.8	AMDT_ERROR_ABORT	52
7.1.2.9	AMDT_ERROR_NOTIMPL	52
7.1.2.10	AMDT_ERROR_NOFILE	52
7.1.2.11	AMDT_ERROR_INVALIDPATH	52
7.1.2.12	AMDT_ERROR_INVALIDDATA	52
7.1.2.13	AMDT_ERROR_NOTAVAILABLE	52
7.1.2.14	AMDT_ERROR_NODATA	53
7.1.2.15	AMDT_ERROR_LOCKED	53
7.1.2.16	AMDT_ERROR_TIMEOUT	53
7.1.2.17	AMDT_STATUS_PENDING	53
7.1.2.18	AMDT_ERROR_NOTSUPPORTED	53
7.1.2.19	AMDT_ERROR_DRIVER_ALREADY_- INITIALIZED	53
7.1.2.20	AMDT_ERROR_DRIVER_UNAVAILABLE	53

7.1.2.21	AMDT_WARN_SMU_DISABLED . . . . .	53
7.1.2.22	AMDT_WARN_IGPU_DISABLED . . . . .	54
7.1.2.23	AMDT_ERROR_DRIVER_UNINITIALIZED . . . . .	54
7.1.2.24	AMDT_ERROR_INVALID_DEVICEID . . . . .	54
7.1.2.25	AMDT_ERROR_INVALID_COUNTERID . . . . .	54
7.1.2.26	AMDT_ERROR_COUNTER_ALREADY_- ENABLED . . . . .	54
7.1.2.27	AMDT_ERROR_NO_WRITE_PERMISSION . . . . .	54
7.1.2.28	AMDT_ERROR_COUNTER_NOT_ENABLED . . . . .	54
7.1.2.29	AMDT_ERROR_TIMER_NOT_SET . . . . .	55
7.1.2.30	AMDT_ERROR_PROFILE_DATAFILE_NOT_SET . . . . .	55
7.1.2.31	AMDT_ERROR_PROFILE_ALREADY_STARTED . . . . .	55
7.1.2.32	AMDT_ERROR_PROFILE_NOT_STARTED . . . . .	55
7.1.2.33	AMDT_ERROR_PROFILE_NOT_PAUSED . . . . .	55
7.1.2.34	AMDT_ERROR_PROFILE_DATA_NOT_- AVAILABLE . . . . .	55
7.1.2.35	AMDT_ERROR_PLATFORM_NOT_SUPPORTED . . . . .	55
7.1.2.36	AMDT_ERROR_INTERNAL . . . . .	56
7.1.2.37	AMDT_DRIVER_VERSION_MISMATCH . . . . .	56
7.1.2.38	AMDT_ERROR_BIOS_VERSION_NOT_- SUPPORTED . . . . .	56
7.1.2.39	AMDT_ERROR_PROFILE_ALREADY_- CONFIGURED . . . . .	56
7.1.2.40	AMDT_ERROR_PROFILE_NOT_CONFIGURED . . . . .	56
7.1.2.41	AMDT_ERROR_PROFILE_SESSION_EXISTS . . . . .	56
7.1.2.42	AMDT_ERROR_SMU_ACCESS_FAILED . . . . .	56
7.1.2.43	AMDT_ERROR_COUNTERS_NOT_ENABLED . . . . .	57
7.1.2.44	AMDT_ERROR_PREVIOUS_SESSION_NOT_- CLOSED . . . . .	57
7.1.2.45	AMDT_ERROR_COUNTER_NOHIERARCHY . . . . .	57
7.1.2.46	AMDT_ERROR_COUNTER_NOT_ACCESSIBLE . . . . .	57
7.1.2.47	AMDT_ERROR_HYPERVISOR_NOT_- SUPPORTED . . . . .	57
7.1.2.48	AMDT_WARN_PROCESS_PROFILE_NOT_- SUPPORTED . . . . .	57
7.1.2.49	AMDT_ERROR_MARKER_NOT_SET . . . . .	57



---

7.1.3	Typedef Documentation . . . . .	58
7.1.3.1	AMDTResult . . . . .	58
7.2	AMDTPowerProfileApi.h File Reference . . . . .	59
7.2.1	Detailed Description . . . . .	60
7.3	AMDTPowerProfileDataTypes.h File Reference . . . . .	61
7.3.1	Detailed Description . . . . .	62
7.3.2	Define Documentation . . . . .	63
7.3.2.1	AMDT_PWR_ALL_DEVICES . . . . .	63
7.3.2.2	AMDT_PWR_ALL_COUNTERS . . . . .	63
7.3.2.3	AMDT_PWR_EXE_NAME_LENGTH . . . . .	63
7.3.2.4	AMDT_PWR_EXE_PATH_LENGTH . . . . .	63
7.3.2.5	AMDT_MAX_PSTATES . . . . .	63
7.3.2.6	AMDT_PWR_MARKER_BUFFER_LENGTH . . . . .	63
7.3.3	Typedef Documentation . . . . .	64
7.3.3.1	AMDTPwrDeviceId . . . . .	64
<b>8</b>	<b>Example Documentation</b>	<b>65</b>
8.1	CollectAllCounters.cpp . . . . .	65



# Chapter 1

## CodeXL Power Profiler API

The AMDTPwrProfileAPI is a powerful library to help analyze the energy efficiency of systems based on AMD CPUs, APU's and Discrete GPU's.

This API:

- Provides counters to read the power, thermal and frequency characteristics of APU/dGPU and their subcomponents.
- Supports AMD APU's (Kaveri, Temash, Mullins, Carrizo), Discrete GPU's (Tonga, Iceland, Bonaire, Hawaii and other newer graphics cards)
- Supports AMD FirePro discrete GPU cards (W9100, W8100, W7100, W5100 and other newer graphics cards).
- Supports Microsoft Windows as a dynamically loaded library or as a static library.
- Supports Linux as a shared library.
- Manages memory automatically - no allocation and free required.

Using this API, counter values can be read at regular sampling interval. Before any profiling done, the [AMDTPwrProfileInitialize\(\)](#) API must be called. When all the profiling is finished, the [AMDTPwrProfileClose\(\)](#) API must be called. Upon successful completion all the APIs will return AMDT\_STATUS\_OK, otherwise they return appropriate error codes.



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Power Profiling . . . . .	9
---------------------------	---



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">AMDTPwrApuPstate</a>	31
<a href="#">AMDTPwrApuPstateList</a>	33
<a href="#">AMDTPwrCounterDesc</a>	34
<a href="#">AMDTPwrCounterHierarchy</a>	36
<a href="#">AMDTPwrCounterValue</a>	37
<a href="#">AMDTPwrDevice</a>	38
<a href="#">AMDTPwrHistogram</a>	40
<a href="#">AMDTPwrInstrumentedPowerData</a>	41
<a href="#">AMDTPwrProcessInfo</a>	43
<a href="#">AMDTPwrSample</a>	45
<a href="#">AMDTPwrSystemTime</a>	47





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AMDDefinitions.h</a> (Basic data type definitions and error codes used by the AMD CodeXL Power Profiler APIs ) . . . . .	49
<a href="#">AMDTPowerProfileApi.h</a> (AMD Power Profiler APIs to configure, control and collect the power profile counters ) . . . . .	59
<a href="#">AMDTPowerProfileDataTypes.h</a> (Data types and structure definitions used by CodeXL Power Profiler APIs ) . . . . .	61



# Chapter 5

## Module Documentation

### 5.1 Power Profiling

AMDT Power Profiler APIs.

#### Data Structures

- struct [AMDTPwrDevice](#)
- struct [AMDTPwrCounterDesc](#)
- struct [AMDTPwrCounterValue](#)
- struct [AMDTPwrSystemTime](#)
- struct [AMDTPwrSample](#)
- struct [AMDTPwrApuPstate](#)
- struct [AMDTPwrApuPstateList](#)
- struct [AMDTPwrCounterHierarchy](#)
- struct [AMDTPwrHistogram](#)
- struct [AMDTPwrProcessInfo](#)
- struct [AMDTPwrInstrumentedPowerData](#)

#### Enumerations

- enum [AMDTPwrProfileMode](#) { [AMDT\\_PWR\\_PROFILE\\_MODE\\_ONLINE](#), [AMDT\\_PWR\\_PROFILE\\_MODE\\_OFFLINE](#) }
- enum [AMDTPwrDeviceType](#) {  
    [AMDT\\_PWR\\_DEVICE\\_SYSTEM](#),      [AMDT\\_PWR\\_DEVICE\\_PACKAGE](#),  
    [AMDT\\_PWR\\_DEVICE\\_CPU\\_COMPUTE\\_UNIT](#), [AMDT\\_PWR\\_DEVICE\\_CPU\\_CORE](#),  
    [AMDT\\_PWR\\_DEVICE\\_INTERNAL\\_GPU](#),      [AMDT\\_PWR\\_DEVICE\\_EXTERNAL\\_GPU](#), [AMDT\\_PWR\\_DEVICE\\_SVI2](#), [AMDT\\_PWR\\_DEVICE\\_CNT](#) }

- enum `AMDTPwrCategory` {  
`AMDT_PWR_CATEGORY_POWER,`            `AMDT_PWR_CATEGORY_-`  
`FREQUENCY,`   `AMDT_PWR_CATEGORY_TEMPERATURE,`   `AMDT_-`  
`PWR_CATEGORY_VOLTAGE,`  
`AMDT_PWR_CATEGORY_CURRENT,` `AMDT_PWR_CATEGORY_DVFS,`  
`AMDT_PWR_CATEGORY_PROCESS,` `AMDT_PWR_CATEGORY_TIME,`  
`AMDT_PWR_CATEGORY_COUNT,` `AMDT_PWR_CATEGORY_CNT` }
- enum `AMDTPwrAggregation` { `AMDT_PWR_VALUE_SINGLE,` `AMDT_-`  
`PWR_VALUE_CUMULATIVE,`            `AMDT_PWR_VALUE_HISTOGRAM,`  
`AMDT_PWR_VALUE_CNT` }
- enum `AMDTPwrUnit` {  
`AMDT_PWR_UNIT_TYPE_COUNT,`            `AMDT_PWR_UNIT_TYPE_-`  
`PERCENT,`   `AMDT_PWR_UNIT_TYPE_RATIO,`   `AMDT_PWR_UNIT_-`  
`TYPE_MILLI_SECOND,`  
`AMDT_PWR_UNIT_TYPE_JOULE,`    `AMDT_PWR_UNIT_TYPE_WATT,`  
`AMDT_PWR_UNIT_TYPE_VOLT,`    `AMDT_PWR_UNIT_TYPE_MILLI_-`  
`AMPERE,`  
`AMDT_PWR_UNIT_TYPE_MEGA_HERTZ,` `AMDT_PWR_UNIT_TYPE_-`  
`CENTIGRADE,` `AMDT_PWR_UNIT_TYPE_CNT` }
- enum `AMDTPwrProfileState` {  
`AMDT_PWR_PROFILE_STATE_UNINITIALIZED,`            `AMDT_PWR_-`  
`PROFILE_STATE_IDLE,`            `AMDT_PWR_PROFILE_STATE_RUNNING,`  
`AMDT_PWR_PROFILE_STATE_PAUSED,`  
`AMDT_PWR_PROFILE_STATE_STOPPED,`            `AMDT_PWR_PROFILE_-`  
`STATE_ABORTED,` `AMDT_PWR_PROFILE_STATE_CNT` }
- enum `AMDTSampleValueOption` { `AMDT_PWR_SAMPLE_VALUE_-`  
`INSTANTANEOUS,` `AMDT_PWR_SAMPLE_VALUE_LIST,` `AMDT_PWR_-`  
`SAMPLE_VALUE_AVERAGE,`            `AMDT_PWR_SAMPLE_VALUE_CNT`  
}
- enum `AMDTApuPStates` {  
`AMDT_PWR_PSTATE_PB0,` `AMDT_PWR_PSTATE_PB1,` `AMDT_PWR_-`  
`PSTATE_PB2,` `AMDT_PWR_PSTATE_PB3,`  
`AMDT_PWR_PSTATE_PB4,` `AMDT_PWR_PSTATE_PB5,` `AMDT_PWR_-`  
`PSTATE_PB6,` `AMDT_PWR_PSTATE_P0,`  
`AMDT_PWR_PSTATE_P1,`   `AMDT_PWR_PSTATE_P2,`   `AMDT_PWR_-`  
`PSTATE_P3,` `AMDT_PWR_PSTATE_P4,`  
`AMDT_PWR_PSTATE_P5,`   `AMDT_PWR_PSTATE_P6,`   `AMDT_PWR_-`  
`PSTATE_P7` }

## Functions

- `AMDTResult` `AMDTPwrProfileInitialize` (`AMDTPwrProfileMode` `profile-`  
`Mode`)
- `AMDTResult`            `AMDTPwrGetSystemTopology`            (`AMDTPwrDevice`  
`**ppTopology`)

- [AMDTRResult AMDTPwrGetDeviceCounters](#) ([AMDTPwrDeviceId](#) deviceId, [AMDТУInt32](#) \*pNumCounters, [AMDTPwrCounterDesc](#) \*\*ppCounterDescs)
- [AMDTRResult AMDTPwrGetCounterDesc](#) ([AMDТУInt32](#) counterId, [AMDTPwrCounterDesc](#) \*pCounterDesc)
- [AMDTRResult AMDTPwrEnableCounter](#) ([AMDТУInt32](#) counterId)
- [AMDTRResult AMDTPwrDisableCounter](#) ([AMDТУInt32](#) counterId)
- [AMDTRResult AMDTPwrEnableAllCounters](#) ()
- [AMDTRResult AMDTPwrGetMinimalTimerSamplingPeriod](#) ([AMDТУInt32](#) \*pIntervalMilliSec)
- [AMDTRResult AMDTPwrSetTimerSamplingPeriod](#) ([AMDТУInt32](#) interval)
- [AMDTRResult AMDTPwrStartProfiling](#) ()
- [AMDTRResult AMDTPwrStopProfiling](#) ()
- [AMDTRResult AMDTPwrPauseProfiling](#) ()
- [AMDTRResult AMDTPwrResumeProfiling](#) ()
- [AMDTRResult AMDTPwrGetProfilingState](#) ([AMDTPwrProfileState](#) \*pState)
- [AMDTRResult AMDTPwrProfileClose](#) ()
- [AMDTRResult AMDTPwrSetSampleValueOption](#) ([AMDTSampleValueOption](#) opt)
- [AMDTRResult AMDTPwrGetSampleValueOption](#) ([AMDTSampleValueOption](#) \*pOpt)
- [AMDTRResult AMDTPwrReadAllEnabledCounters](#) ([AMDТУInt32](#) \*pNumOfSamples, [AMDTPwrSample](#) \*\*ppData)
- [AMDTRResult AMDTPwrReadCounterHistogram](#) ([AMDТУInt32](#) counterId, [AMDТУInt32](#) \*pNumEntries, [AMDTPwrHistogram](#) \*\*ppData)
- [AMDTRResult AMDTPwrReadCumulativeCounter](#) ([AMDТУInt32](#) counterId, [AMDТУInt32](#) \*pNumEntries, [AMDТFloat32](#) \*\*ppData)
- [AMDTRResult AMDTPwrGetTimerSamplingPeriod](#) ([AMDТУInt32](#) \*pIntervalMilliSec)
- [AMDTRResult AMDTPwrIsCounterEnabled](#) ([AMDТУInt32](#) counterId)
- [AMDTRResult AMDTPwrGetNumEnabledCounters](#) ([AMDТУInt32](#) \*pCount)
- [AMDTRResult AMDTPwrGetApuPstateInfo](#) ([AMDTPwrApuPstateList](#) \*pList)
- [AMDTRResult AMDTPwrGetCounterHierarchy](#) ([AMDТУInt32](#) counterId, [AMDTPwrCounterHierarchy](#) \*pInfo)
- [AMDTRResult AMDTPwrGetNodeTemperature](#) ([AMDТFloat32](#) \*pNodeTemp)
- [AMDTRResult AMDTEnableProcessProfiling](#) (void)
- [AMDTRResult AMDTReadProcessProfileData](#) ([AMDТУInt32](#) \*pPIDCount, [AMDTPwrProcessInfo](#) \*\*ppData)

### 5.1.1 Detailed Description

AMDT Power Profiler APIs.

## 5.1.2 Enumeration Type Documentation

### 5.1.2.1 enum AMDTPwrProfileMode

Following power profile modes are supported.

**Enumerator:**

*AMDT\_PWR\_PROFILE\_MODE\_ONLINE* Power profile mode is online  
*AMDT\_PWR\_PROFILE\_MODE\_OFFLINE* Power profile mode is offline

Definition at line 58 of file AMDTPowerProfileDataTypes.h.

### 5.1.2.2 enum AMDTDeviceType

Each package (processor node) and its sub-components and dGPUs are considered as devices here. Following are the various types of devices supported by power profiler.

**Enumerator:**

*AMDT\_PWR\_DEVICE\_SYSTEM* Dummy root node. All the top-level devices like CPU,APU,dGPU are its children  
*AMDT\_PWR\_DEVICE\_PACKAGE* In a multi-node system, each node will be a separate package  
*AMDT\_PWR\_DEVICE\_CPU\_COMPUTE\_UNIT* Each CPU Compute-Unit within a package  
*AMDT\_PWR\_DEVICE\_CPU\_CORE* Each CPU core within a CPU Compute-Unit  
*AMDT\_PWR\_DEVICE\_INTERNAL\_GPU* Integrated GPU within a AMD APU  
*AMDT\_PWR\_DEVICE\_EXTERNAL\_GPU* Each AMD dGPU connected in the system  
*AMDT\_PWR\_DEVICE\_SVI2* Serial Voltage Interface 2  
*AMDT\_PWR\_DEVICE\_CNT* Total device count

Definition at line 68 of file AMDTPowerProfileDataTypes.h.

### 5.1.2.3 enum AMDTPwrCategory

Following is the list of counter category supported by power profiler.

**Enumerator:**

*AMDT\_PWR\_CATEGORY\_POWER* Instantaneous power  
*AMDT\_PWR\_CATEGORY\_FREQUENCY* Frequency  
*AMDT\_PWR\_CATEGORY\_TEMPERATURE* Temperature in centigrade

*AMDT\_PWR\_CATEGORY\_VOLTAGE* Voltage  
*AMDT\_PWR\_CATEGORY\_CURRENT* Current  
*AMDT\_PWR\_CATEGORY\_DVFS* P-State, C-State  
*AMDT\_PWR\_CATEGORY\_PROCESS* PID, TID  
*AMDT\_PWR\_CATEGORY\_TIME* Time  
*AMDT\_PWR\_CATEGORY\_COUNT* Generic count value  
*AMDT\_PWR\_CATEGORY\_CNT* Total category count

Definition at line 83 of file AMDTPowerProfileDataTypes.h.

#### 5.1.2.4 enum AMDTPwrAggregation

Following is the list of aggregation types supported by power profiler.

##### Enumerator:

*AMDT\_PWR\_VALUE\_SINGLE* Single instantaneous value  
*AMDT\_PWR\_VALUE\_CUMULATIVE* Cumulative value  
*AMDT\_PWR\_VALUE\_HISTOGRAM* Histogram value  
*AMDT\_PWR\_VALUE\_CNT* Total power value

Definition at line 100 of file AMDTPowerProfileDataTypes.h.

#### 5.1.2.5 enum AMDTPwrUnit

Various unit types for the output values for the counter types.

##### Enumerator:

*AMDT\_PWR\_UNIT\_TYPE\_COUNT* Count index  
*AMDT\_PWR\_UNIT\_TYPE\_PERCENT* Percentage  
*AMDT\_PWR\_UNIT\_TYPE\_RATIO* Ratio  
*AMDT\_PWR\_UNIT\_TYPE\_MILLI\_SECOND* Time in milli seconds  
*AMDT\_PWR\_UNIT\_TYPE\_JOULE* Energy consumption  
*AMDT\_PWR\_UNIT\_TYPE\_WATT* Power consumption  
*AMDT\_PWR\_UNIT\_TYPE\_VOLT* Voltage  
*AMDT\_PWR\_UNIT\_TYPE\_MILLI\_AMPERE* Current  
*AMDT\_PWR\_UNIT\_TYPE\_MEGA\_HERTZ* Frequency type unit  
*AMDT\_PWR\_UNIT\_TYPE\_CENTIGRADE* Temperature type unit  
*AMDT\_PWR\_UNIT\_TYPE\_CNT* Total power unit

Definition at line 111 of file AMDTPowerProfileDataTypes.h.

### 5.1.2.6 enum AMDTPwrProfileState

States of Power profiler.

#### Enumerator:

**AMDT\_PWR\_PROFILE\_STATE\_UNINITIALIZED** Profiler is not initialized

**AMDT\_PWR\_PROFILE\_STATE\_IDLE** Profiler is initialized

**AMDT\_PWR\_PROFILE\_STATE\_RUNNING** Profiler is running

**AMDT\_PWR\_PROFILE\_STATE\_PAUSED** Profiler is paused

**AMDT\_PWR\_PROFILE\_STATE\_STOPPED** Profiler is Stopped

**AMDT\_PWR\_PROFILE\_STATE\_ABORTED** Profiler is aborted

**AMDT\_PWR\_PROFILE\_STATE\_CNT** Total number of profiler states

Definition at line 129 of file AMDTPowerProfileDataTypes.h.

### 5.1.2.7 enum AMDTSampleValueOption

Options to retrieve the profiled counter data using AMDTPwrReadAllEnabledCounters function

#### Enumerator:

**AMDT\_PWR\_SAMPLE\_VALUE\_INSTANTANEOUS** Default. The latest/instantaneous

**AMDT\_PWR\_SAMPLE\_VALUE\_LIST** List of sampled counter values

**AMDT\_PWR\_SAMPLE\_VALUE\_AVERAGE** Average of the sampled counter

**AMDT\_PWR\_SAMPLE\_VALUE\_CNT** Maximum Sample value count

Definition at line 143 of file AMDTPowerProfileDataTypes.h.

### 5.1.2.8 enum AMDTApuPStates

P-States can be either hardware or software P-States. Hardware P-States are also known as Boosted P-States. These are defined as AMDT\_PWR\_PSTATES\_PBx. The Software P-States are defined as AMDT\_PWR\_PSTATES\_Px, where x is the P-State number. Hardware(Boosted) P-States are not software visible.

#### Enumerator:

**AMDT\_PWR\_PSTATE\_PB0** Boosted P-State 0

**AMDT\_PWR\_PSTATE\_PB1** Boosted P-State 1

**AMDT\_PWR\_PSTATE\_PB2** Boosted P-State 2

**AMDT\_PWR\_PSTATE\_PB3** Boosted P-State 3

**AMDT\_PWR\_PSTATE\_PB4** Boosted P-State 4



***AMDT\_PWR\_PSTATE\_PB5*** Boosted P-State 5  
***AMDT\_PWR\_PSTATE\_PB6*** Boosted P-State 6  
***AMDT\_PWR\_PSTATE\_P0*** Software P-State 0  
***AMDT\_PWR\_PSTATE\_P1*** Software P-State 1  
***AMDT\_PWR\_PSTATE\_P2*** Software P-State 2  
***AMDT\_PWR\_PSTATE\_P3*** Software P-State 3  
***AMDT\_PWR\_PSTATE\_P4*** Software P-State 4  
***AMDT\_PWR\_PSTATE\_P5*** Software P-State 5  
***AMDT\_PWR\_PSTATE\_P6*** Software P-State 6  
***AMDT\_PWR\_PSTATE\_P7*** Software P-State 7

Definition at line 157 of file AMDTPowerProfileDataTypes.h.

### 5.1.3 Function Documentation

#### 5.1.3.1 AMDTResult AMDTPwrProfileInitialize (AMDTProfileMode profileMode)

This API loads and initializes the AMDT Power Profile drivers. This API should be the first one to be called.

##### Parameters:

← ***profileMode***,: Client should select any one of the predefined profile modes that are defined in [AMDTPwrProfileMode](#).

##### Returns:

The status of initialization request

##### Return values:

***AMDT\_STATUS\_OK***,: Success  
***AMDT\_ERROR\_INVALIDARG***,: An invalid profileMode parameter was passed  
***AMDT\_ERROR\_DRIVER\_UNAVAILABLE***,: Driver not available  
***AMDT\_ERROR\_DRIVER\_ALREADY\_INITIALIZED***,: Already initialized  
***AMDT\_DRIVER\_VERSION\_MISMATCH***,: Mismatch between the expected and installed driver versions  
***AMDT\_ERROR\_PLATFORM\_NOT\_SUPPORTED***,: Platform not supported  
***AMDT\_WARN\_SMU\_DISABLED***,: SMU is disabled and hence power and thermal values provided by SMU will not be available  
***AMDT\_WARN\_IGPU\_DISABLED***,: Internal GPU is disabled  
***AMDT\_ERROR\_FAIL***,: An internal error occurred  
***AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED***,: Previous session was not closed.

### 5.1.3.2 **AMDTResult AMDTPwrGetSystemTopology (AMDTPwrDevice \*\* ppTopology)**

This API provides device tree that represents the current system topology relevant to power profiler. The nodes (a processor package or a dGPU) and as well as their sub-components are considered as devices. Each device in the tree points to their siblings and children, if any.

#### Parameters:

→ *ppTopology*,: Device tree

#### Returns:

The status of system topology request

#### Return values:

*AMDT\_STATUS\_OK*,: On Success

*AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as ppTopology parameter

*AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

*AMDT\_ERROR\_OUTOFMEMORY*,: Failed to allocate required memory

*AMDT\_ERROR\_FAIL*,: An internal error occurred

### 5.1.3.3 **AMDTResult AMDTPwrGetDeviceCounters (AMDTPwrDeviceId deviceId, AMDTUInt32 \*pNumCounters, AMDTPwrCounterDesc \*\* ppCounterDescs)**

This API provides the list of supported counters for the given device id. If the device id is [AMDT\\_PWR\\_ALL\\_DEVICES](#), then counters for all the available devices will be returned. The pointer returned will be valid till the client calls [AMDTPwrProfileClose\(\)](#) function.

#### Parameters:

← *deviceId*,: The deviceId provided by [AMDTPwrGetSystemTopology\(\)](#) function or [AMDT\\_PWR\\_ALL\\_DEVICES](#) to represent all the devices returned by [AMDTPwrGetSystemTopology\(\)](#)

→ *pNumCounters*,: Number of counters supported by the device

→ *ppCounterDescs*,: Description of each counter supported by the device

#### Returns:

The status of device counter details request

#### Return values:

*AMDT\_STATUS\_OK*,: On Success

**AMDT\_ERROR\_INVALIDARG,:** NULL pointer was passed as ppCounterDescs or pNumCounters parameters

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_INVALID\_DEVICEID,:** invalid deviceId parameter was passed

**AMDT\_ERROR\_OUTOFMEMORY,:** Failed to allocate required memory

**AMDT\_ERROR\_FAIL,:** An internal error occurred

#### 5.1.3.4 AMDTResult AMDTPwrGetCounterDesc (AMDTUInt32 counterId, AMDTPwrCounterDesc \* pCounterDesc)

This API provides the description for the given counter Index.

##### Parameters:

← **counterId,:** Counter index

→ **pCounterDesc,:** Description of the counter which index is counterId

##### Returns:

The status of counter description request

##### Return values:

**AMDT\_STATUS\_OK,:** On Success

**AMDT\_ERROR\_INVALIDARG,:** NULL pointer was passed as pCounterDesc parameter

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_INVALID\_COUNTERID,:** Invalid counterId parameter was passed

**AMDT\_ERROR\_FAIL,:** An internal error occurred

#### 5.1.3.5 AMDTResult AMDTPwrEnableCounter (AMDTUInt32 counterId)

This API will enable the counter to be sampled. This API cannot be used once profile is started.

- If histogram/cumulative counters are enabled along with simple counters, then it is expected that the [AMDTPwrReadAllEnabledCounters\(\)](#) API is regularly called to read the simple counters value. Only then the values for histogram/cumulative counters will be aggregated and the [AMDTPwrReadCounterHistogram\(\)](#) API will return the correct values.

- If only the histogram/cumulative counters are enabled, calling [AMDTPwrReadCounterHistogram\(\)](#) is sufficient to get the values for the enabled histogram/cumulative counters.

**Parameters:**

← *counterId*,: Counter index

**Returns:**

The status of counter enable request

**Return values:**

*AMDT\_STATUS\_OK*,: On Success

*AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

*AMDT\_ERROR\_INVALID\_COUNTERID*,: Invalid *counterId* parameter was passed

*AMDT\_ERROR\_COUNTER\_ALREADY\_ENABLED*,: Specified counter is already enabled

*AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED*,: Counters cannot be enabled on the fly when the profile is already started

*AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED*,: Previous session was not closed

*AMDT\_ERROR\_COUNTER\_NOT\_ACCESSIBLE*,: Counter is not accessible

*AMDT\_ERROR\_FAIL*,: An internal error occurred

**5.1.3.6 AMDTResult AMDTPwrDisableCounter (AMDTUInt32 *counterId*)**

This API will disable the counter to be sampled from the active list. This API cannot be used once profile is started.

**Parameters:**

← *counterId*,: Counter index

**Returns:**

The status of counter disable request

**Return values:**

*AMDT\_STATUS\_OK*,: On Success

*AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

*AMDT\_ERROR\_INVALID\_COUNTERID*,: Invalid *counterId* parameter was passed

***AMDT\_ERROR\_COUNTER\_NOT\_ENABLED,:*** Specified counter is not enabled

***AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED,:*** Counters cannot be disabled on the fly when the profile run is already started

***AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED,:*** Previous session was not closed

***AMDT\_ERROR\_FAIL,:*** An internal error occurred

#### 5.1.3.7 AMDTResult AMDTPwrEnableAllCounters ()

This API will enable all the simple counters. This will NOT enable the histogram counters. This API cannot be used once profile is started.

##### Returns:

The status of enabling all the supported counters request

##### Return values:

***AMDT\_STATUS\_OK,:*** On Success

***AMDT\_ERROR\_FAIL,:*** An internal error occurred

***AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:*** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

***AMDT\_ERROR\_COUNTER\_ALREADY\_ENABLED,:*** Some of the counters are already enabled

***AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED,:*** Counters cannot be enabled on the fly when the profile is already started

***AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED,:*** Previous session was not closed

#### 5.1.3.8 AMDTResult AMDTPwrGetMinimalTimerSamplingPeriod (AMDTUInt32 \* *pIntervalMilliSec*)

This API provides the minimum sampling interval which can be set by the client.

##### Parameters:

→ ***pIntervalMilliSec,:*** The sampling interval in milli-second

##### Returns:

The status of retrieving the minimum supported sampling interval request

##### Return values:

***AMDT\_STATUS\_OK,:*** On Success

**AMDT\_ERROR\_INVALIDARG,:** NULL pointer was passed as pIntervalMilliSec parameter

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_FAIL,:** An internal error occurred

#### 5.1.3.9 AMDTResult AMDTPwrSetTimerSamplingPeriod (AMDTUInt32 interval)

This API will set the driver to periodically sample the counter values and store them in a buffer. This cannot be called once the profile run is started.

##### Parameters:

← **interval,:** sampling period in millisecond

##### Returns:

The status of sampling time set request

##### Return values:

**AMDT\_STATUS\_OK,:** On Success

**AMDT\_ERROR\_INVALIDARG,:** Invalid interval value was passed as IntervalMilliSec parameter

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED,:** Timer interval cannot be changed when the profile is already started

**AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED,:** Previous session was not closed

**AMDT\_ERROR\_FAIL,:** An internal error occurred

#### 5.1.3.10 AMDTResult AMDTPwrStartProfiling ()

This API will start the profiling and the driver will collect the data at regular interval specified by [AMDTPwrSetTimerSamplingPeriod\(\)](#). This has to be called after enabling the required counters by using [AMDTPwrEnableCounter\(\)](#) or [AMDTPwrEnableAllCounters\(\)](#).

##### Returns:

The status of starting the profile

##### Return values:

**AMDT\_STATUS\_OK,:** On Success

***AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:*** [AMDTPwrProfileInitialize](#) function was neither called nor successful  
***AMDT\_ERROR\_TIMER\_NOT\_SET,:*** Sampling timer was not set  
***AMDT\_ERROR\_COUNTERS\_NOT\_ENABLED,:*** No counters are enabled for collecting profile data  
***AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED,:*** Profile is already started  
***AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED,:*** Previous session was not closed  
***AMDT\_ERROR\_BIOS\_VERSION\_NOT\_SUPPORTED,:*** BIOS needs to be upgraded  
***AMDT\_ERROR\_FAIL,:*** An internal error occurred  
***AMDT\_ERROR\_ACCESSDENIED,:*** Profiler is busy, currently not accessible

#### 5.1.3.11 AMDTResult AMDTPwrStopProfiling ()

This APIs will stop the profiling run which was started by [AMDTPwrStartProfiling\(\)](#) function call.

**Returns:**

The status of stopping the profile

**Return values:**

***AMDT\_STATUS\_OK,:*** On Success  
***AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:*** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful  
***AMDT\_ERROR\_PROFILE\_NOT\_STARTED,:*** Profile is not started  
***AMDT\_ERROR\_FAIL,:*** An internal error occurred

#### 5.1.3.12 AMDTResult AMDTPwrPauseProfiling ()

This API will pause the profiling. The driver and the backend will retain the profile configuration details provided by the client.

**Returns:**

The status of pausing the profile

**Return values:**

***AMDT\_STATUS\_OK,:*** On Success  
***AMDT\_ERROR\_FAIL,:*** An internal error occurred  
***AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:*** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful  
***AMDT\_ERROR\_PROFILE\_NOT\_STARTED,:*** Profile not started

### 5.1.3.13 AMDTResult AMDTPwrResumeProfiling ()

This API will resume the profiling which is in paused state.

**Returns:**

The status of resuming the profile

**Return values:**

*AMDT\_STATUS\_OK*,: On Success

*AMDT\_ERROR\_FAIL*,: An internal error occurred

*AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

*AMDT\_ERROR\_PROFILE\_NOT\_PAUSED*,: Profile is not in paused state

### 5.1.3.14 AMDTResult AMDTPwrGetProfilingState (AMDTPwrProfileState \* pState)

This API provides the current state of the profile.

**Parameters:**

→ *pState* Current profile state

**Returns:**

The status of getting the profile state

**Return values:**

*AMDT\_STATUS\_OK*,: On Success

*AMDT\_ERROR\_FAIL*,: An internal error occurred

*AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as pState parameter

### 5.1.3.15 AMDTResult AMDTPwrProfileClose ()

This API will close the power profiler and unregister driver and cleanup all memory allocated during [AMDTPwrProfileInitialize\(\)](#).

**Returns:**

The status of closing the profiler

**Return values:**

*AMDT\_STATUS\_OK*,: On Success

*AMDT\_ERROR\_FAIL*,: An internal error occurred

*AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful



#### 5.1.3.16 AMDTResult AMDTPwrSetSampleValueOption (AMDTSampleValueOption *opt*)

API to set the sample value options to be returned by the [AMDTPwrReadAllEnabled-Counters\(\)](#) function.

##### Parameters:

← *opt*,: One of the output value options defined in AMDTSampleValueOption

##### Returns:

The status of setting the output value option

##### Return values:

**AMDT\_STATUS\_OK**,: On Success

**AMDT\_ERROR\_FAIL**,: An internal error occurred

**AMDT\_ERROR\_INVALIDARG**,: An invalid opt was specified as parameter

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED**,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED**,: Cannot set the sample value option when the profile is running

#### 5.1.3.17 AMDTResult AMDTPwrGetSampleValueOption (AMDTSampleValueOption \**pOpt*)

API to get the sample value option set for the current profile session.

##### Parameters:

→ *pOpt*,: One of the output value options defined in AMDTSampleValueOption

##### Returns:

The status of setting the output value option

##### Return values:

**AMDT\_STATUS\_OK**,: On Success

**AMDT\_ERROR\_FAIL**,: An internal error occurred

**AMDT\_ERROR\_INVALIDARG**,: An invalid opt was specified as parameter

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED**,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

### 5.1.3.18 AMDTResult AMDTPwrReadAllEnabledCounters (AMDTUInt32 \* pNumOfSamples, AMDTPwrSample \*\* ppData)

API to read all the counters that are enabled. This will NOT read the histogram counters. This can return an array of {CounterID, Float-Value}. If there are no new samples, this API will return AMDTResult NO\_NEW\_DATA and pNumOfSamples will point to value of zero. If there are new samples, this API will return AMDT\_STATUS\_OK and pNumOfSamples will point to value greater than zero.

#### Parameters:

- *ppData*,: Processed profile data. No need to allocate or free the memory data is valid till we call this API next time
- *pNumOfSamples*,: Number of sample based on the [AMDTPwrSetSampleValueOption\(\)](#) set

#### Returns:

The status reading all enabled counters

#### Return values:

- AMDT\_STATUS\_OK*,: On Success
- AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as pNumSamples or ppData parameters
- AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful
- AMDT\_ERROR\_PROFILE\_NOT\_STARTED*,: Profile is not started
- AMDT\_ERROR\_PROFILE\_DATA\_NOT\_AVAILABLE*,: Profile data is not yet available
- AMDT\_ERROR\_OUTOFMEMORY*,: Memory not available
- AMDT\_ERROR\_SMU\_ACCESS\_FAILED*,: One of the configured SMU data access has problem it is advisable to stop the profiling session
- AMDT\_ERROR\_FAIL*,: An internal error occurred

### 5.1.3.19 AMDTResult AMDTPwrReadCounterHistogram (AMDTUInt32 counterId, AMDTUInt32 \* pNumEntries, AMDTPwrHistogram \*\* ppData)

API to read one of the derived counters generate histograms from the raw counter values. Since the histogram may contain multiple entries and according to the counter values, a derived histogram counter type specific will be used to provide the output data.

#### Parameters:

- ← *counterId*,: Histogram type counter id. AMDT\_PWR\_ALL\_COUNTERS to represent all supported histogram counters.

- *pNumEntries*,: Number of entries in the histogram
- *ppData*,: Compute histogram data for the given counter id

**Returns:**

The status of reading histogram data

**Return values:**

- AMDT\_STATUS\_OK*,: On Success
- AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as *pNumEntries* or *ppData* parameters
- AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful
- AMDT\_ERROR\_INVALID\_COUNTERID*,: An invalid *counterId* was passed
- AMDT\_ERROR\_PROFILE\_NOT\_STARTED*,: Profile is not started
- AMDT\_ERROR\_PROFILE\_DATA\_NOT\_AVAILABLE*,: Profile data is not yet available
- AMDT\_ERROR\_OUTOFMEMORY*,: Memory not available
- AMDT\_ERROR\_FAIL*,: An internal error occurred

#### 5.1.3.20 AMDTResult AMDTPwrReadCumulativeCounter (AMDTUInt32 *counterId*, AMDTUInt32 \**pNumEntries*, AMDTFloat32 \*\**ppData*)

API to read one of the derived accumulated counters values from the raw counter values.

**Parameters:**

- ← *counterId*,: Cumulative type counter id. *AMDT\_PWR\_ALL\_COUNTERS* to represent all supported accumulated counters.
- *pNumEntries*,: Number of cumulative counters
- *ppData*,: Accumulated counter data for the given counter id

**Returns:**

The status of reading accumulated counter data

**Return values:**

- AMDT\_STATUS\_OK*,: On Success
- AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as *pNumEntries* or *ppData* parameters
- AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful
- AMDT\_ERROR\_INVALID\_COUNTERID*,: An invalid *counterId* was passed

***AMDT\_ERROR\_PROFILE\_NOT\_STARTED,***: Profile is not started  
***AMDT\_ERROR\_PROFILE\_DATA\_NOT\_AVAILABLE,***: Profile data is not yet available  
***AMDT\_ERROR\_OUTOFMEMORY,***: Memory not available  
***AMDT\_ERROR\_FAIL,***: An internal error occurred

#### 5.1.3.21 **AMDTResult AMDTPwrGetTimerSamplingPeriod (AMDTUInt32 \* *pIntervalMilliSec*)**

This API will get the timer sampling period at which the samples are collected by the driver.

##### **Parameters:**

→ *pIntervalMilliSec,*: sampling period in millisecond

##### **Returns:**

The status of the get sampling interval request

##### **Return values:**

***AMDT\_STATUS\_OK,***: On Success  
***AMDT\_ERROR\_INVALIDARG,***: NULL pointer was passed as *pIntervalMilliSec* parameter  
***AMDT\_ERROR\_DRIVER\_UNINITIALIZED,***: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful  
***AMDT\_ERROR\_FAIL,***: An internal error occurred

#### 5.1.3.22 **AMDTResult AMDTPwrIsCounterEnabled (AMDTUInt32 *counterId*)**

This query API is to check whether a counter is enabled for profiling or not.

##### **Parameters:**

← *counterId,*: Counter index

##### **Returns:**

The status of query request.

##### **Return values:**

***AMDT\_STATUS\_OK,***: On Success; Counter is enabled  
***AMDT\_ERROR\_DRIVER\_UNINITIALIZED,***: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful  
***AMDT\_ERROR\_INVALID\_COUNTERID,***: An invalid *counterId* was passed  
***AMDT\_ERROR\_COUNTER\_NOT\_ENABLED,***: Counter is not enabled already  
***AMDT\_ERROR\_FAIL,***: An internal error occurred

### 5.1.3.23 AMDTResult AMDTPwrGetNumEnabledCounters (AMDTUInt32 \* *pCount*)

This query API is to get the number of counters that are enabled for profiling.

**Parameters:**

→ *pCount*,: Number of enabled counters

**Returns:**

The status of query request

**Return values:**

**AMDT\_STATUS\_OK**,: On Success; Counter is enabled

**AMDT\_ERROR\_INVALIDARG**,: NULL pointer is passed as an argument

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED**,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_FAIL**,: An internal error occurred

### 5.1.3.24 AMDTResult AMDTPwrGetApuPstateInfo (AMDTApuPstateList \* *pList*)

API to get the list of pstate supported by the target APU, where power profile is running. List contains both hardware and software P-States with their corresponding frequencies.

**Parameters:**

→ *pList*,: List of P-States

**Returns:**

The status reading the pstate list for the platform

**Return values:**

**AMDT\_STATUS\_OK**,: On Success

**AMDT\_ERROR\_INVALIDARG**,: NULL pointer was passed as argument

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED**,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_PLATFORM\_NOT\_SUPPORTED**,: Platform not supported

**AMDT\_ERROR\_FAIL**,: An internal error occurred

### 5.1.3.25 **AMDTResult AMDTPwrGetCounterHierarchy** (AMDTUInt32 *counterId*, AMDTPwrCounterHierarchy \* *pInfo*)

This API provides the relationship with other counters for the given counter id. For the given counter id, this API provides the parent counter and as well the child counters list.

#### Parameters:

- ← *counterId*,: The counter id for which the dependent counters information is requested
- *pInfo*,: Provides hierarchical relationship for the given counterId

#### Returns:

The status retrieving hierarchical information for the given counters

#### Return values:

- AMDT\_STATUS\_OK*,: On Success
- AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as argument
- AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful
- AMDT\_ERROR\_INVALID\_COUNTERID*,: Invalid counterId parameter was passed
- AMDT\_ERROR\_COUNTER\_NOHIERARCHY*,: Counter does not have any hierarchical relationship
- AMDT\_ERROR\_FAIL*,: An internal error occurred

### 5.1.3.26 **AMDTResult AMDTPwrGetNodeTemperature** (AMDTFloat32 \* *pNodeTemp*)

This API provides the node temperature in Tctl scale. This temperature is not absolute.

#### Parameters:

- *pNodeTemp*,: Provides node temperature.

#### Returns:

The status retrieving hierarchical information for the given counters

#### Return values:

- AMDT\_STATUS\_OK*,: On Success
- AMDT\_ERROR\_INVALIDARG*,: NULL pointer was passed as argument
- AMDT\_ERROR\_DRIVER\_UNINITIALIZED*,: [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful
- AMDT\_ERROR\_FAIL*,: An internal error occurred

**5.1.3.27 AMDTResult AMDTEnableProcessProfiling (void)**

This API enables process profiling. This API will enable backend and driver to collect running PIDs at lowest possible granularity and attribute them against the power values provided by the SMU.

**Returns:**

The status of the process profiling enable request

**Return values:**

**AMDT\_STATUS\_OK,:** On Success

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_PROFILE\_ALREADY\_STARTED,:** Process profiling can not be set when the profile is already started

**AMDT\_WARN\_PROCESS\_PROFILE\_ALREADY\_ENABLED,:** Process profiling already enabled

**AMDT\_ERROR\_OUTOFMEMORY,:** Failed to allocate required memory

**AMDT\_ERROR\_PROCESS\_PROFILE\_NOT\_SUPPORTED,:** Platform not supported

**5.1.3.28 AMDTResult AMDTReadProcessProfileData (AMDTUInt32 \* pPIDCount, AMDTPwrProcessInfo \*\* ppData)**

This API will provide the list of running PIDs so far from the time of profile start and their aggregated power indicators. This API can be called at any point of time from start of the profile to the stop of the profile.

**Parameters:**

→ **pPIDCount,:** Total number of PIDs running during the profile session

→ **ppData,:** List of PIDs with their power indicators

**Returns:**

The status reading process profiling data

**Return values:**

**AMDT\_STATUS\_OK,:** On Success

**AMDT\_ERROR\_INVALIDARG,:** NULL pointer was passed as pData parameters

**AMDT\_ERROR\_DRIVER\_UNINITIALIZED,:** [AMDTPwrProfileInitialize\(\)](#) function was neither called nor successful

**AMDT\_ERROR\_PROFILE\_NOT\_STARTED,:** Profile is not started

***AMDT\_ERROR\_PROFILE\_DATA\_NOT\_AVAILABLE,:*** Profile data is not yet available

***AMDT\_ERROR\_OUTOFMEMORY,:*** Memory not available

***AMDT\_ERROR\_PROCESS\_PROFILE\_NOT\_ENABLED,:*** Process profiling not enabled

***AMDT\_ERROR\_FAIL,:*** An internal error occurred

***AMDT\_ERROR\_PROCESS\_PROFILE\_NOT\_SUPPORTED,:*** Platform not supported



## Chapter 6

# Data Structure Documentation

### 6.1 AMDTPwrApuPstate Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

#### Data Fields

- [AMDTApuPStates m\\_state](#)
- bool [m\\_isBoosted](#)
- AMDTUInt32 [m\\_frequency](#)

#### 6.1.1 Detailed Description

Provides various P-States and their corresponding frequencies.

Definition at line 245 of file AMDTPowerProfileDataTypes.h.

#### 6.1.2 Field Documentation

##### 6.1.2.1 AMDTApuPStates m\_state

P-State number

Definition at line 247 of file AMDTPowerProfileDataTypes.h.

##### 6.1.2.2 bool m\_isBoosted

Boosted P-State flag

Definition at line 248 of file AMDTPowerProfileDataTypes.h.

### 6.1.2.3 AMDTUInt32 m\_frequency

P-State frequency

Definition at line 249 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.2 AMDTPwrApuPstateList Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt32 [m\\_cnt](#)
- [AMDTPwrApuPstate m\\_stateInfo](#) [AMDT\_MAX\_PSTATES]

### 6.2.1 Detailed Description

List of the supported APU P-States details

Definition at line 255 of file AMDTPowerProfileDataTypes.h.

### 6.2.2 Field Documentation

#### 6.2.2.1 AMDTUInt32 m\_cnt

Number of P-States

Definition at line 257 of file AMDTPowerProfileDataTypes.h.

#### 6.2.2.2 AMDTPwrApuPstate m\_stateInfo[AMDT\_MAX\_PSTATES]

P-States list

Definition at line 258 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.3 AMDTPwrCounterDesc Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt32 [m\\_counterID](#)
- AMDTUInt32 [m\\_deviceId](#)
- char \* [m\\_name](#)
- char \* [m\\_description](#)
- AMDTPwrCategory [m\\_category](#)
- AMDTPwrAggregation [m\\_aggregation](#)
- AMDTFloat64 [m\\_minValue](#)
- AMDTFloat64 [m\\_maxValue](#)
- AMDTPwrUnit [m\\_units](#)

### 6.3.1 Detailed Description

Details of a supported power counter and its associated device. Following counter types are supported:

- Simple Counters has `m_aggregation` type as `AMDT_PWR_VALUE_SINGLE`.
- Histogram Counters has `m_aggregation` type as `AMDT_PWR_VALUE_HISTOGRAM`.
- Cumulative Counters has `m_aggregation` type as `AMDT_PWR_VALUE_CUMULATIVE`.

Definition at line 199 of file `AMDTPowerProfileDataTypes.h`.

### 6.3.2 Field Documentation

#### 6.3.2.1 AMDTUInt32 `m_counterID`

Counter index

Definition at line 201 of file `AMDTPowerProfileDataTypes.h`.

#### 6.3.2.2 AMDTUInt32 `m_deviceId`

Device Id

Definition at line 202 of file `AMDTPowerProfileDataTypes.h`.

**6.3.2.3 char\* m\_name**

Name of the counter

Definition at line 203 of file AMDTPowerProfileDataTypes.h.

**6.3.2.4 char\* m\_description**

Description of the counter

Definition at line 204 of file AMDTPowerProfileDataTypes.h.

**6.3.2.5 AMDTPwrCategory m\_category**

Power/Freq/Temperature

Definition at line 205 of file AMDTPowerProfileDataTypes.h.

**6.3.2.6 AMDTPwrAggregation m\_aggregation**

Single/Histogram/Cumulative

Definition at line 206 of file AMDTPowerProfileDataTypes.h.

**6.3.2.7 AMDTFloat64 m\_minValue**

Minimum possible counter value

Definition at line 207 of file AMDTPowerProfileDataTypes.h.

**6.3.2.8 AMDTFloat64 m\_maxValue**

Maximum possible counter value

Definition at line 208 of file AMDTPowerProfileDataTypes.h.

**6.3.2.9 AMDTPwrUnit m\_units**

Seconds/MHz/Joules/Watts/Volt/Ampere

Definition at line 209 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.4 AMDTPwrCounterHierarchy Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt32 [m\\_counter](#)
- AMDTUInt32 [m\\_parent](#)
- AMDTUInt32 [m\\_childCnt](#)
- AMDTUInt32 \* [m\\_pChildList](#)

### 6.4.1 Detailed Description

Provides hierarchical relationship details of a power counter. Both the parent and children counter details will be provided.

Definition at line 265 of file AMDTPowerProfileDataTypes.h.

### 6.4.2 Field Documentation

#### 6.4.2.1 AMDTUInt32 m\_counter

Counter Id

Definition at line 267 of file AMDTPowerProfileDataTypes.h.

#### 6.4.2.2 AMDTUInt32 m\_parent

Parent counter Id

Definition at line 268 of file AMDTPowerProfileDataTypes.h.

#### 6.4.2.3 AMDTUInt32 m\_childCnt

Number of child counters

Definition at line 269 of file AMDTPowerProfileDataTypes.h.

#### 6.4.2.4 AMDTUInt32\* m\_pChildList

List of child counters

Definition at line 270 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.5 AMDTPwrCounterValue Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt32 [m\\_counterID](#)
- AMDTFloat32 [m\\_counterValue](#)

### 6.5.1 Detailed Description

Structure represents a counter ID and its value

Definition at line 215 of file AMDTPowerProfileDataTypes.h.

### 6.5.2 Field Documentation

#### 6.5.2.1 AMDTUInt32 m\_counterID

Counter index

Definition at line 217 of file AMDTPowerProfileDataTypes.h.

#### 6.5.2.2 AMDTFloat32 m\_counterValue

Counter value

Definition at line 218 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.6 AMDTPwrDevice Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- [AMDTPwrDeviceType m\\_type](#)
- [AMDTPwrDeviceId m\\_deviceID](#)
- [char \\* m\\_pName](#)
- [char \\* m\\_pDescription](#)
- [bool m\\_isAccessible](#)
- [AMDTPwrDevice \\* m\\_pFirstChild](#)
- [AMDTPwrDevice \\* m\\_pNextDevice](#)

### 6.6.1 Detailed Description

Following structure represents the device tree of the target system. Nodes will be available for components for which power counters are supported. Following are such components - AMD APUs and its subcomponents like CPU Compute-units, CPU Cores, integrated GPUs & AMD discrete GPUs.

Definition at line 181 of file AMDTPowerProfileDataTypes.h.

### 6.6.2 Field Documentation

#### 6.6.2.1 AMDTPwrDeviceType m\_type

Device type- compute unit/Core/ package/ dGPU

Definition at line 183 of file AMDTPowerProfileDataTypes.h.

#### 6.6.2.2 AMDTPwrDeviceId m\_deviceID

Device Id

Definition at line 184 of file AMDTPowerProfileDataTypes.h.

#### 6.6.2.3 char\* m\_pName

Name of the device

Definition at line 185 of file AMDTPowerProfileDataTypes.h.

#### 6.6.2.4 char\* m\_pDescription

Description about the device

Definition at line 186 of file AMDTPowerProfileDataTypes.h.



#### 6.6.2.5 bool m\_isAccessible

If counters are accessible

Definition at line 187 of file AMDTPowerProfileDataTypes.h.

#### 6.6.2.6 AMDTPwrDevice\* m\_pFirstChild

Points to the sub-devices of this device

Definition at line 188 of file AMDTPowerProfileDataTypes.h.

#### 6.6.2.7 AMDTPwrDevice\* m\_pNextDevice

Points to the next device at the same hierarchy

Definition at line 189 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.7 AMDTPwrHistogram Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt32 [m\\_counterId](#)
- AMDTUInt32 [m\\_numOfBins](#)
- AMDTFloat32 \* [m\\_pRange](#)
- AMDTFloat32 \* [m\\_pBins](#)

### 6.7.1 Detailed Description

Represents a generic histogram.

Definition at line 276 of file AMDTPowerProfileDataTypes.h.

### 6.7.2 Field Documentation

#### 6.7.2.1 AMDTUInt32 m\_counterId

Counter being aggregated

Definition at line 278 of file AMDTPowerProfileDataTypes.h.

#### 6.7.2.2 AMDTUInt32 m\_numOfBins

This is the number of histogram bins

Definition at line 279 of file AMDTPowerProfileDataTypes.h.

#### 6.7.2.3 AMDTFloat32\* m\_pRange

The ranges of the bins are stored in an array of  $n + 1$  elements pointed to by range

Definition at line 280 of file AMDTPowerProfileDataTypes.h.

#### 6.7.2.4 AMDTFloat32\* m\_pBins

The counts for each bin are stored in an array of  $n$  elements pointed to by bin

Definition at line 281 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.8 AMDTPwrInstrumentedPowerData Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt8 [m\\_name](#) [AMDT\_PWR\_MARKER\_BUFFER\_LENGTH]
- AMDTUInt8 [m\\_userBuffer](#) [AMDT\_PWR\_MARKER\_BUFFER\_LENGTH]
- [AMDTPwrSystemTime m\\_systemStartTime](#)
- AMDTUInt64 [m\\_startTs](#)
- AMDTUInt64 [m\\_endTs](#)
- [AMDTPwrProcessInfo m\\_pidInfo](#)

### 6.8.1 Detailed Description

Represents the instrumented power data.

Definition at line 301 of file AMDTPowerProfileDataTypes.h.

### 6.8.2 Field Documentation

#### 6.8.2.1 AMDTUInt8 m\_name[AMDT\_PWR\_MARKER\_BUFFER\_LENGTH]

Name of the user marker

Definition at line 303 of file AMDTPowerProfileDataTypes.h.

#### 6.8.2.2 AMDTUInt8 m\_userBuffer[AMDT\_PWR\_MARKER\_BUFFER\_LENGTH]

User supplied buffer

Definition at line 304 of file AMDTPowerProfileDataTypes.h.

#### 6.8.2.3 AMDTPwrSystemTime m\_systemStartTime

Profile start time

Definition at line 305 of file AMDTPowerProfileDataTypes.h.

#### 6.8.2.4 AMDTUInt64 m\_startTs

Marker start elapsed time

Definition at line 306 of file AMDTPowerProfileDataTypes.h.

#### 6.8.2.5 AMDTUInt64 m\_endTs

Marker end elapsed time

Definition at line 307 of file AMDTPowerProfileDataTypes.h.

#### 6.8.2.6 AMDTPwrProcessInfo m\_pidInfo

Process information

Definition at line 308 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.9 AMDTPwrProcessInfo Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt32 [m\\_pid](#)
- AMDTUInt32 [m\\_sampleCnt](#)
- AMDTFloat32 [m\\_power](#)
- AMDTFloat32 [m\\_ipc](#)
- char [m\\_name](#) [AMDT\_PWR\_EXE\_NAME\_LENGTH]
- char [m\\_path](#) [AMDT\_PWR\_EXE\_PATH\_LENGTH]

### 6.9.1 Detailed Description

Represents process power info.

Definition at line 287 of file AMDTPowerProfileDataTypes.h.

### 6.9.2 Field Documentation

#### 6.9.2.1 AMDTUInt32 m\_pid

Process id

Definition at line 289 of file AMDTPowerProfileDataTypes.h.

#### 6.9.2.2 AMDTUInt32 m\_sampleCnt

Number of PID samples

Definition at line 290 of file AMDTPowerProfileDataTypes.h.

#### 6.9.2.3 AMDTFloat32 m\_power

PID power indicator

Definition at line 291 of file AMDTPowerProfileDataTypes.h.

#### 6.9.2.4 AMDTFloat32 m\_ipc

Agreegated IPC value

Definition at line 292 of file AMDTPowerProfileDataTypes.h.

**6.9.2.5 char m\_name[AMDT\_PWR\_EXE\_NAME\_LENGTH]**

Executable name

Definition at line 293 of file AMDTPowerProfileDataTypes.h.

**6.9.2.6 char m\_path[AMDT\_PWR\_EXE\_PATH\_LENGTH]**

Path

Definition at line 294 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)

## 6.10 AMDTPwrSample Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- [AMDTPwrSystemTime m\\_systemTime](#)
- AMDTUInt64 [m\\_elapsedTimeMs](#)
- AMDTUInt64 [m\\_recordId](#)
- AMDTUInt32 [m\\_numOfValues](#)
- [AMDTPwrCounterValue](#) \* [m\\_counterValues](#)

### 6.10.1 Detailed Description

Output sample with timestamp and the counter values for all the enabled counters.

Definition at line 233 of file AMDTPowerProfileDataTypes.h.

### 6.10.2 Field Documentation

#### 6.10.2.1 AMDTPwrSystemTime m\_systemTime

Start time of Profiling

Definition at line 235 of file AMDTPowerProfileDataTypes.h.

#### 6.10.2.2 AMDTUInt64 m\_elapsedTimeMs

Elapsed time in milliseconds - relative to the start time of the profile

Definition at line 236 of file AMDTPowerProfileDataTypes.h.

#### 6.10.2.3 AMDTUInt64 m\_recordId

Record id

Definition at line 237 of file AMDTPowerProfileDataTypes.h.

#### 6.10.2.4 AMDTUInt32 m\_numOfValues

Number of counter values available

Definition at line 238 of file AMDTPowerProfileDataTypes.h.

#### 6.10.2.5 AMDTPwrCounterValue\* m\_counterValues

list of counter values

Definition at line 239 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)



## 6.11 AMDTPwrSystemTime Struct Reference

```
#include <AMDTPowerProfileDataTypes.h>
```

### Data Fields

- AMDTUInt64 [m\\_second](#)
- AMDTUInt64 [m\\_microSecond](#)

### 6.11.1 Detailed Description

This structure represents the system time in second and milliseconds

Definition at line 224 of file AMDTPowerProfileDataTypes.h.

### 6.11.2 Field Documentation

#### 6.11.2.1 AMDTUInt64 m\_second

Seconds

Definition at line 226 of file AMDTPowerProfileDataTypes.h.

#### 6.11.2.2 AMDTUInt64 m\_microSecond

Milliseconds

Definition at line 227 of file AMDTPowerProfileDataTypes.h.

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

- [AMDTPowerProfileDataTypes.h](#)



# Chapter 7

## File Documentation

### 7.1 AMDTDefinitions.h File Reference

Basic data type definitions and error codes used by the AMD CodeXL Power Profiler APIs. `#include <limits.h>`

#### Defines

- `#define AMDT_STATUS_OK AMDTResult(0)`
- `#define AMDT_ERROR_FAIL AMDTResult(0x80004005)`
- `#define AMDT_ERROR_INVALIDARG AMDTResult(0x80070057)`
- `#define AMDT_ERROR_OUTOFMEMORY AMDTResult(0x8007000E)`
- `#define AMDT_ERROR_UNEXPECTED AMDTResult(0x8000FFFF)`
- `#define AMDT_ERROR_ACCESSDENIED AMDTResult(0x80070005)`
- `#define AMDT_ERROR_HANDLE AMDTResult(0x80070006)`
- `#define AMDT_ERROR_ABORT AMDTResult(0x80004004)`
- `#define AMDT_ERROR_NOTIMPL AMDTResult(0x80004001)`
- `#define AMDT_ERROR_NOFILE AMDTResult(0x80070002)`
- `#define AMDT_ERROR_INVALIDPATH AMDTResult(0x80070003)`
- `#define AMDT_ERROR_INVALIDDATA AMDTResult(0x8007000D)`
- `#define AMDT_ERROR_NOTAVAILABLE AMDTResult(0x80075006)`
- `#define AMDT_ERROR_NODATA AMDTResult(0x800700E8)`
- `#define AMDT_ERROR_LOCKED AMDTResult(0x80070021)`
- `#define AMDT_ERROR_TIMEOUT AMDTResult(0x800705B4)`
- `#define AMDT_STATUS_PENDING AMDTResult(0x8000000A)`
- `#define AMDT_ERROR_NOTSUPPORTED AMDTResult(0x8000FFFE)`
- `#define AMDT_ERROR_DRIVER_ALREADY_INITIALIZED AMDTResult(0x80080001)`
- `#define AMDT_ERROR_DRIVER_UNAVAILABLE AMDTResult(0x80080002)`
- `#define AMDT_WARN_SMU_DISABLED AMDTResult(0x80080003)`

- #define [AMDT\\_WARN\\_IGPU\\_DISABLED](#) AMDTResult(0x80080004)
- #define [AMDT\\_ERROR\\_DRIVER\\_UNINITIALIZED](#) AMDTResult(0x80080005)
- #define [AMDT\\_ERROR\\_INVALID\\_DEVICEID](#) AMDTResult(0x80080006)
- #define [AMDT\\_ERROR\\_INVALID\\_COUNTERID](#) AMDTResult(0x80080007)
- #define [AMDT\\_ERROR\\_COUNTER\\_ALREADY\\_ENABLED](#) AMDTResult(0x80080008)
- #define [AMDT\\_ERROR\\_NO\\_WRITE\\_PERMISSION](#) AMDTResult(0x80080009)
- #define [AMDT\\_ERROR\\_COUNTER\\_NOT\\_ENABLED](#) AMDTResult(0x8008000A)
- #define [AMDT\\_ERROR\\_TIMER\\_NOT\\_SET](#) AMDTResult(0x8008000B)
- #define [AMDT\\_ERROR\\_PROFILE\\_DATAFILE\\_NOT\\_SET](#) AMDTResult(0x8008000C)
- #define [AMDT\\_ERROR\\_PROFILE\\_ALREADY\\_STARTED](#) AMDTResult(0x8008000D)
- #define [AMDT\\_ERROR\\_PROFILE\\_NOT\\_STARTED](#) AMDTResult(0x8008000E)
- #define [AMDT\\_ERROR\\_PROFILE\\_NOT\\_PAUSED](#) AMDTResult(0x8008000F)
- #define [AMDT\\_ERROR\\_PROFILE\\_DATA\\_NOT\\_AVAILABLE](#) AMDTResult(0x80080010)
- #define [AMDT\\_ERROR\\_PLATFORM\\_NOT\\_SUPPORTED](#) AMDTResult(0x80080011)
- #define [AMDT\\_ERROR\\_INTERNAL](#) AMDTResult(0x80080012)
- #define [AMDT\\_DRIVER\\_VERSION\\_MISMATCH](#) AMDTResult(0x80080013)
- #define [AMDT\\_ERROR\\_BIOS\\_VERSION\\_NOT\\_SUPPORTED](#) AMDTResult(0x80080014)
- #define [AMDT\\_ERROR\\_PROFILE\\_ALREADY\\_CONFIGURED](#) AMDTResult(0x80080015)
- #define [AMDT\\_ERROR\\_PROFILE\\_NOT\\_CONFIGURED](#) AMDTResult(0x80080016)
- #define [AMDT\\_ERROR\\_PROFILE\\_SESSION\\_EXISTS](#) AMDTResult(0x80080017)
- #define [AMDT\\_ERROR\\_SMU\\_ACCESS\\_FAILED](#) AMDTResult(0x80080018)
- #define [AMDT\\_ERROR\\_COUNTERS\\_NOT\\_ENABLED](#) AMDTResult(0x80080019)
- #define [AMDT\\_ERROR\\_PREVIOUS\\_SESSION\\_NOT\\_CLOSED](#) AMDTResult(0x80080020)
- #define [AMDT\\_ERROR\\_COUNTER\\_NOHIERARCHY](#) AMDTResult(0x80080021)
- #define [AMDT\\_ERROR\\_COUNTER\\_NOT\\_ACCESSIBLE](#) AMDTResult(0x80080022)
- #define [AMDT\\_ERROR\\_HYPERVISOR\\_NOT\\_SUPPORTED](#) AMDTResult(0x80080023)

- `#define` [AMDT\\_WARN\\_PROCESS\\_PROFILE\\_NOT\\_SUPPORTED](#) [AMDTResult\(0x80080024\)](#)
- `#define` [AMDT\\_ERROR\\_MARKER\\_NOT\\_SET](#) [AMDTResult\(0x80080025\)](#)

## Typedefs

- `typedef unsigned int` [AMDTResult](#)

### 7.1.1 Detailed Description

Basic data type definitions and error codes used by the AMD CodeXL Power Profiler APIs.

Definition in file [AMDTDefinitions.h](#).

### 7.1.2 Define Documentation

#### 7.1.2.1 `#define` [AMDT\\_STATUS\\_OK](#) [AMDTResult\(0\)](#)

Returned on success

Definition at line 76 of file [AMDTDefinitions.h](#).

#### 7.1.2.2 `#define` [AMDT\\_ERROR\\_FAIL](#) [AMDTResult\(0x80004005\)](#)

An internal error occurred.

Definition at line 80 of file [AMDTDefinitions.h](#).

#### 7.1.2.3 `#define` [AMDT\\_ERROR\\_INVALIDARG](#) [AMDTResult\(0x80070057\)](#)

Invalid argument is passed.

Definition at line 84 of file [AMDTDefinitions.h](#).

#### 7.1.2.4 `#define` [AMDT\\_ERROR\\_OUTOFMEMORY](#) [AMDTResult\(0x8007000E\)](#)

Memory allocation failed.

Definition at line 88 of file [AMDTDefinitions.h](#).

#### 7.1.2.5 `#define` [AMDT\\_ERROR\\_UNEXPECTED](#) [AMDTResult\(0x8000FFFF\)](#)

An unexpected error occurred.

Definition at line 92 of file [AMDTDefinitions.h](#).

**7.1.2.6 #define AMDT\_ERROR\_ACCESSDENIED AMDTResult(0x80070005)**

Profiler not available

Definition at line 96 of file AMDTDefinitions.h.

**7.1.2.7 #define AMDT\_ERROR\_HANDLE AMDTResult(0x80070006)**

Invalid handler is passed

Definition at line 100 of file AMDTDefinitions.h.

**7.1.2.8 #define AMDT\_ERROR\_ABORT AMDTResult(0x80004004)**

Profiler aborted due to an internal error

Definition at line 104 of file AMDTDefinitions.h.

**7.1.2.9 #define AMDT\_ERROR\_NOTIMPL AMDTResult(0x80004001)**

Requested profiler functionality is not yet implemented.

Definition at line 108 of file AMDTDefinitions.h.

**7.1.2.10 #define AMDT\_ERROR\_NOFILE AMDTResult(0x80070002)**

File not found.

Definition at line 112 of file AMDTDefinitions.h.

**7.1.2.11 #define AMDT\_ERROR\_INVALIDPATH AMDTResult(0x80070003)**

Invalid file path specified.

Definition at line 116 of file AMDTDefinitions.h.

**7.1.2.12 #define AMDT\_ERROR\_INVALIDDATA AMDTResult(0x8007000D)**

Invalid data is passed as a parameter.

Definition at line 120 of file AMDTDefinitions.h.

**7.1.2.13 #define AMDT\_ERROR\_NOTAVAILABLE AMDTResult(0x80075006)**

Requested functionality or data is not yet available.

Definition at line 124 of file AMDTDefinitions.h.

**7.1.2.14 #define AMDT\_ERROR\_NODATA AMDTResult(0x800700E8)**

No profile data is available.

Definition at line 128 of file AMDTDefinitions.h.

**7.1.2.15 #define AMDT\_ERROR\_LOCKED AMDTResult(0x80070021)**

Already locked.

Definition at line 132 of file AMDTDefinitions.h.

**7.1.2.16 #define AMDT\_ERROR\_TIMEOUT AMDTResult(0x800705B4)**

Timeout.

Definition at line 136 of file AMDTDefinitions.h.

**7.1.2.17 #define AMDT\_STATUS\_PENDING AMDTResult(0x8000000A)**

Profiler is currently active and the requested action is pending.

Definition at line 140 of file AMDTDefinitions.h.

**7.1.2.18 #define AMDT\_ERROR\_NOTSUPPORTED AMDTResult(0x8000FFFE)**

The requested functionality is not supported

Definition at line 144 of file AMDTDefinitions.h.

**7.1.2.19 #define AMDT\_ERROR\_DRIVER\_ALREADY\_INITIALIZED AMDTResult(0x80080001)**

Profiler is already initialized.

Definition at line 148 of file AMDTDefinitions.h.

**7.1.2.20 #define AMDT\_ERROR\_DRIVER\_UNAVAILABLE AMDTResult(0x80080002)**

Profile driver is not available.

Definition at line 152 of file AMDTDefinitions.h.

**7.1.2.21 #define AMDT\_WARN\_SMU\_DISABLED AMDTResult(0x80080003)**

SMU is disabled.

Definition at line 156 of file AMDTDefinitions.h.

**7.1.2.22 #define AMDT\_WARN\_IGPU\_DISABLED AMDTResult(0x80080004)**

Internal GPU is disabled.

Definition at line 160 of file AMDTDefinitions.h.

**7.1.2.23 #define AMDT\_ERROR\_DRIVER\_UNINITIALIZED AMDTResult(0x80080005)**

Driver is not yet initialized.

Definition at line 164 of file AMDTDefinitions.h.

**7.1.2.24 #define AMDT\_ERROR\_INVALID\_DEVICEID AMDTResult(0x80080006)**

Invalid device ID is passed as a parameter.

Definition at line 168 of file AMDTDefinitions.h.

**7.1.2.25 #define AMDT\_ERROR\_INVALID\_COUNTERID AMDTResult(0x80080007)**

Invalid profile counter id is passes as a parameter.

Definition at line 172 of file AMDTDefinitions.h.

**7.1.2.26 #define AMDT\_ERROR\_COUNTER\_ALREADY\_ENABLED AMDTResult(0x80080008)**

Specified counter ID is already enabled.

Definition at line 176 of file AMDTDefinitions.h.

**7.1.2.27 #define AMDT\_ERROR\_NO\_WRITE\_PERMISSION AMDTResult(0x80080009)**

No write permission to create the specified profile data file.

Definition at line 180 of file AMDTDefinitions.h.

**7.1.2.28 #define AMDT\_ERROR\_COUNTER\_NOT\_ENABLED AMDTResult(0x8008000A)**

Specified counter ID is not enabled.

Definition at line 184 of file AMDTDefinitions.h.



**7.1.2.29 #define AMDT\_ERROR\_TIMER\_NOT\_-  
SET AMDTResult(0x8008000B)**

Sampling timer is not set.

Definition at line 188 of file AMDTDefinitions.h.

**7.1.2.30 #define AMDT\_ERROR\_PROFILE\_DATAFILE\_NOT\_-  
SET AMDTResult(0x8008000C)**

Profile data file is not set.

Definition at line 192 of file AMDTDefinitions.h.

**7.1.2.31 #define AMDT\_ERROR\_PROFILE\_ALREADY\_-  
STARTED AMDTResult(0x8008000D)**

Profile was already started.

Definition at line 196 of file AMDTDefinitions.h.

**7.1.2.32 #define AMDT\_ERROR\_PROFILE\_NOT\_-  
STARTED AMDTResult(0x8008000E)**

Profile was not started.

Definition at line 200 of file AMDTDefinitions.h.

**7.1.2.33 #define AMDT\_ERROR\_PROFILE\_NOT\_-  
PAUSED AMDTResult(0x8008000F)**

Profile is not in paused state.

Definition at line 204 of file AMDTDefinitions.h.

**7.1.2.34 #define AMDT\_ERROR\_PROFILE\_DATA\_NOT\_-  
AVAILABLE AMDTResult(0x80080010)**

Profile data is not yet available.

Definition at line 208 of file AMDTDefinitions.h.

**7.1.2.35 #define AMDT\_ERROR\_PLATFORM\_NOT\_-  
SUPPORTED AMDTResult(0x80080011)**

This HW platform is not supported.

Definition at line 212 of file AMDTDefinitions.h.

**7.1.2.36 #define AMDT\_ERROR\_INTERNAL AMDTResult(0x80080012)**

An Internal error occurred.

Definition at line 216 of file AMDTDefinitions.h.

**7.1.2.37 #define AMDT\_DRIVER\_VERSION\_-  
MISMATCH AMDTResult(0x80080013)**

Mismatch between the expected and installed driver versions.

Definition at line 220 of file AMDTDefinitions.h.

**7.1.2.38 #define AMDT\_ERROR\_BIOS\_VERSION\_NOT\_-  
SUPPORTED AMDTResult(0x80080014)**

Bios needs to be upgraded in the system.

Definition at line 224 of file AMDTDefinitions.h.

**7.1.2.39 #define AMDT\_ERROR\_PROFILE\_ALREADY\_-  
CONFIGURED AMDTResult(0x80080015)**

Profile is already configured.

Definition at line 228 of file AMDTDefinitions.h.

**7.1.2.40 #define AMDT\_ERROR\_PROFILE\_NOT\_-  
CONFIGURED AMDTResult(0x80080016)**

Profile is not yet configured.

Definition at line 232 of file AMDTDefinitions.h.

**7.1.2.41 #define AMDT\_ERROR\_PROFILE\_SESSION\_-  
EXISTS AMDTResult(0x80080017)**

Profile session already exists.

Definition at line 236 of file AMDTDefinitions.h.

**7.1.2.42 #define AMDT\_ERROR\_SMU\_ACCESS\_-  
FAILED AMDTResult(0x80080018)**

Could not access the configured profile counter due to access failure.

Definition at line 240 of file AMDTDefinitions.h.

**7.1.2.43 #define AMDT\_ERROR\_COUNTERS\_NOT\_ENABLED AMDTResult(0x80080019)**

Could not start the profile session as counters are not enabled.

Definition at line 244 of file AMDTDefinitions.h.

**7.1.2.44 #define AMDT\_ERROR\_PREVIOUS\_SESSION\_NOT\_CLOSED AMDTResult(0x80080020)**

Previous profile session was not closed.

Definition at line 248 of file AMDTDefinitions.h.

**7.1.2.45 #define AMDT\_ERROR\_COUNTER\_NOHIERARCHY AMDTResult(0x80080021)**

Counter does not have any hierarchical relationship

Definition at line 252 of file AMDTDefinitions.h.

**7.1.2.46 #define AMDT\_ERROR\_COUNTER\_NOT\_ACCESSIBLE AMDTResult(0x80080022)**

Counter is not accessible

Definition at line 256 of file AMDTDefinitions.h.

**7.1.2.47 #define AMDT\_ERROR\_HYPERVISOR\_NOT\_SUPPORTED AMDTResult(0x80080023)**

Profiling not supported on Hypervisor

Definition at line 260 of file AMDTDefinitions.h.

**7.1.2.48 #define AMDT\_WARN\_PROCESS\_PROFILE\_NOT\_SUPPORTED AMDTResult(0x80080024)**

Process profiling not supported

Definition at line 264 of file AMDTDefinitions.h.

**7.1.2.49 #define AMDT\_ERROR\_MARKER\_NOT\_SET AMDTResult(0x80080025)**

Unable to configure the marker

Definition at line 268 of file AMDTDefinitions.h.

### 7.1.3 Typedef Documentation

#### 7.1.3.1 typedef unsigned int AMDTResult

Definition at line 72 of file AMDTDefinitions.h.

## 7.2 AMDTPowerProfileApi.h File Reference

AMD Power Profiler APIs to configure, control and collect the power profile counters.

```
#include <AMDTDefinitions.h>
```

```
#include <AMDTPowerProfileDataTypes.h>
```

### Functions

- [AMDTRResult AMDTPwrProfileInitialize](#) ([AMDTPwrProfileMode](#) profileMode)
- [AMDTRResult AMDTPwrGetSystemTopology](#) ([AMDTPwrDevice](#) \*\*ppTopology)
- [AMDTRResult AMDTPwrGetDeviceCounters](#) ([AMDTPwrDeviceId](#) deviceId, [AMDТУInt32](#) \*pNumCounters, [AMDTPwrCounterDesc](#) \*\*ppCounterDescs)
- [AMDTRResult AMDTPwrGetCounterDesc](#) ([AMDТУInt32](#) counterId, [AMDTPwrCounterDesc](#) \*pCounterDesc)
- [AMDTRResult AMDTPwrEnableCounter](#) ([AMDТУInt32](#) counterId)
- [AMDTRResult AMDTPwrDisableCounter](#) ([AMDТУInt32](#) counterId)
- [AMDTRResult AMDTPwrEnableAllCounters](#) ()
- [AMDTRResult AMDTPwrGetMinimalTimerSamplingPeriod](#) ([AMDТУInt32](#) \*pIntervalMilliSec)
- [AMDTRResult AMDTPwrSetTimerSamplingPeriod](#) ([AMDТУInt32](#) interval)
- [AMDTRResult AMDTPwrStartProfiling](#) ()
- [AMDTRResult AMDTPwrStopProfiling](#) ()
- [AMDTRResult AMDTPwrPauseProfiling](#) ()
- [AMDTRResult AMDTPwrResumeProfiling](#) ()
- [AMDTRResult AMDTPwrGetProfilingState](#) ([AMDTPwrProfileState](#) \*pState)
- [AMDTRResult AMDTPwrProfileClose](#) ()
- [AMDTRResult AMDTPwrSetSampleValueOption](#) ([AMDTSampleValueOption](#) opt)
- [AMDTRResult AMDTPwrGetSampleValueOption](#) ([AMDTSampleValueOption](#) \*pOpt)
- [AMDTRResult AMDTPwrReadAllEnabledCounters](#) ([AMDТУInt32](#) \*pNumOfSamples, [AMDTPwrSample](#) \*\*ppData)
- [AMDTRResult AMDTPwrReadCounterHistogram](#) ([AMDТУInt32](#) counterId, [AMDТУInt32](#) \*pNumEntries, [AMDTPwrHistogram](#) \*\*ppData)
- [AMDTRResult AMDTPwrReadCumulativeCounter](#) ([AMDТУInt32](#) counterId, [AMDТУInt32](#) \*pNumEntries, [AMDТFloat32](#) \*\*ppData)
- [AMDTRResult AMDTPwrGetTimerSamplingPeriod](#) ([AMDТУInt32](#) \*pIntervalMilliSec)
- [AMDTRResult AMDTPwrIsCounterEnabled](#) ([AMDТУInt32](#) counterId)
- [AMDTRResult AMDTPwrGetNumEnabledCounters](#) ([AMDТУInt32](#) \*pCount)
- [AMDTRResult AMDTPwrGetApuPstateInfo](#) ([AMDTPwrApuPstateList](#) \*pList)
- [AMDTRResult AMDTPwrGetCounterHierarchy](#) ([AMDТУInt32](#) counterId, [AMDTPwrCounterHierarchy](#) \*pInfo)
- [AMDTRResult AMDTPwrGetNodeTemperature](#) ([AMDТFloat32](#) \*pNodeTemp)

- [AMDTReturn AMDTEnableProcessProfiling](#) (void)
- [AMDTReturn AMDTReadProcessProfileData](#) (AMDTUInt32 \*pPIDCount, AMDTPwrProcessInfo \*\*ppData)

### 7.2.1 Detailed Description

AMD Power Profiler APIs to configure, control and collect the power profile counters.

Definition in file [AMDTPowerProfileApi.h](#).

## 7.3 AMDTPowerProfileDataTypes.h File Reference

Data types and structure definitions used by CodeXL Power Profiler APIs. #include <AMDTDefinitions.h>

### Data Structures

- struct [AMDTPwrDevice](#)
- struct [AMDTPwrCounterDesc](#)
- struct [AMDTPwrCounterValue](#)
- struct [AMDTPwrSystemTime](#)
- struct [AMDTPwrSample](#)
- struct [AMDTPwrApuPstate](#)
- struct [AMDTPwrApuPstateList](#)
- struct [AMDTPwrCounterHierarchy](#)
- struct [AMDTPwrHistogram](#)
- struct [AMDTPwrProcessInfo](#)
- struct [AMDTPwrInstrumentedPowerData](#)

### Defines

- #define [AMDT\\_PWR\\_ALL\\_DEVICES](#) 0xFFFFFFFFFUL
- #define [AMDT\\_PWR\\_ALL\\_COUNTERS](#) 0xFFFFFFFFFUL
- #define [AMDT\\_PWR\\_EXE\\_NAME\\_LENGTH](#) 64
- #define [AMDT\\_PWR\\_EXE\\_PATH\\_LENGTH](#) 256
- #define [AMDT\\_MAX\\_PSTATES](#) 8
- #define [AMDT\\_PWR\\_MARKER\\_BUFFER\\_LENGTH](#) 32

### Typedefs

- typedef AMDTUInt32 [AMDTPwrDeviceId](#)

### Enumerations

- enum [AMDTPwrProfileMode](#) { [AMDT\\_PWR\\_PROFILE\\_MODE\\_ONLINE](#), [AMDT\\_PWR\\_PROFILE\\_MODE\\_OFFLINE](#) }
- enum [AMDTPwrDeviceType](#) { [AMDT\\_PWR\\_DEVICE\\_SYSTEM](#), [AMDT\\_PWR\\_DEVICE\\_PACKAGE](#), [AMDT\\_PWR\\_DEVICE\\_CPU\\_COMPUTE\\_UNIT](#), [AMDT\\_PWR\\_DEVICE\\_CPU\\_CORE](#), [AMDT\\_PWR\\_DEVICE\\_INTERNAL\\_GPU](#), [AMDT\\_PWR\\_DEVICE\\_EXTERNAL\\_GPU](#), [AMDT\\_PWR\\_DEVICE\\_SVI2](#), [AMDT\\_PWR\\_DEVICE\\_CNT](#) }

- enum `AMDTPwrCategory` {  
`AMDT_PWR_CATEGORY_POWER,`            `AMDT_PWR_CATEGORY_-`  
`FREQUENCY,`   `AMDT_PWR_CATEGORY_TEMPERATURE,`   `AMDT_-`  
`PWR_CATEGORY_VOLTAGE,`  
  
`AMDT_PWR_CATEGORY_CURRENT,` `AMDT_PWR_CATEGORY_DVFS,`  
`AMDT_PWR_CATEGORY_PROCESS,` `AMDT_PWR_CATEGORY_TIME,`  
  
`AMDT_PWR_CATEGORY_COUNT,` `AMDT_PWR_CATEGORY_CNT` }
- enum `AMDTPwrAggregation` { `AMDT_PWR_VALUE_SINGLE,` `AMDT_-`  
`PWR_VALUE_CUMULATIVE,`            `AMDT_PWR_VALUE_HISTOGRAM,`  
`AMDT_PWR_VALUE_CNT` }
- enum `AMDTPwrUnit` {  
`AMDT_PWR_UNIT_TYPE_COUNT,`            `AMDT_PWR_UNIT_TYPE_-`  
`PERCENT,`   `AMDT_PWR_UNIT_TYPE_RATIO,`   `AMDT_PWR_UNIT_-`  
`TYPE_MILLI_SECOND,`  
  
`AMDT_PWR_UNIT_TYPE_JOULE,`    `AMDT_PWR_UNIT_TYPE_WATT,`  
`AMDT_PWR_UNIT_TYPE_VOLT,`    `AMDT_PWR_UNIT_TYPE_MILLI_-`  
`AMPERE,`  
  
`AMDT_PWR_UNIT_TYPE_MEGA_HERTZ,` `AMDT_PWR_UNIT_TYPE_-`  
`CENTIGRADE,` `AMDT_PWR_UNIT_TYPE_CNT` }
- enum `AMDTPwrProfileState` {  
`AMDT_PWR_PROFILE_STATE_UNINITIALIZED,`            `AMDT_PWR_-`  
`PROFILE_STATE_IDLE,`            `AMDT_PWR_PROFILE_STATE_RUNNING,`  
`AMDT_PWR_PROFILE_STATE_PAUSED,`  
  
`AMDT_PWR_PROFILE_STATE_STOPPED,`            `AMDT_PWR_PROFILE_-`  
`STATE_ABORTED,` `AMDT_PWR_PROFILE_STATE_CNT` }
- enum `AMDTSampleValueOption` { `AMDT_PWR_SAMPLE_VALUE_-`  
`INSTANTANEOUS,` `AMDT_PWR_SAMPLE_VALUE_LIST,` `AMDT_PWR_-`  
`SAMPLE_VALUE_AVERAGE,`            `AMDT_PWR_SAMPLE_VALUE_CNT`  
}
- enum `AMDTApuPStates` {  
`AMDT_PWR_PSTATE_PB0,` `AMDT_PWR_PSTATE_PB1,` `AMDT_PWR_-`  
`PSTATE_PB2,` `AMDT_PWR_PSTATE_PB3,`  
  
`AMDT_PWR_PSTATE_PB4,` `AMDT_PWR_PSTATE_PB5,` `AMDT_PWR_-`  
`PSTATE_PB6,` `AMDT_PWR_PSTATE_P0,`  
  
`AMDT_PWR_PSTATE_P1,`   `AMDT_PWR_PSTATE_P2,`   `AMDT_PWR_-`  
`PSTATE_P3,` `AMDT_PWR_PSTATE_P4,`  
  
`AMDT_PWR_PSTATE_P5,`   `AMDT_PWR_PSTATE_P6,`   `AMDT_PWR_-`  
`PSTATE_P7` }

### 7.3.1 Detailed Description

Data types and structure definitions used by CodeXL Power Profiler APIs.

Definition in file [AMDTPowerProfileDataTypes.h](#).



### 7.3.2 Define Documentation

#### 7.3.2.1 **#define AMDT\_PWR\_ALL\_DEVICES 0xFFFFFFFFFUL**

HW Components for which power counters are supported are called devices. Following are such components:

- AMD APU's and its subcomponents like CPU Compute-units, CPU Cores, integrated GPU's
- AMD discrete GPU's This macro denotes all the devices that are relevant to power profiling.

Definition at line 28 of file AMDTPowerProfileDataTypes.h.

#### 7.3.2.2 **#define AMDT\_PWR\_ALL\_COUNTERS 0xFFFFFFFFFUL**

This macro denotes all the counters that are relevant to power profiling.

Definition at line 33 of file AMDTPowerProfileDataTypes.h.

#### 7.3.2.3 **#define AMDT\_PWR\_EXE\_NAME\_LENGTH 64**

Process name length

Definition at line 37 of file AMDTPowerProfileDataTypes.h.

#### 7.3.2.4 **#define AMDT\_PWR\_EXE\_PATH\_LENGTH 256**

Process name length

Definition at line 41 of file AMDTPowerProfileDataTypes.h.

#### 7.3.2.5 **#define AMDT\_MAX\_PSTATES 8**

Maximum number of available APU P-States

Definition at line 45 of file AMDTPowerProfileDataTypes.h.

#### 7.3.2.6 **#define AMDT\_PWR\_MARKER\_BUFFER\_LENGTH 32**

Process marker buffer length

Definition at line 49 of file AMDTPowerProfileDataTypes.h.

### 7.3.3 Typedef Documentation

#### 7.3.3.1 typedef AMDTUInt32 AMDTPwrDeviceId

Device Id

Definition at line 53 of file AMDTPowerProfileDataTypes.h.

## **Chapter 8**

# **Example Documentation**

### **8.1 CollectAllCounters.cpp**

Example program to collect all the available counters.

# Index

AMDT_PWR_CATEGORY_CNT <a href="#">profiling, 13</a>	AMDT_PWR_PROFILE_MODE_- OFFLINE <a href="#">profiling, 12</a>
AMDT_PWR_CATEGORY_COUNT <a href="#">profiling, 13</a>	AMDT_PWR_PROFILE_MODE_- ONLINE <a href="#">profiling, 12</a>
AMDT_PWR_CATEGORY_CURRENT <a href="#">profiling, 13</a>	AMDT_PWR_PROFILE_STATE_- ABORTED <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_DVFS <a href="#">profiling, 13</a>	AMDT_PWR_PROFILE_STATE_CNT <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_- FREQUENCY <a href="#">profiling, 12</a>	AMDT_PWR_PROFILE_STATE_IDLE <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_POWER <a href="#">profiling, 12</a>	AMDT_PWR_PROFILE_STATE_- PAUSED <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_PROCESS <a href="#">profiling, 13</a>	AMDT_PWR_PROFILE_STATE_- RUNNING <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_- TEMPERATURE <a href="#">profiling, 12</a>	AMDT_PWR_PROFILE_STATE_- STOPPED <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_TIME <a href="#">profiling, 13</a>	AMDT_PWR_PROFILE_STATE_- UNINITIALIZED <a href="#">profiling, 14</a>
AMDT_PWR_CATEGORY_VOLTAGE <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P0 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_CNT <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P1 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_CPU_- COMPUTE_UNIT <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P2 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_CPU_CORE <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P3 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_EXTERNAL_- GPU <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P4 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_INTERNAL_- GPU <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P5 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_PACKAGE <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P6 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_SVI2 <a href="#">profiling, 12</a>	AMDT_PWR_PSTATE_P7 <a href="#">profiling, 15</a>
AMDT_PWR_DEVICE_SYSTEM <a href="#">profiling, 12</a>	

- AMDT\_PWR\_PSTATE\_PB0
  - profiling, [14](#)
- AMDT\_PWR\_PSTATE\_PB1
  - profiling, [14](#)
- AMDT\_PWR\_PSTATE\_PB2
  - profiling, [14](#)
- AMDT\_PWR\_PSTATE\_PB3
  - profiling, [14](#)
- AMDT\_PWR\_PSTATE\_PB4
  - profiling, [14](#)
- AMDT\_PWR\_PSTATE\_PB5
  - profiling, [14](#)
- AMDT\_PWR\_PSTATE\_PB6
  - profiling, [15](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_-  
AVERAGE
  - profiling, [14](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_CNT
  - profiling, [14](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_-  
INSTANTANEOUS
  - profiling, [14](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_LIST
  - profiling, [14](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
CENTIGRADE
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_CNT
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_COUNT
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_JOULE
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_MEGA\_-  
HERTZ
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_MILLI\_-  
AMPERE
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_MILLI\_-  
SECOND
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_PERCENT
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_RATIO
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_VOLT
  - profiling, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_WATT
  - profiling, [13](#)
- AMDT\_PWR\_VALUE\_CNT
  - profiling, [13](#)
- AMDT\_PWR\_VALUE\_CUMULATIVE
  - profiling, [13](#)
- AMDT\_PWR\_VALUE\_HISTOGRAM
  - profiling, [13](#)
- AMDT\_PWR\_VALUE\_SINGLE
  - profiling, [13](#)
- AMDT\_DRIVER\_VERSION\_-  
MISMATCH
  - AMDTDefinitions.h, [56](#)
- AMDT\_ERROR\_ABORT
  - AMDTDefinitions.h, [52](#)
- AMDT\_ERROR\_ACCESSDENIED
  - AMDTDefinitions.h, [51](#)
- AMDT\_ERROR\_BIOS\_VERSION\_-  
NOT\_SUPPORTED
  - AMDTDefinitions.h, [56](#)
- AMDT\_ERROR\_COUNTER\_-  
ALREADY\_ENABLED
  - AMDTDefinitions.h, [54](#)
- AMDT\_ERROR\_COUNTER\_-  
NOHIERARCHY
  - AMDTDefinitions.h, [57](#)
- AMDT\_ERROR\_COUNTER\_NOT\_-  
ACCESSIBLE
  - AMDTDefinitions.h, [57](#)
- AMDT\_ERROR\_COUNTER\_NOT\_-  
ENABLED
  - AMDTDefinitions.h, [54](#)
- AMDT\_ERROR\_COUNTERS\_NOT\_-  
ENABLED
  - AMDTDefinitions.h, [56](#)
- AMDT\_ERROR\_DRIVER\_-  
ALREADY\_INITIALIZED
  - AMDTDefinitions.h, [53](#)
- AMDT\_ERROR\_DRIVER\_-  
UNAVAILABLE
  - AMDTDefinitions.h, [53](#)
- AMDT\_ERROR\_DRIVER\_-  
UNINITIALIZED
  - AMDTDefinitions.h, [54](#)
- AMDT\_ERROR\_FAIL
  - AMDTDefinitions.h, [51](#)
- AMDT\_ERROR\_HANDLE
  - AMDTDefinitions.h, [52](#)
- AMDT\_ERROR\_HYPERVISOR\_-  
NOT\_SUPPORTED
  - AMDTDefinitions.h, [57](#)
- AMDT\_ERROR\_INTERNAL

- AMDTDefinitions.h, 55
- AMDT\_ERROR\_INVALID\_-  
COUNTERID  
AMDTDefinitions.h, 54
- AMDT\_ERROR\_INVALID\_DEVICEID  
AMDTDefinitions.h, 54
- AMDT\_ERROR\_INVALIDARG  
AMDTDefinitions.h, 51
- AMDT\_ERROR\_INVALIDDATA  
AMDTDefinitions.h, 52
- AMDT\_ERROR\_INVALIDPATH  
AMDTDefinitions.h, 52
- AMDT\_ERROR\_LOCKED  
AMDTDefinitions.h, 53
- AMDT\_ERROR\_MARKER\_NOT\_SET  
AMDTDefinitions.h, 57
- AMDT\_ERROR\_NO\_WRITE\_-  
PERMISSION  
AMDTDefinitions.h, 54
- AMDT\_ERROR\_NODATA  
AMDTDefinitions.h, 52
- AMDT\_ERROR\_NOFILE  
AMDTDefinitions.h, 52
- AMDT\_ERROR\_NOTAVAILABLE  
AMDTDefinitions.h, 52
- AMDT\_ERROR\_NOTIMPL  
AMDTDefinitions.h, 52
- AMDT\_ERROR\_NOTSUPPORTED  
AMDTDefinitions.h, 53
- AMDT\_ERROR\_OUTOFMEMORY  
AMDTDefinitions.h, 51
- AMDT\_ERROR\_PLATFORM\_NOT\_-  
SUPPORTED  
AMDTDefinitions.h, 55
- AMDT\_ERROR\_PREVIOUS\_-  
SESSION\_NOT\_CLOSED  
AMDTDefinitions.h, 57
- AMDT\_ERROR\_PROFILE\_-  
ALREADY\_CONFIGURED  
AMDTDefinitions.h, 56
- AMDT\_ERROR\_PROFILE\_-  
ALREADY\_STARTED  
AMDTDefinitions.h, 55
- AMDT\_ERROR\_PROFILE\_DATA\_-  
NOT\_AVAILABLE  
AMDTDefinitions.h, 55
- AMDT\_ERROR\_PROFILE\_-  
DATAFILE\_NOT\_SET  
AMDTDefinitions.h, 55
- AMDT\_ERROR\_PROFILE\_NOT\_-  
CONFIGURED  
AMDTDefinitions.h, 56
- AMDT\_ERROR\_PROFILE\_NOT\_-  
PAUSED  
AMDTDefinitions.h, 55
- AMDT\_ERROR\_PROFILE\_NOT\_-  
STARTED  
AMDTDefinitions.h, 55
- AMDT\_ERROR\_PROFILE\_SESSION\_-  
EXISTS  
AMDTDefinitions.h, 56
- AMDT\_ERROR\_SMU\_ACCESS\_-  
FAILED  
AMDTDefinitions.h, 56
- AMDT\_ERROR\_TIMEOUT  
AMDTDefinitions.h, 53
- AMDT\_ERROR\_TIMER\_NOT\_SET  
AMDTDefinitions.h, 54
- AMDT\_ERROR\_UNEXPECTED  
AMDTDefinitions.h, 51
- AMDT\_MAX\_PSTATES  
AMDTPowerProfileDataTypes.h, 63
- AMDT\_PWR\_ALL\_COUNTERS  
AMDTPowerProfileDataTypes.h, 63
- AMDT\_PWR\_ALL\_DEVICES  
AMDTPowerProfileDataTypes.h, 63
- AMDT\_PWR\_EXE\_NAME\_LENGTH  
AMDTPowerProfileDataTypes.h, 63
- AMDT\_PWR\_EXE\_PATH\_LENGTH  
AMDTPowerProfileDataTypes.h, 63
- AMDT\_PWR\_MARKER\_BUFFER\_-  
LENGTH  
AMDTPowerProfileDataTypes.h, 63
- AMDT\_STATUS\_OK  
AMDTDefinitions.h, 51
- AMDT\_STATUS\_PENDING  
AMDTDefinitions.h, 53
- AMDT\_WARN\_IGPU\_DISABLED  
AMDTDefinitions.h, 54
- AMDT\_WARN\_PROCESS\_PROFILE\_-  
NOT\_SUPPORTED  
AMDTDefinitions.h, 57
- AMDT\_WARN\_SMU\_DISABLED  
AMDTDefinitions.h, 53
- AMDTApuPStates  
profiling, 14
- AMDTDefinitions.h, 49
- AMDT\_DRIVER\_VERSION\_-  
MISMATCH, 56

- AMDT\_ERROR\_ABORT, [52](#)
- AMDT\_ERROR\_-
  - ACCESSDENIED, [51](#)
- AMDT\_ERROR\_BIOS\_-
  - VERSION\_NOT\_-  
SUPPORTED, [56](#)
- AMDT\_ERROR\_COUNTER\_-
  - ALREADY\_ENABLED, [54](#)
- AMDT\_ERROR\_COUNTER\_-
  - NOHIERARCHY, [57](#)
- AMDT\_ERROR\_COUNTER\_-
  - NOT\_ACCESSIBLE, [57](#)
- AMDT\_ERROR\_COUNTER\_-
  - NOT\_ENABLED, [54](#)
- AMDT\_ERROR\_COUNTERS\_-
  - NOT\_ENABLED, [56](#)
- AMDT\_ERROR\_DRIVER\_-
  - ALREADY\_INITIALIZED,  
[53](#)
- AMDT\_ERROR\_DRIVER\_-
  - UNAVAILABLE, [53](#)
- AMDT\_ERROR\_DRIVER\_-
  - UNINITIALIZED, [54](#)
- AMDT\_ERROR\_FAIL, [51](#)
- AMDT\_ERROR\_HANDLE, [52](#)
- AMDT\_ERROR\_HYPERVISOR\_-
  - NOT\_SUPPORTED, [57](#)
- AMDT\_ERROR\_INTERNAL, [55](#)
- AMDT\_ERROR\_INVALID\_-
  - COUNTERID, [54](#)
- AMDT\_ERROR\_INVALID\_-
  - DEVICEID, [54](#)
- AMDT\_ERROR\_INVALIDARG,  
[51](#)
- AMDT\_ERROR\_INVALIDDATA,  
[52](#)
- AMDT\_ERROR\_INVALIDPATH,  
[52](#)
- AMDT\_ERROR\_LOCKED, [53](#)
- AMDT\_ERROR\_MARKER\_-
  - NOT\_SET, [57](#)
- AMDT\_ERROR\_NO\_WRITE\_-
  - PERMISSION, [54](#)
- AMDT\_ERROR\_NODATA, [52](#)
- AMDT\_ERROR\_NOFILE, [52](#)
- AMDT\_ERROR\_-
  - NOTAVAILABLE, [52](#)
- AMDT\_ERROR\_NOTIMPL, [52](#)
- AMDT\_ERROR\_-
  - NOTSUPPORTED, [53](#)
- AMDT\_ERROR\_-
  - OUTOFMEMORY, [51](#)
- AMDT\_ERROR\_PLATFORM\_-
  - NOT\_SUPPORTED, [55](#)
- AMDT\_ERROR\_PREVIOUS\_-
  - SESSION\_NOT\_CLOSED,  
[57](#)
- AMDT\_ERROR\_PROFILE\_-
  - ALREADY\_CONFIGURED,  
[56](#)
- AMDT\_ERROR\_PROFILE\_-
  - ALREADY\_STARTED, [55](#)
- AMDT\_ERROR\_PROFILE\_-
  - DATA\_NOT\_AVAILABLE,  
[55](#)
- AMDT\_ERROR\_PROFILE\_-
  - DATAFILE\_NOT\_SET, [55](#)
- AMDT\_ERROR\_PROFILE\_NOT\_-
  - CONFIGURED, [56](#)
- AMDT\_ERROR\_PROFILE\_NOT\_-
  - PAUSED, [55](#)
- AMDT\_ERROR\_PROFILE\_NOT\_-
  - STARTED, [55](#)
- AMDT\_ERROR\_PROFILE\_-
  - SESSION\_EXISTS, [56](#)
- AMDT\_ERROR\_SMU\_ACCESS\_-
  - FAILED, [56](#)
- AMDT\_ERROR\_TIMEOUT, [53](#)
- AMDT\_ERROR\_TIMER\_NOT\_-
  - SET, [54](#)
- AMDT\_ERROR\_UNEXPECTED,  
[51](#)
- AMDT\_STATUS\_OK, [51](#)
- AMDT\_STATUS\_PENDING, [53](#)
- AMDT\_WARN\_IGPU\_-
  - DISABLED, [54](#)
- AMDT\_WARN\_PROCESS\_-
  - PROFILE\_NOT\_-  
SUPPORTED, [57](#)
- AMDT\_WARN\_SMU\_DISABLED,  
[53](#)
- AMDTRResult, [58](#)
- AMDDeviceType
  - profiling, [12](#)
- AMDTEnableProcessProfiling
  - profiling, [28](#)
- AMDTPowerProfileApi.h, [59](#)
- AMDTPowerProfileDataTypes.h, [61](#)
- AMDT\_MAX\_PSTATES, [63](#)

- AMDT\_PWR\_ALL\_COUNTERS, 63
- AMDT\_PWR\_ALL\_DEVICES, 63
- AMDT\_PWR\_EXE\_NAME\_-LENGTH, 63
- AMDT\_PWR\_EXE\_PATH\_-LENGTH, 63
- AMDT\_PWR\_MARKER\_-BUFFER\_LENGTH, 63
- AMDTPwrDeviceId, 64
- AMDTPwrAggregation
  - profiling, 13
- AMDTPwrApuPstate, 31
  - m\_frequency, 31
  - m\_isBoosted, 31
  - m\_state, 31
- AMDTPwrApuPstateList, 33
  - m\_cnt, 33
  - m\_stateInfo, 33
- AMDTPwrCategory
  - profiling, 12
- AMDTPwrCounterDesc, 34
  - m\_aggregation, 35
  - m\_category, 35
  - m\_counterID, 34
  - m\_description, 35
  - m\_deviceId, 34
  - m\_maxValue, 35
  - m\_minValue, 35
  - m\_name, 34
  - m\_units, 35
- AMDTPwrCounterHierarchy, 36
  - m\_childCnt, 36
  - m\_counter, 36
  - m\_parent, 36
  - m\_pChildList, 36
- AMDTPwrCounterValue, 37
  - m\_counterID, 37
  - m\_counterValue, 37
- AMDTPwrDevice, 38
  - m\_deviceID, 38
  - m\_isAccessible, 38
  - m\_pDescription, 38
  - m\_pFirstChild, 39
  - m\_pName, 38
  - m\_pNextDevice, 39
  - m\_type, 38
- AMDTPwrDeviceId
  - AMDTPowerProfileDataTypes.h, 64
- AMDTPwrDisableCounter
  - profiling, 18
- AMDTPwrEnableAllCounters
  - profiling, 19
- AMDTPwrEnableCounter
  - profiling, 17
- AMDTPwrGetApuPstateInfo
  - profiling, 27
- AMDTPwrGetCounterDesc
  - profiling, 17
- AMDTPwrGetCounterHierarchy
  - profiling, 27
- AMDTPwrGetDeviceCounters
  - profiling, 16
- AMDTPwrGetMinimalTimerSamplingPeriod
  - profiling, 19
- AMDTPwrGetNodeTemperature
  - profiling, 28
- AMDTPwrGetNumEnabledCounters
  - profiling, 26
- AMDTPwrGetProfilingState
  - profiling, 22
- AMDTPwrGetSampleValueOption
  - profiling, 23
- AMDTPwrGetSystemTopology
  - profiling, 15
- AMDTPwrGetTimerSamplingPeriod
  - profiling, 26
- AMDTPwrHistogram, 40
  - m\_counterId, 40
  - m\_numOfBins, 40
  - m\_pBins, 40
  - m\_pRange, 40
- AMDTPwrInstrumentedPowerData, 41
  - m\_endTs, 41
  - m\_name, 41
  - m\_pidInfo, 42
  - m\_startTs, 41
  - m\_systemStartTime, 41
  - m\_userBuffer, 41
- AMDTPwrIsCounterEnabled
  - profiling, 26
- AMDTPwrPauseProfiling
  - profiling, 21
- AMDTPwrProcessInfo, 43
  - m\_ipc, 43
  - m\_name, 43
  - m\_path, 44
  - m\_pid, 43
  - m\_power, 43
  - m\_sampleCnt, 43



- AMDTPwrProfileClose
  - profiling, [22](#)
- AMDTPwrProfileInitialize
  - profiling, [15](#)
- AMDTPwrProfileMode
  - profiling, [12](#)
- AMDTPwrProfileState
  - profiling, [13](#)
- AMDTPwrReadAllEnabledCounters
  - profiling, [23](#)
- AMDTPwrReadCounterHistogram
  - profiling, [24](#)
- AMDTPwrReadCumulativeCounter
  - profiling, [25](#)
- AMDTPwrResumeProfiling
  - profiling, [21](#)
- AMDTPwrSample, [45](#)
  - m\_counterValues, [45](#)
  - m\_elapsedTimeMs, [45](#)
  - m\_numOfValues, [45](#)
  - m\_recordId, [45](#)
  - m\_systemTime, [45](#)
- AMDTPwrSetSampleValueOption
  - profiling, [22](#)
- AMDTPwrSetTimerSamplingPeriod
  - profiling, [20](#)
- AMDTPwrStartProfiling
  - profiling, [20](#)
- AMDTPwrStopProfiling
  - profiling, [21](#)
- AMDTPwrSystemTime, [47](#)
  - m\_microSecond, [47](#)
  - m\_second, [47](#)
- AMDTPwrUnit
  - profiling, [13](#)
- AMDTPwrReadProcessProfileData
  - profiling, [29](#)
- AMDTPwrResult
  - AMDTDefinitions.h, [58](#)
- AMDTPwrSampleValueOption
  - profiling, [14](#)
- m\_aggregation
  - AMDTPwrCounterDesc, [35](#)
- m\_category
  - AMDTPwrCounterDesc, [35](#)
- m\_childCnt
  - AMDTPwrCounterHierarchy, [36](#)
- m\_cnt
  - AMDTPwrApuPstateList, [33](#)
- m\_counter
  - AMDTPwrCounterHierarchy, [36](#)
- m\_counterID
  - AMDTPwrCounterDesc, [34](#)
  - AMDTPwrCounterValue, [37](#)
- m\_counterId
  - AMDTPwrHistogram, [40](#)
- m\_counterValue
  - AMDTPwrCounterValue, [37](#)
- m\_counterValues
  - AMDTPwrSample, [45](#)
- m\_description
  - AMDTPwrCounterDesc, [35](#)
- m\_deviceID
  - AMDTPwrDevice, [38](#)
- m\_deviceId
  - AMDTPwrCounterDesc, [34](#)
- m\_elapsedTimeMs
  - AMDTPwrSample, [45](#)
- m\_endTs
  - AMDTPwrInstrumentedPowerData, [41](#)
- m\_frequency
  - AMDTPwrApuPstate, [31](#)
- m\_ipc
  - AMDTPwrProcessInfo, [43](#)
- m\_isAccessible
  - AMDTPwrDevice, [38](#)
- m\_isBoosted
  - AMDTPwrApuPstate, [31](#)
- m\_maxValue
  - AMDTPwrCounterDesc, [35](#)
- m\_microSecond
  - AMDTPwrSystemTime, [47](#)
- m\_minValue
  - AMDTPwrCounterDesc, [35](#)
- m\_name
  - AMDTPwrCounterDesc, [34](#)
  - AMDTPwrInstrumentedPowerData, [41](#)
  - AMDTPwrProcessInfo, [43](#)
- m\_numOfBins
  - AMDTPwrHistogram, [40](#)
- m\_numOfValues
  - AMDTPwrSample, [45](#)
- m\_parent
  - AMDTPwrCounterHierarchy, [36](#)
- m\_path
  - AMDTPwrProcessInfo, [44](#)
- m\_pBins

- AMDTPwrHistogram, [40](#)
- m\_pChildList
  - AMDTPwrCounterHierarchy, [36](#)
- m\_pDescription
  - AMDTPwrDevice, [38](#)
- m\_pFirstChild
  - AMDTPwrDevice, [39](#)
- m\_pid
  - AMDTPwrProcessInfo, [43](#)
- m\_pidInfo
  - AMDTPwrInstrumentedPowerData, [42](#)
- m\_pName
  - AMDTPwrDevice, [38](#)
- m\_pNextDevice
  - AMDTPwrDevice, [39](#)
- m\_power
  - AMDTPwrProcessInfo, [43](#)
- m\_pRange
  - AMDTPwrHistogram, [40](#)
- m\_recordId
  - AMDTPwrSample, [45](#)
- m\_sampleCnt
  - AMDTPwrProcessInfo, [43](#)
- m\_second
  - AMDTPwrSystemTime, [47](#)
- m\_startTs
  - AMDTPwrInstrumentedPowerData, [41](#)
- m\_state
  - AMDTPwrApuPstate, [31](#)
- m\_stateInfo
  - AMDTPwrApuPstateList, [33](#)
- m\_systemStartTime
  - AMDTPwrInstrumentedPowerData, [41](#)
- m\_systemTime
  - AMDTPwrSample, [45](#)
- m\_type
  - AMDTPwrDevice, [38](#)
- m\_units
  - AMDTPwrCounterDesc, [35](#)
- m\_userBuffer
  - AMDTPwrInstrumentedPowerData, [41](#)
- Power Profiling, [9](#)
- profiling
  - AMDT\_PWR\_CATEGORY\_CNT, [13](#)
  - AMDT\_PWR\_CATEGORY\_-  
COUNT, [13](#)
  - AMDT\_PWR\_CATEGORY\_-  
CURRENT, [13](#)
  - AMDT\_PWR\_CATEGORY\_DVFS, [13](#)
  - AMDT\_PWR\_CATEGORY\_-  
FREQUENCY, [12](#)
  - AMDT\_PWR\_CATEGORY\_-  
POWER, [12](#)
  - AMDT\_PWR\_CATEGORY\_-  
PROCESS, [13](#)
  - AMDT\_PWR\_CATEGORY\_-  
TEMPERATURE, [12](#)
  - AMDT\_PWR\_CATEGORY\_TIME, [13](#)
  - AMDT\_PWR\_CATEGORY\_-  
VOLTAGE, [12](#)
  - AMDT\_PWR\_DEVICE\_CNT, [12](#)
  - AMDT\_PWR\_DEVICE\_CPU\_-  
COMPUTE\_UNIT, [12](#)
  - AMDT\_PWR\_DEVICE\_CPU\_-  
CORE, [12](#)
  - AMDT\_PWR\_DEVICE\_-  
EXTERNAL\_GPU, [12](#)
  - AMDT\_PWR\_DEVICE\_-  
INTERNAL\_GPU, [12](#)
  - AMDT\_PWR\_DEVICE\_-  
PACKAGE, [12](#)
  - AMDT\_PWR\_DEVICE\_SVI2, [12](#)
  - AMDT\_PWR\_DEVICE\_SYSTEM, [12](#)
  - AMDT\_PWR\_PROFILE\_MODE\_-  
OFFLINE, [12](#)
  - AMDT\_PWR\_PROFILE\_MODE\_-  
ONLINE, [12](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
ABORTED, [14](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
CNT, [14](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
IDLE, [14](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
PAUSED, [14](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
RUNNING, [14](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
STOPPED, [14](#)
  - AMDT\_PWR\_PROFILE\_STATE\_-  
UNINITIALIZED, [14](#)

- AMDT\_PWR\_PSTATE\_P0, [15](#)
- AMDT\_PWR\_PSTATE\_P1, [15](#)
- AMDT\_PWR\_PSTATE\_P2, [15](#)
- AMDT\_PWR\_PSTATE\_P3, [15](#)
- AMDT\_PWR\_PSTATE\_P4, [15](#)
- AMDT\_PWR\_PSTATE\_P5, [15](#)
- AMDT\_PWR\_PSTATE\_P6, [15](#)
- AMDT\_PWR\_PSTATE\_P7, [15](#)
- AMDT\_PWR\_PSTATE\_PB0, [14](#)
- AMDT\_PWR\_PSTATE\_PB1, [14](#)
- AMDT\_PWR\_PSTATE\_PB2, [14](#)
- AMDT\_PWR\_PSTATE\_PB3, [14](#)
- AMDT\_PWR\_PSTATE\_PB4, [14](#)
- AMDT\_PWR\_PSTATE\_PB5, [14](#)
- AMDT\_PWR\_PSTATE\_PB6, [15](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_-  
AVERAGE, [14](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_-  
CNT, [14](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_-  
INSTANTANEOUS, [14](#)
- AMDT\_PWR\_SAMPLE\_VALUE\_-  
LIST, [14](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
CENTIGRADE, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_CNT,  
[13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
COUNT, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
JOULE, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
MEGA\_HERTZ, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
MILLI\_AMPERE, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
MILLI\_SECOND, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
PERCENT, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
RATIO, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
VOLT, [13](#)
- AMDT\_PWR\_UNIT\_TYPE\_-  
WATT, [13](#)
- AMDT\_PWR\_VALUE\_CNT, [13](#)
- AMDT\_PWR\_VALUE\_-  
CUMULATIVE, [13](#)
- AMDT\_PWR\_VALUE\_-  
HISTOGRAM, [13](#)
- AMDT\_PWR\_VALUE\_SINGLE,  
[13](#)
- AMDTApuPStates, [14](#)
- AMDTDeviceType, [12](#)
- AMDTEnableProcessProfiling, [28](#)
- AMDTPwrAggregation, [13](#)
- AMDTPwrCategory, [12](#)
- AMDTPwrDisableCounter, [18](#)
- AMDTPwrEnableAllCounters, [19](#)
- AMDTPwrEnableCounter, [17](#)
- AMDTPwrGetApuPstateInfo, [27](#)
- AMDTPwrGetCounterDesc, [17](#)
- AMDTPwrGetCounterHierarchy, [27](#)
- AMDTPwrGetDeviceCounters, [16](#)
- AMDTPwrGetMinimalTimerSam-  
plingPeriod, [19](#)
- AMDTPwrGetNodeTemperature, [28](#)
- AMDTPwrGetNumEnabledCoun-  
ters, [26](#)
- AMDTPwrGetProfilingState, [22](#)
- AMDTPwrGetSampleValueOption,  
[23](#)
- AMDTPwrGetSystemTopology, [15](#)
- AMDTPwrGetTimerSamplingPe-  
riod, [26](#)
- AMDTPwrIsCounterEnabled, [26](#)
- AMDTPwrPauseProfiling, [21](#)
- AMDTPwrProfileClose, [22](#)
- AMDTPwrProfileInitialize, [15](#)
- AMDTPwrProfileMode, [12](#)
- AMDTPwrProfileState, [13](#)
- AMDTPwrReadAllEnabledCoun-  
ters, [23](#)
- AMDTPwrReadCounterHistogram,  
[24](#)
- AMDTPwrReadCumulativeCounter,  
[25](#)
- AMDTPwrResumeProfiling, [21](#)
- AMDTPwrSetSampleValueOption,  
[22](#)
- AMDTPwrSetTimerSamplingPe-  
riod, [20](#)
- AMDTPwrStartProfiling, [20](#)
- AMDTPwrStopProfiling, [21](#)
- AMDTPwrUnit, [13](#)
- AMDTReadProcessProfileData, [29](#)
- AMDTSampleValueOption, [14](#)