Reference Manual

Generated by Doxygen 1.8.6

Sat Jul 11 2015 08:58:01

Contents

Chapter 1

MeterPU

1.1 Introduction

MeterPU: Library for measuring different metrics on different backends.

1.2 Installation

- Include "MeterPU.h" in your program
- Compile your program with native measurement library paths if necessary.
  - E.g. if CUDA Energy Meter is used, compile your program with NVML library by passing -I and -L flag.
  - For CPU Time Meter, no library is needed.
  - Also pass the following flag if the following Meter is used.
    - CUDA Time Meter: -DENABLE_CUDA_TIME
    - CPU and DRAM Energy Meter: -DENABLE_PCM
    - CUDA GPU Energy Meter: -DENABLE_NVML
    - System Energy Meter: -DENABLE_SYSTEM_ENERGY
- Done!

See more examples in example folder.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Module Documentation

7.1 Interface

Classes

- class MeterPU::Meter< Type >

  The software multi-meters.

7.1.1 Detailed Description

Describe interface exposed to the user programs.

7.2 Traits for Different Meter Type

Classes

- struct MeterPU::Meter_Traits< Type >
  Traits Interface.
- struct MeterPU::CPU_Time
  Time Traits.
- struct MeterPU::CUDA_Time
  Cuda Timer Traits.
- struct MeterPU::PCM_Energy
  PCM Energy Traits.
- struct MeterPU::NVML_Energy< device_id >
  GPU Energy Traits.
- struct MeterPU::System_Energy< gpu_ids >
  System Energy Traits.

Functions

- CPU_Time::ResultType MeterPU::operator- (CPU_Time::Time_Unit const &stop_time, CPU_Time::Time_Unit const &start_time)
  Calculate elapsed time between two time stamp.
- bool MeterPU::operator< (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)
  Check if a time stamp is earlier than another.
- bool MeterPU::operator== (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)
  Check if two time stamps are the same.
- bool MeterPU::operator<= (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)
  Check if a time stamp is earlier or equal to another.

7.2.1 Detailed Description

Distinguish between Energy and Time etc.

7.2.2 Function Documentation

7.2.2.1 CPU_Time::ResultType MeterPU::operator- (CPU_Time::Time_Unit const & *stop_time*, CPU_Time::Time_Unit const & *start_time*) [inline]

Calculate elapsed time between two time stamp.

7.2.2.2 bool MeterPU::operator< (CPU_Time::Time_Unit const & *small_time*, CPU_Time::Time_Unit const & *large_time*) [inline]

Check if a time stamp is earlier than another.

7.2.2.3 bool MeterPU::operator<= (CPU_Time::Time_Unit const & *small_time*, CPU_Time::Time_Unit const & *large_time*) [inline]

Check if a time stamp is earlier or equal to another.

7.2.2.4 bool MeterPU::operator== (CPU_Time::Time_Unit const & *small_time,* CPU_Time::Time_Unit const & *large_time*) [inline]

Check if two time stamps are the same.

7.3 Native Measurement Library Initializer

Classes

- struct MeterPU::Environment_Init

  Library Initializer Interface.
- struct MeterPU::CPU_Time_Environment_Init

  CPU Time Library Initializer.
- struct MeterPU::CUDA_Time_Environment_Init

  GPU Time Library Initializer.
- struct MeterPU::PCM_Energy_Environment_Init

  CPU Energy Library Initializer.
- struct MeterPU::NVML_Energy_Environment_Init

  GPU Energy Library Initializer.
- struct MeterPU::System_Energy_Environment_Init

  System Energy Library Initializer.

7.3.1 Detailed Description

Take GPU energy measurement as an example, initialize NVML library.

7.4 Measurement controller

Classes

- struct MeterPU::Measurement_Controller

  Measurement Controller Interface.
- struct MeterPU::CPU_Time_Measurement_Controller

  CPU Time Measurement Controller.
- struct MeterPU::CUDA_Time_Measurement_Controller

  CUDA-enabled GPU Time Measurement Controller.
- struct MeterPU::PCM_Energy_Measurement_Controller

  CPU and DRAME Energy Measurement Controller.
- struct MeterPU::NVML_Energy_Measurement_Controller< device_id >

  CUDA-enabled GPU Energy Measurement Controller.
- struct MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >

  System Energy Measurement Controller.

Macros

- #define POINTER ((Meter<NVML_Energy<first_device_id> >*)x[array_index])

  A macro for downcast a void pointer to a NVML Energy Meter.
- #define NOTHING
- #define LOOPER(name, return_type, base_code, general_code)

  A macro to build variadic template to recursively apply code snippets.

7.4.1 Detailed Description

Encapsulate logic that use native measurement libraries to start, stop and calculate measurement samples.

7.4.2 Macro Definition Documentation

7.4.2.1 #define LOOPER(*name, return_type, base_code, general_code*)

Value:

```
template <GPU_Device_Id_Type ...default_case>  \
        struct name { static return_type apply(void *x[]) { base_code  } }; \
        template <GPU_Device_Id_Type array_index,  \
        GPU_Device_Id_Type first_device_id,  \
        GPU_Device_Id_Type ...rest> \
        struct name <array_index,first_device_id, rest...> \
        { \
                static return_type apply(void *x[]) \
                { \
                        general_code \
                        name <array_index+1,rest...>::apply(x); \
                } \
        };
```

A macro to build variadic template to recursively apply code snippets.

7.4.2.2 #define NOTHING

7.4.2.3 #define POINTER ((Meter<NVML_Energy<first_device_id> >*)x[array_index])

A macro for downcast a void pointer to a NVML Energy Meter.

Chapter 8

Namespace Documentation

8.1 MeterPU Namespace Reference

Classes

- struct Meter_Traits

  Traits Interface.
- struct NVML_Energy

  GPU Energy Traits.
- struct NVML_Energy_Measurement_Controller

  CUDA-enabled GPU Energy Measurement Controller.
- struct System_Energy

  System Energy Traits.
- struct System_Energy_Measurement_Controller

  System Energy Measurement Controller.
- class Meter

  The software multi-meters.
- struct CPU_Time

  Time Traits.
- struct CUDA_Time

  Cuda Timer Traits.
- struct PCM_Energy

  PCM Energy Traits.
- struct Container_Traits
- struct Power_Vector_Container
- struct Hp_Power_Vector_Container
- struct Time_Vector_Container
- struct Environment_Init

  Library Initializer Interface.
- struct CPU_Time_Environment_Init

  CPU Time Library Initializer.
- struct CUDA_Time_Environment_Init

  GPU Time Library Initializer.
- struct PCM_Energy_Environment_Init

  CPU Energy Library Initializer.
- struct NVML_Energy_Environment_Init

  GPU Energy Library Initializer.
- struct System_Energy_Environment_Init

System Energy Library Initializer.

- struct Measurement_Controller

  Measurement Controller Interface.
- struct CPU_Time_Measurement_Controller

  CPU Time Measurement Controller.
- struct CUDA_Time_Measurement_Controller

  CUDA-enabled GPU Time Measurement Controller.
- struct PCM_Energy_Measurement_Controller

  CPU and DRAME Energy Measurement Controller.

Typedefs

- typedef unsigned int GPU_Device_Id_Type

Functions

- CPU_Time::ResultType operator- (CPU_Time::Time_Unit const &stop_time, CPU_Time::Time_Unit const &start_time)

  Calculate elapsed time between two time stamp.
- bool operator< (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)

  Check if a time stamp is earlier than another.
- bool operator== (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)

  Check if two time stamps are the same.
- bool operator<= (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)

  Check if a time stamp is earlier or equal to another.
- std::ostream & operator<< (std::ostream &out, NVML_Energy<>::Time_Unit ts)

8.1.1 Typedef Documentation

8.1.1.1 typedef unsigned int MeterPU::GPU_Device_Id_Type

8.1.2 Function Documentation

8.1.2.1 std::ostream& MeterPU::operator<< (std::ostream & *out*, NVML_Energy<>::Time_Unit *ts*) [inline]

Chapter 9

Class Documentation

9.1 MeterPU::Container_Traits< Type > Struct Template Reference

`#include <MeterPU.h>`

Public Types

- typedef Type::Data_Type Data_Type
- typedef Type::Const_Iterator_Type Const_Iterator_Type

Static Public Member Functions

- static std::string header_message ()
- static void print (const Data_Type &a)

9.1.1 Member Typedef Documentation

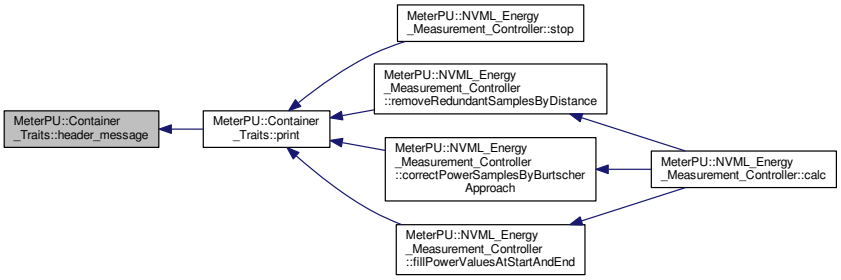
9.1.1.1 template<class Type> typedef Type::Const_Iterator_Type MeterPU::Container_Traits< Type >::Const_Iterator_Type

9.1.1.2 template<class Type> typedef Type::Data_Type MeterPU::Container_Traits< Type >::Data_Type

9.1.2 Member Function Documentation

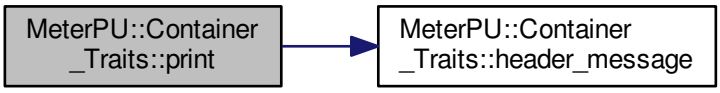
9.1.2.1 template<class Type> static std::string MeterPU::Container_Traits< Type >::header_message ( ) [inline], [static]

Here is the caller graph for this function:

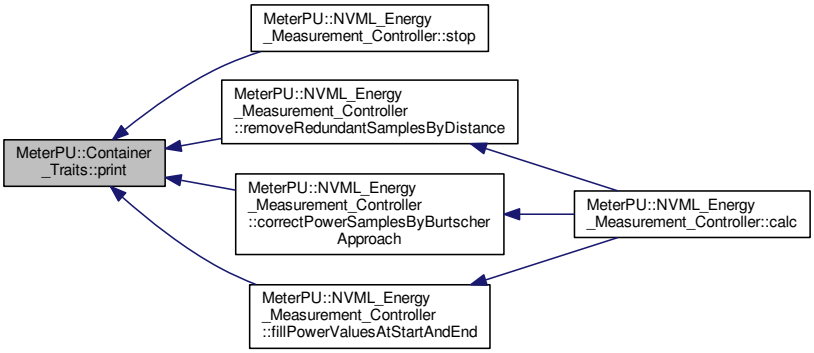


9.1.2.2 template<class Type> static void MeterPU::Container_Traits< Type >::print (const Data_Type & *a*) [inline], [static]

Here is the call graph for this function:



Here is the caller graph for this function:



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

- MeterPU.h

9.2 MeterPU::CPU_Time Struct Reference

Time Traits.

`#include <MeterPU.h>`

Public Types

- typedef CPU_Time_Environment_Init Environment_Init_Type
- typedef double ResultType
- typedef CPU_Time_Measurement_Controller Measurement_Controller
- typedef struct timespec Time_Unit

  Time stamp unit.

9.2.1 Detailed Description

Time Traits.

Describe types relevant to CPU time measurement.

9.2.2 Member Typedef Documentation

9.2.2.1 typedef CPU_Time_Environment_Init MeterPU::CPU_Time::Environment_Init_Type

9.2.2.2 typedef CPU_Time_Measurement_Controller MeterPU::CPU_Time::Measurement_Controller

9.2.2.3 typedef double MeterPU::CPU_Time::ResultType

9.2.2.4 typedef struct timespec MeterPU::CPU_Time::Time_Unit

Time stamp unit.

Different with ResultType, since the latter is the elapsed time.

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

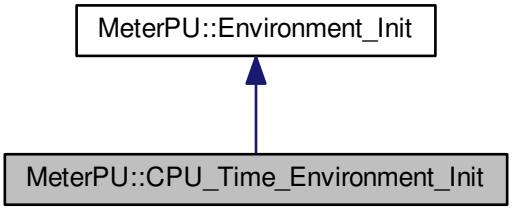
- MeterPU.h

9.3 MeterPU::CPU_Time_Environment_Init Struct Reference

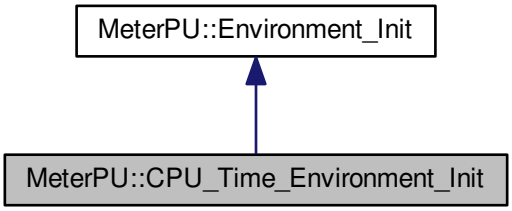
CPU Time Library Initializer.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::CPU_Time_Environment_Init:



Collaboration diagram for MeterPU::CPU_Time_Environment_Init:



Public Member Functions

- void init ()

Private Member Functions

- DECLARE_CLASS_NAME ("CPU_Time_Environment_Init")

9.3.1 Detailed Description

CPU Time Library Initializer.

9.3.2 Member Function Documentation

9.3.2.1 MeterPU::CPU_Time_Environment_Init::DECLARE_CLASS_NAME ("CPU_Time_Environment_Init") [private]

9.3.2.2 void MeterPU::CPU_Time_Environment_Init::init ( ) [inline], [virtual]

Implements MeterPU::Environment_Init.

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

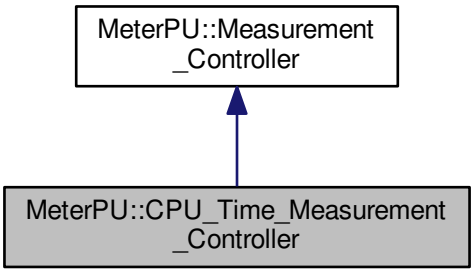
- MeterPU.h

9.4 MeterPU::CPU_Time_Measurement_Controller Struct Reference

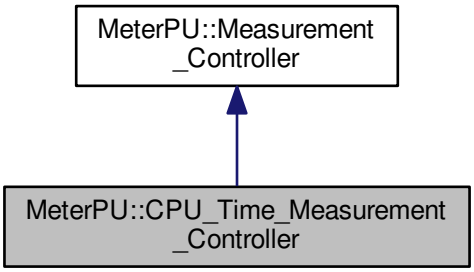
CPU Time Measurement Controller.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::CPU_Time_Measurement_Controller:



Collaboration diagram for MeterPU::CPU_Time_Measurement_Controller:



Public Member Functions

- CPU_Time_Measurement_Controller ()
- CPU_Time_Measurement_Controller (CPU_Time::Time_Unit const &start_time_p, CPU_Time::Time_Unit const &stop_time_p)
- void init ()
- void start ()

  mark the start of a measurement phase/period.
- void stop ()

  mark the end of a measurement phase/period.

- void calc ()

  calculate the metric value between start() and stop().
- CPU_Time::ResultType const & get_value () const

  Get calculated metric value, require calc() to be called already.
- void show_meter_reading () const

  Print the calculated metric value to standard output, requires an invocation of calc() already done.

Private Member Functions

- DECLARE_CLASS_NAME ("CPU_Time_Measurement_Controller")

Private Attributes

- CPU_Time::Time_Unit start_time
- CPU_Time::Time_Unit stop_time
- CPU_Time::ResultType meter_reading

9.4.1 Detailed Description

CPU Time Measurement Controller.

Internally it uses clock_gettime() etc.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 MeterPU::CPU_Time_Measurement_Controller::CPU_Time_Measurement_Controller ( ) [inline]

9.4.2.2 MeterPU::CPU_Time_Measurement_Controller::CPU_Time_Measurement_Controller (CPU_Time::Time_Unit const & *start_time_p*, CPU_Time::Time_Unit const & *stop_time_p*) [inline]

9.4.3 Member Function Documentation

9.4.3.1 void MeterPU::CPU_Time_Measurement_Controller::calc ( ) [inline], [virtual]

calculate the metric value between start() and stop().

Implements MeterPU::Measurement_Controller.

Here is the caller graph for this function:

9.4.3.2 MeterPU::CPU_Time_Measurement_Controller::DECLARE_CLASS_NAME ("CPU_Time_Measurement_Controller") [private]

9.4.3.3 CPU_Time::ResultType const& MeterPU::CPU_Time_Measurement_Controller::get_value ( ) const [inline]

Get calculated metric value, require calc() to be called already.

Here is the caller graph for this function:

9.4.3.4 void MeterPU::CPU_Time_Measurement_Controller::init ( ) [inline]

9.4.3.5 void MeterPU::CPU_Time_Measurement_Controller::show_meter_reading ( ) const [inline], [virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:

9.4.3.6 void MeterPU::CPU_Time_Measurement_Controller::start ( ) [inline], [virtual]

mark the start of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

9.4.3.7 void MeterPU::CPU_Time_Measurement_Controller::stop ( ) [inline], [virtual]

mark the end of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

9.4.4 Member Data Documentation

9.4.4.1 CPU_Time::ResultType MeterPU::CPU_Time_Measurement_Controller::meter_reading [private]

9.4.4.2 CPU_Time::Time_Unit MeterPU::CPU_Time_Measurement_Controller::start_time [private]

9.4.4.3 CPU_Time::Time_Unit MeterPU::CPU_Time_Measurement_Controller::stop_time [private]

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

- MeterPU.h

9.5 MeterPU::CUDA_Time Struct Reference

Cuda Timer Traits.

`#include <MeterPU.h>`

Public Types

- typedef CUDA_Time_Environment_Init Environment_Init_Type
- typedef float ResultType
- typedef CUDA_Time_Measurement_Controller Measurement_Controller

9.5.1 Detailed Description

Cuda Timer Traits.

Describe types relevant to GPU time measurement.

9.5.2 Member Typedef Documentation

9.5.2.1 typedef CUDA_Time_Environment_Init MeterPU::CUDA_Time::Environment_Init_Type

9.5.2.2 typedef CUDA_Time_Measurement_Controller MeterPU::CUDA_Time::Measurement_Controller

9.5.2.3 typedef float MeterPU::CUDA_Time::ResultType

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

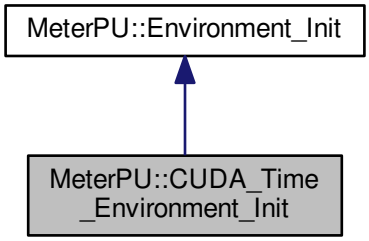
- MeterPU.h

9.6 MeterPU::CUDA_Time_Environment_Init Struct Reference

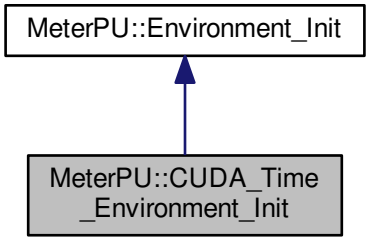
GPU Time Library Initializer.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::CUDA_Time_Environment_Init:



Collaboration diagram for MeterPU::CUDA_Time_Environment_Init:



Public Member Functions

- void init ()

Private Member Functions

- DECLARE_CLASS_NAME ("CUDA_Time_Environment_Init")

9.6.1 Detailed Description

GPU Time Library Initializer.

9.6.2 Member Function Documentation

9.6.2.1 MeterPU::CUDA_Time_Environment_Init::DECLARE_CLASS_NAME ("CUDA_Time_Environment_Init") [private]

9.6.2.2 void MeterPU::CUDA_Time_Environment_Init::init ( ) [inline], [virtual]

Implements MeterPU::Environment_Init.

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

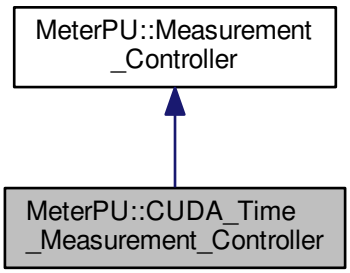
- MeterPU.h

9.7 MeterPU::CUDA_Time_Measurement_Controller Struct Reference

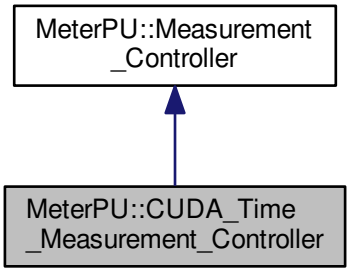
CUDA-enabled GPU Time Measurement Controller.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::CUDA_Time_Measurement_Controller:



Collaboration diagram for MeterPU::CUDA_Time_Measurement_Controller:



Public Member Functions

- CUDA_Time_Measurement_Controller ()
- ~CUDA_Time_Measurement_Controller ()
- void init ()
- void start ()

  mark the start of a measurement phase/period.
- void stop ()

  mark the end of a measurement phase/period.
- void calc ()

  calculate the metric value between start() and stop().
- void show_meter_reading () const

  Print the calculated metric value to standard output, requires an invocation of calc() already done.
- CUDA_Time::ResultType const & get_value () const

  Get calculated metric value, require calc() to be called already.

Private Attributes

- cudaEvent_t start_time
- cudaEvent_t stop_time
- CUDA_Time::ResultType meter_reading

9.7.1 Detailed Description

CUDA-enabled GPU Time Measurement Controller.

It internally uses cudaEventCreate() etc.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 MeterPU::CUDA_Time_Measurement_Controller::CUDA_Time_Measurement_Controller ( ) [inline]

9.7.2.2 MeterPU::CUDA_Time_Measurement_Controller::~CUDA_Time_Measurement_Controller ( ) [inline]

9.7.3 Member Function Documentation

9.7.3.1 void MeterPU::CUDA_Time_Measurement_Controller::calc ( ) [inline], [virtual]

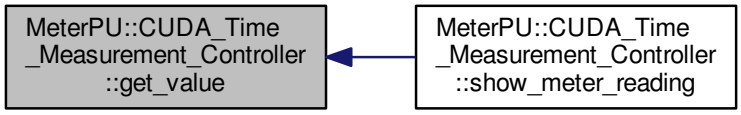
calculate the metric value between start() and stop().

Implements MeterPU::Measurement_Controller.

9.7.3.2 CUDA_Time::ResultType const& MeterPU::CUDA_Time_Measurement_Controller::get_value ( ) const [inline]

Get calculated metric value, require calc() to be called already.

Here is the caller graph for this function:



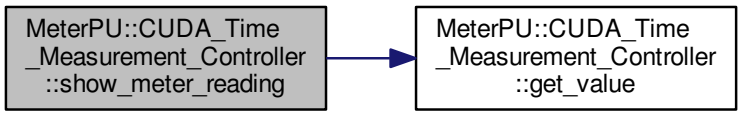
9.7.3.3 void MeterPU::CUDA_Time_Measurement_Controller::init ( ) [inline]

9.7.3.4 void MeterPU::CUDA_Time_Measurement_Controller::show_meter_reading ( ) const [inline], [virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



9.7.3.5 void MeterPU::CUDA_Time_Measurement_Controller::start ( ) [inline], [virtual]

mark the start of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

9.7.3.6 void MeterPU::CUDA_Time_Measurement_Controller::stop ( ) [inline], [virtual]

mark the end of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

9.7.4 Member Data Documentation

9.7.4.1 CUDA_Time::ResultType MeterPU::CUDA_Time_Measurement_Controller::meter_reading [private]

9.7.4.2 cudaEvent_t MeterPU::CUDA_Time_Measurement_Controller::start_time [private]

9.7.4.3 cudaEvent_t MeterPU::CUDA_Time_Measurement_Controller::stop_time [private]

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

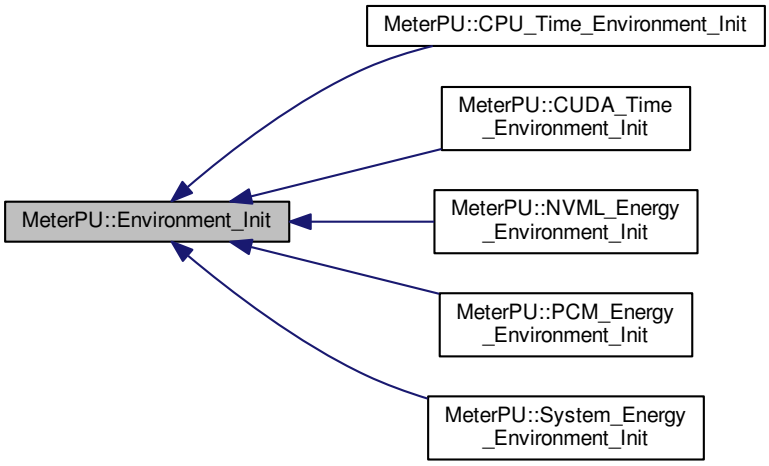
- MeterPU.h

9.8 MeterPU::Environment_Init Struct Reference

Library Initializer Interface.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::Environment_Init:



Public Member Functions

- virtual void init ()=0

9.8.1 Detailed Description

Library Initializer Interface.

9.8.2 Member Function Documentation

9.8.2.1 virtual void MeterPU::Environment_Init::init ( ) [pure virtual]

Implemented in MeterPU::System_Energy_Environment_Init, MeterPU::NVML_Energy_Environment_Init, MeterPU::PCM_Energy_Environment_Init, MeterPU::CUDA_Time_Environment_Init, and MeterPU::CPU_Time_Environment_Init.

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

- MeterPU.h

9.9 MeterPU::Hp_Power_Vector_Container Struct Reference

`#include <MeterPU.h>`

Public Types

- typedef NVML_Energy::Hp_Power_DB_Type Data_Type
- typedef NVML_Energy::Hp_Power_DB_Const_Iterator_Type Const_Iterator_Type

Static Public Member Functions

- static std::string header_message ()

9.9.1 Member Typedef Documentation

9.9.1.1 typedef NVML_Energy ::Hp_Power_DB_Const_Iterator_Type MeterPU::Hp_Power_Vector_Container::-Const_Iterator_Type

9.9.1.2 typedef NVML_Energy ::Hp_Power_DB_Type MeterPU::Hp_Power_Vector_Container::Data_Type

9.9.2 Member Function Documentation

9.9.2.1 static std::string MeterPU::Hp_Power_Vector_Container::header_message ( ) [inline], [static]

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

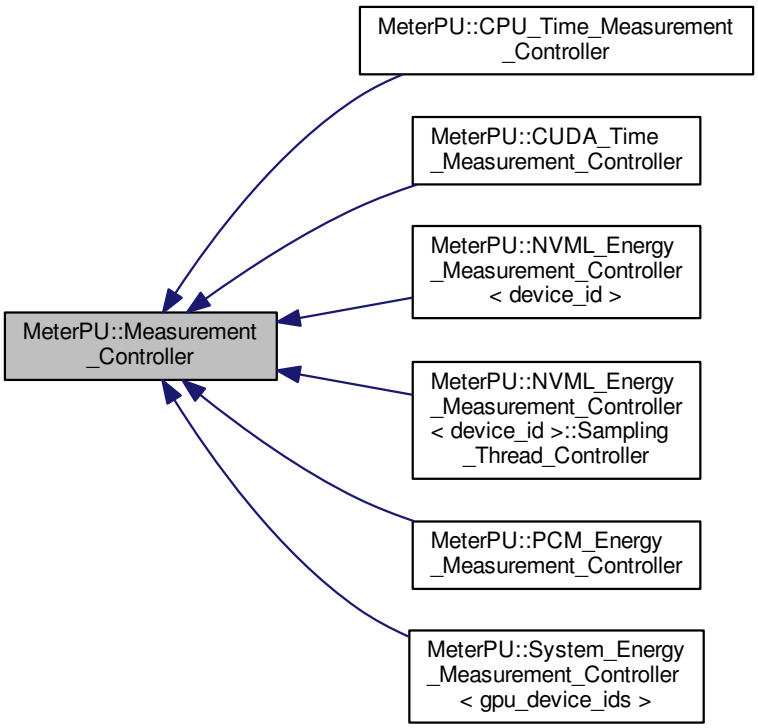
- MeterPU.h

9.10 MeterPU::Measurement_Controller Struct Reference

Measurement Controller Interface.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::Measurement_Controller:



Public Member Functions

- virtual void start ()=0

  mark the start of a measurement phase/period.

- virtual void stop ()=0

  mark the end of a measurement phase/period.

- virtual void calc ()=0

  calculate the metric value between start() and stop().

- virtual void show_meter_reading () const =0

  Print the calculated metric value to standard output, requires an invocation of calc() already done.

9.10.1 Detailed Description

Measurement Controller Interface.

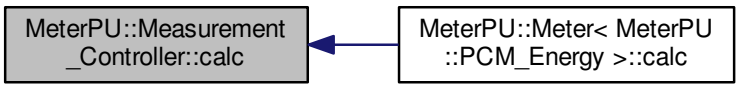
9.10.2 Member Function Documentation

9.10.2.1 virtual void MeterPU::Measurement_Controller::calc ( ) [pure virtual]

calculate the metric value between start() and stop().

Implemented in MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >, MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller, MeterPU::NVML_Energy_Measurement_Controller< device_id >, MeterPU::PCM_Energy_Measurement_Controller, MeterPU::CUDA_Time_Measurement_Controller, and MeterPU::CPU_Time_Measurement_Controller.

Here is the caller graph for this function:

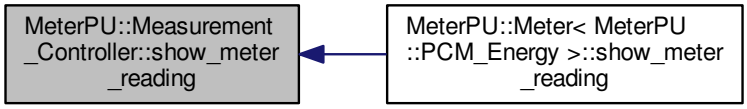


9.10.2.2 virtual void MeterPU::Measurement_Controller::show_meter_reading ( ) const [pure virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

Implemented in MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >, MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller, MeterPU::NVML_Energy_Measurement_Controller< device_id >, MeterPU::PCM_Energy_Measurement_Controller, MeterPU::CUDA_Time_Measurement_Controller, and MeterPU::CPU_Time_Measurement_Controller.

Here is the caller graph for this function:

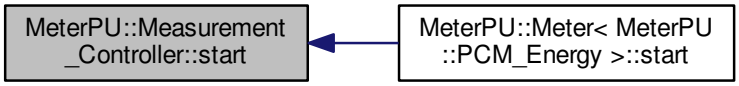


9.10.2.3 virtual void MeterPU::Measurement_Controller::start ( ) [pure virtual]

mark the start of a measurement phase/period.

Implemented in MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >, MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller, MeterPU::NVML_Energy_Measurement_Controller< device_id >, MeterPU::PCM_Energy_Measurement_Controller, MeterPU::CUDA_Time_Measurement_Controller, and MeterPU::CPU_Time_Measurement_Controller.

Here is the caller graph for this function:

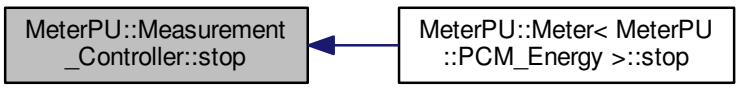


9.10.2.4 virtual void MeterPU::Measurement_Controller::stop ( ) [pure virtual]

mark the end of a measurement phase/period.

Implemented in MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >, MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller, MeterPU::NVML_Energy_Measurement_Controller< device_id >, MeterPU::PCM_Energy_Measurement_Controller, MeterPU::CUDA_Time_Measurement_Controller, and MeterPU::CPU_Time_Measurement_Controller.

Here is the caller graph for this function:



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

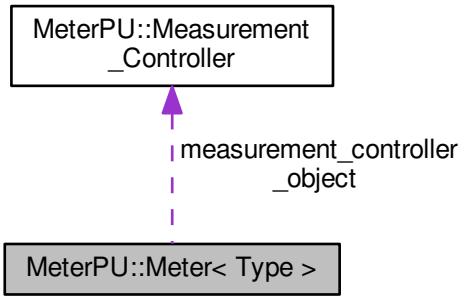
- MeterPU.h

9.11 MeterPU::Meter< Type > Class Template Reference

The software multi-meters.

`#include <MeterPU.h>`

Collaboration diagram for MeterPU::Meter< Type >:



Public Member Functions

- Meter ()
- void start ()

  start a measurement
- void stop ()

  stop a measurement
- void calc ()

  calculate measurement value
- Meter_Traits< Type >
  ::ResultType const & get_value () const

  Get calculated metric value, require calc() to be called already.
- void show_meter_reading ()

  Write measurement value on standard output with its unit.

Private Attributes

- Meter_Traits< Type >
  ::Environment_Init_Type environment_init_object

  Native library initializer.
- Meter_Traits< Type >
  ::Measurement_Controller measurement_controller_object

  Measurement controller.
- Meter_Traits< Type >::ResultType meter_reading

  The variable used to store the calculated.

9.11.1 Detailed Description

template<class Type>class MeterPU::Meter< Type >

The software multi-meters.

The only class the user program should use. When equipped with different template parameter, it can measure different metrics on code regions. So far supported metrics are

9.11.2 Constructor & Destructor Documentation

9.11.2.1 template<class Type> MeterPU::Meter< Type >::Meter ( ) [inline]

9.11.3 Member Function Documentation

9.11.3.1 template<class Type> void MeterPU::Meter< Type >::calc ( ) [inline]

calculate measurement value

9.11.3.2 template<class Type> Meter_Traits<Type>::ResultType const& MeterPU::Meter< Type >::get_value ( ) const [inline]

Get calculated metric value, require calc() to be called already.

9.11.3.3 template<class Type> void MeterPU::Meter< Type >::show_meter_reading ( ) [inline]

Write measurement value on standard output with its unit.

9.11.3.4 template<class Type> void MeterPU::Meter< Type >::start ( ) [inline]

start a measurement

9.11.3.5 template<class Type> void MeterPU::Meter< Type >::stop ( ) [inline]

stop a measurement

9.11.4 Member Data Documentation

9.11.4.1 template<class Type> Meter_Traits<Type>::Environment_Init_Type MeterPU::Meter< Type >::environment_init_object [private]

Native library initializer.

For GPU, initialize NVML library.

9.11.4.2 template<class Type> Meter_Traits<Type>::Measurement_Controller MeterPU::Meter< Type >::measurement_controller_object [private]

Measurement controller.

The object that do the measurement, Meter class pass control signal such as start and stop to it.

9.11.4.3 template<class Type> Meter_Traits<Type>::ResultType MeterPU::Meter< Type >::meter_reading [private]

The variable used to store the calculated.

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

- MeterPU.h

9.12 MeterPU::Meter_Traits< Type > Struct Template Reference

Traits Interface.

`#include <MeterPU.h>`

Public Types

- typedef Type::Environment_Init_Type Environment_Init_Type

  Object for initialization of a native measurement library.
- typedef Type::ResultType ResultType

  Result type.
- typedef Type::Measurement_Controller Measurement_Controller

  Measurement Controller Object.

9.12.1 Detailed Description

template<class Type>struct MeterPU::Meter_Traits< Type >

Traits Interface.

If initialized with a type for measurement, all types stores in Meter_Traits changes accordingly.

9.12.2 Member Typedef Documentation

9.12.2.1 template<class Type> typedef Type::Environment_Init_Type MeterPU::Meter_Traits< Type >::Environment_Init_Type

Object for initialization of a native measurement library.

MeterPU use native measurement libraries for measurement, e.g. NVML for GPU. If a native library needs initialization, then the initialization class should be implemented.

9.12.2.2 template<class Type> typedef Type::Measurement_Controller MeterPU::Meter_Traits< Type >::Measurement_Controller

Measurement Controller Object.

The Measurement Controller class implement functionality needed to start, stop and calculation of a measurement by a native measurement library.

9.12.2.3 template<class Type> typedef Type::ResultType MeterPU::Meter_Traits< Type >::ResultType

Result type.

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

- MeterPU.h

9.13 MeterPU::NVML_Energy< device_id > Struct Template Reference

GPU Energy Traits.

`#include <MeterPU.h>`

Public Types

- typedef double Energy_Unit
- typedef NVML_Energy_Environment_Init Environment_Init_Type
- typedef Energy_Unit ResultType
- typedef NVML_Energy_Measurement_Controller < device_id > Measurement_Controller
- typedef struct timespec Time_Unit
- typedef unsigned int Power_Unit
- typedef long double Hp_Power_Unit

  High precision power unit.
- typedef std::vector< Time_Unit > Time_DB_Type
- typedef Time_DB_Type::const_iterator Time_DB_Const_Iterator_Type
- typedef std::vector< Power_Unit > Power_DB_Type
- typedef std::vector < Hp_Power_Unit > Hp_Power_DB_Type
- typedef Power_DB_Type::const_iterator Power_DB_Const_Iterator_Type
- typedef Hp_Power_DB_Type::const_iterator Hp_Power_DB_Const_Iterator_Type

9.13.1 Detailed Description

template<GPU_Device_Id_Type device_id = 0>struct MeterPU::NVML_Energy< device_id >

GPU Energy Traits.

Describe types relevant to energy measurement. Customed GPU device id is possible by template argument.

9.13.2 Member Typedef Documentation

9.13.2.1 template<GPU_Device_Id_Type device_id = 0> typedef double MeterPU::NVML_Energy< device_id >::Energy_Unit

9.13.2.2 template<GPU_Device_Id_Type device_id = 0> typedef NVML_Energy_Environment_Init MeterPU::NVML_Energy< device_id >::Environment_Init_Type

9.13.2.3 template<GPU_Device_Id_Type device_id = 0> typedef Hp_Power_DB_Type::const_iterator MeterPU::NVML_Energy< device_id >::Hp_Power_DB_Const_Iterator_Type

9.13.2.4 template<GPU_Device_Id_Type device_id = 0> typedef std::vector<Hp_Power_Unit> MeterPU::NVML_Energy< device_id >::Hp_Power_DB_Type

9.13.2.5 template<GPU_Device_Id_Type device_id = 0> typedef long double MeterPU::NVML_Energy< device_id >::Hp_Power_Unit

High precision power unit.

Used for calculation with minimal precision loss.

9.13.2.6 template<GPU_Device_Id_Type device_id = 0> typedef NVML_Energy_Measurement_Controller<device_id> MeterPU::NVML_Energy< device_id >::Measurement_Controller

9.13.2.7 template<GPU_Device_Id_Type device_id = 0> typedef Power_DB_Type::const_iterator MeterPU::NVML_Energy< device_id >::Power_DB_Const_Iterator_Type

9.13.2.8 template<GPU_Device_Id_Type device_id = 0> typedef std::vector<Power_Unit> MeterPU::NVML_Energy< device_id >::Power_DB_Type

9.13.2.9 template<GPU_Device_Id_Type device_id = 0> typedef unsigned int MeterPU::NVML_Energy< device_id >::Power_Unit

9.13.2.10 template<GPU_Device_Id_Type device_id = 0> typedef Energy_Unit MeterPU::NVML_Energy< device_id >::ResultType

9.13.2.11 template<GPU_Device_Id_Type device_id = 0> typedef Time_DB_Type::const_iterator MeterPU::NVML_Energy< device_id >::Time_DB_Const_Iterator_Type

9.13.2.12 template<GPU_Device_Id_Type device_id = 0> typedef std::vector<Time_Unit> MeterPU::NVML_Energy< device_id >::Time_DB_Type

9.13.2.13 template<GPU_Device_Id_Type device_id = 0> typedef struct timespec MeterPU::NVML_Energy< device_id >::Time_Unit

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

- MeterPU.h

9.14 MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value > Struct Template Reference

Public Member Functions

- NVML_Energy_Device_Init ()
- void set_device_id (GPU_Device_Id_Type id)
- void init ()
- const nvmlDevice_t & get_device () const

Private Attributes

- GPU_Device_Id_Type device_id

Device id.

- nvmlDevice_t device

  Device handle.

9.14.1 Constructor & Destructor Documentation

9.14.1.1 template<GPU_Device_Id_Type device_id> template<GPU_Device_Id_Type device_id_value = device_id_NVML_Energy_Measurement_Controller> MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >::NVML_Energy_Device_Init ( ) [inline]

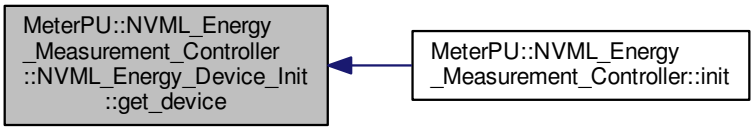
Default way to initialize device number.

9.14.2 Member Function Documentation

9.14.2.1 template<GPU_Device_Id_Type device_id> template<GPU_Device_Id_Type device_id_value = device_id_NVML_Energy_Measurement_Controller> const nvmlDevice_t& MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >::get_device ( ) const [inline]

Expose to Energy measurement, not visible to abstration.

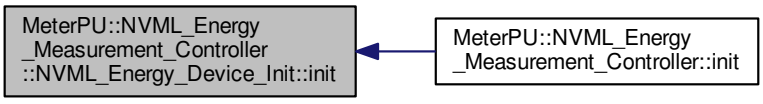
Here is the caller graph for this function:



9.14.2.2 template<GPU_Device_Id_Type device_id> template<GPU_Device_Id_Type device_id_value = device_id_NVML_Energy_Measurement_Controller> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >::init ( ) [inline]

Initialize the device by device_id.

Here is the caller graph for this function:



9.14.2.3 template<GPU_Device_Id_Type device_id> template<GPU_Device_Id_Type device_id_value = device_id_NVML_Energy_Measurement_Controller> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >::set_device_id (GPU_Device_Id_Type *id*) [inline]

Customed way to initialize device number.

9.14.3 Member Data Documentation

9.14.3.1 template<GPU_Device_Id_Type device_id> template<GPU_Device_Id_Type device_id_value = device_id_NVML_Energy_Measurement_Controller> nvmlDevice_t MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >::device [private]

Device handle.

initialized by device_id.

9.14.3.2 template<GPU_Device_Id_Type device_id> template<GPU_Device_Id_Type device_id_value = device_id_NVML_Energy_Measurement_Controller> GPU_Device_Id_Type MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >::device_id [private]

Device id.

By default, it is zero, the first GPU installed on the target system.

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

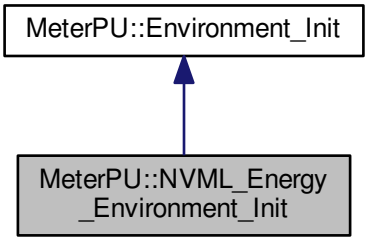
- MeterPU.h

9.15 MeterPU::NVML_Energy_Environment_Init Struct Reference

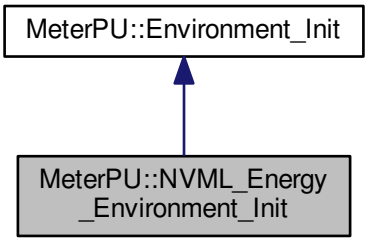
GPU Energy Library Initializer.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::NVML_Energy_Environment_Init:



Collaboration diagram for MeterPU::NVML_Energy_Environment_Init:



Classes

- class NVML_Manager

  NVML library init and teardown.

Public Member Functions

- void init ()

9.15.1 Detailed Description

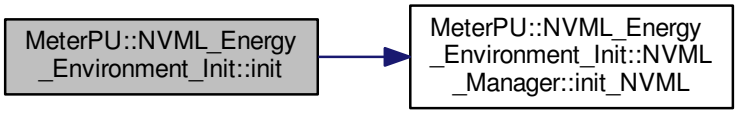
GPU Energy Library Initializer.

9.15.2 Member Function Documentation

9.15.2.1 void MeterPU::NVML_Energy_Environment_Init::init ( ) [inline], [virtual]

Implements MeterPU::Environment_Init.

Here is the call graph for this function:



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

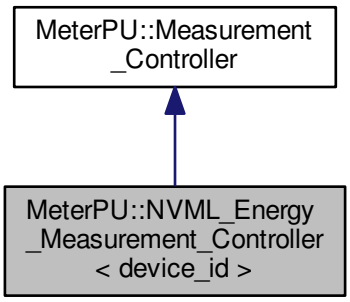
- MeterPU.h

9.16 MeterPU::NVML_Energy_Measurement_Controller< device_id > Struct Template Reference

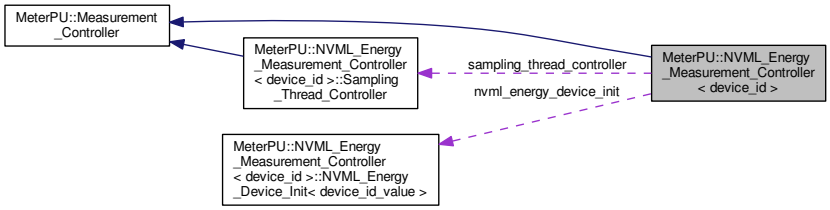
CUDA-enabled GPU Energy Measurement Controller.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::NVML_Energy_Measurement_Controller< device_id >:



Collaboration diagram for MeterPU::NVML_Energy_Measurement_Controller< device_id >:



Classes

- struct NVML_Energy_Device_Init
- class Sampling_Thread_Controller

  Sampling Thread Controller.

Public Member Functions

- void init ()

  Initialize a GPU device.
- void start ()

  mark the start of a measurement phase/period.
- void stop ()

  mark the end of a measurement phase/period.
- void calc ()

calculate the metric value between start() and stop().

- NVML_Energy::Energy_Unit get_value () const

  Get calculated metric value, require calc() to be called already.

- void show_meter_reading () const

  Print the calculated metric value to standard output, requires an invocation of calc() already done.

Private Member Functions

- DECLARE_CLASS_NAME ("NVML_Energy_Measurement_Controller")
- const CPU_Time::Time_Unit & get_start_time () const
- const CPU_Time::Time_Unit & get_stop_time () const
- void record_start_time ()
- void record_stop_time ()
- CPU_Time::ResultType diff ()
- void resetDir ()
- void dumpTimeEvent ()
- template<class T1 , class T2 >
  void dumpTwoVectors (const std::string &filename, const std::string &header, const T1 &keys, const T2 &values) const
- void dumpOriginalPowerData () const
- void dumpCorrectedPowerData ()
- void removeRedundantSamplesByDistance (const CPU_Time::ResultType &time_distance_ms)
- void correctPowerSamplesByBurtscherApproach ()
- void fillPowerValuesAtStartAndEnd ()
- NVML_Energy::Energy_Unit calTotalArea ()
- NVML_Energy::Energy_Unit calTotalArea (NVML_Energy<>::Hp_Power_DB_Type const &power_db, NVML_Energy<>::Time_DB_Type const &time_db)
- NVML_Energy::Energy_Unit calAreaForOneTrapezoid (NVML_Energy<>::Time_Unit start_time, NVML_Energy<>::Time_Unit end_time, NVML_Energy<>::Hp_Power_Unit start_power, NVML_Energy<>::Hp_Power_Unit end_power)

Private Attributes

- NVML_Energy_Device_Init nvml_energy_device_init
- nvmlDevice_t device
- NVML_Energy::Energy_Unit meter_reading
- CPU_Time::Time_Unit start_time
- CPU_Time::Time_Unit stop_time
- std::string path
- NVML_Energy::Hp_Power_DB_Type correctedPowerDB
- Sampling_Thread_Controller sampling_thread_controller

9.16.1 Detailed Description

template<GPU_Device_Id_Type device_id>struct MeterPU::NVML_Energy_Measurement_Controller< device_id >

CUDA-enabled GPU Energy Measurement Controller.

It internally use NVML `nvmlDeviceGetPowerUsage()`, note that not all CUDA-enabled GPU support this functionality, please check your GPU specification.

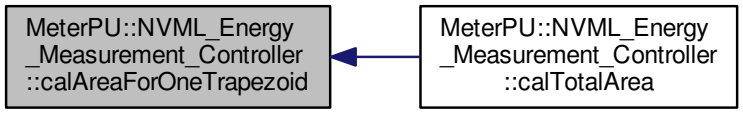
To calculate the energy value based on power samples obtained by NVML, it implement the techniques described in the following paper:

[1] Martin Burtscher, Ivan Zecena, and Ziliang Zong. 2014. Measuring GPU Power with the K20 Built-in Sensor. In Proceedings of Workshop on General Purpose Processing Using GPUs (GPGPU-7). ACM, New York, NY, USA, Pages 28 , 9 pages. DOI=10.1145/2576779.2576783 http://doi.acm.org/10.-1145/2576779.2576783

9.16.2 Member Function Documentation

9.16.2.1 template<GPU_Device_Id_Type device_id> NVML_Energy ::Energy_Unit MeterPU::NVML_Energy_-Measurement_Controller< device_id >::calAreaForOneTrapezoid (NVML_Energy<>::Time_Unit *start_time,* NVML_Energy<>::Time_Unit *end_time,* NVML_Energy<>::Hp_Power_Unit *start_power,* NVML_Energy<>::Hp_Power_Unit *end_power*) [inline], [private]

Here is the caller graph for this function:

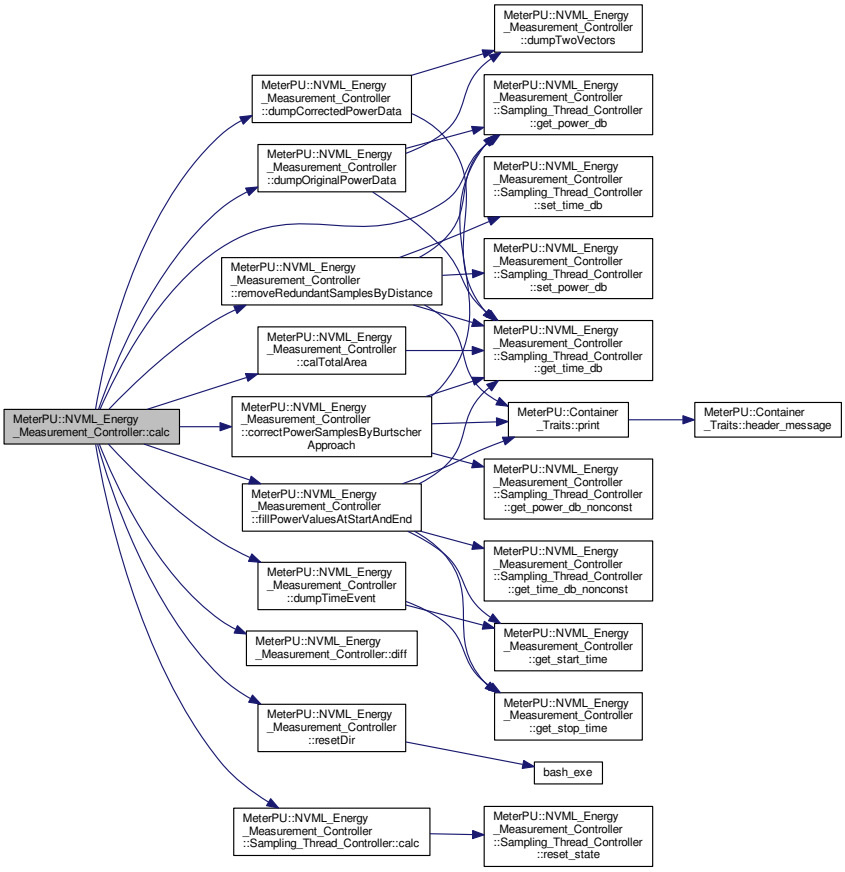


9.16.2.2 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::calc ( ) [inline], [virtual]

calculate the metric value between start() and stop().

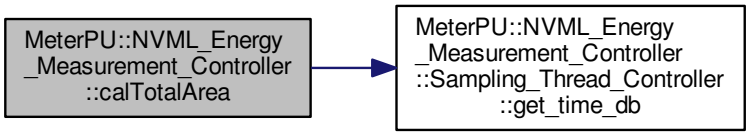
Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:

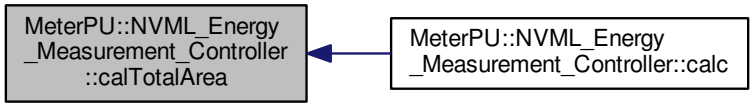


9.16.2.3 template<GPU_Device_Id_Type device_id> NVML_Energy ::Energy_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::calTotalArea ( ) [inline], [private]

Here is the call graph for this function:

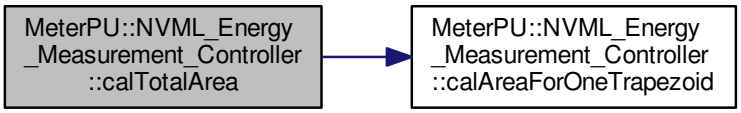


Here is the caller graph for this function:



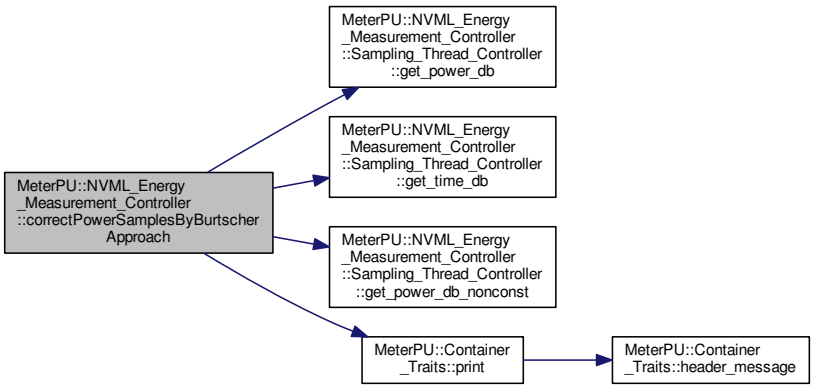
9.16.2.4 template<GPU_Device_Id_Type device_id> NVML_Energy ::Energy_Unit MeterPU::NVML_Energy_-Measurement_Controller< device_id >::calTotalArea (NVML_Energy<>::Hp_Power_DB_Type const & *power_db,* NVML_Energy<>::Time_DB_Type const & *time_db*) [inline], [private]

Here is the call graph for this function:

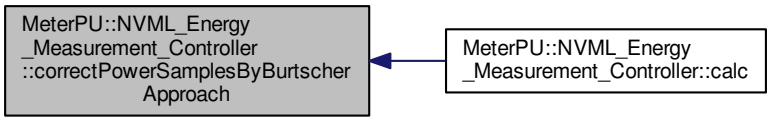


9.16.2.5 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::correctPowerSamplesByBurtscherApproach ( ) [inline], [private]

Here is the call graph for this function:



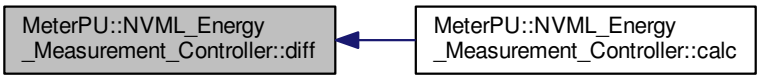
Here is the caller graph for this function:



9.16.2.6 template<GPU_Device_Id_Type device_id> MeterPU::NVML_Energy_Measurement_Controller< device_id >::DECLARE_CLASS_NAME ("NVML_Energy_Measurement_Controller< device_id >") [private]

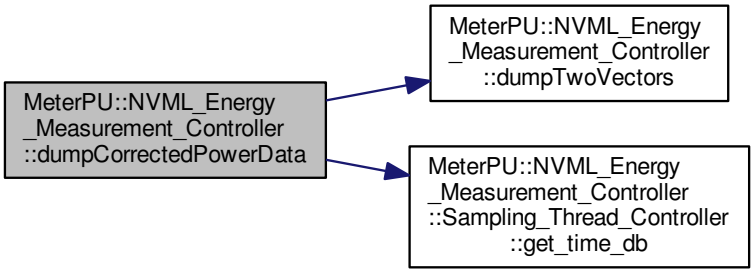
9.16.2.7 template<GPU_Device_Id_Type device_id> CPU_Time::ResultType MeterPU::NVML_Energy_Measurement_Controller< device_id >::diff ( ) [inline], [private]

Here is the caller graph for this function:

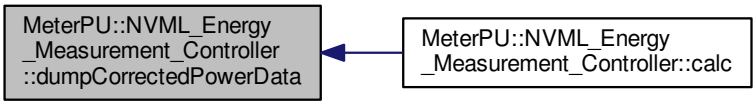


9.16.2.8 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::dumpCorrectedPowerData ( ) [inline], [private]

Here is the call graph for this function:

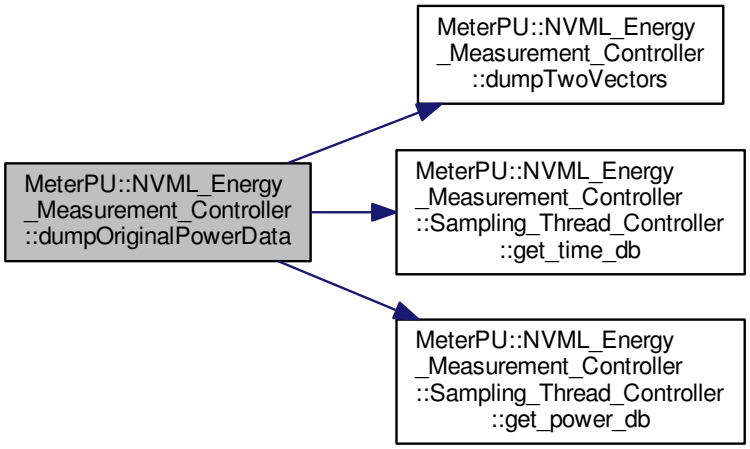


Here is the caller graph for this function:

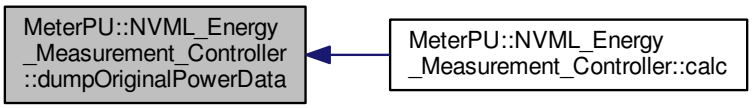


9.16.2.9 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::dumpOriginalPowerData ( ) const [inline], [private]

Here is the call graph for this function:

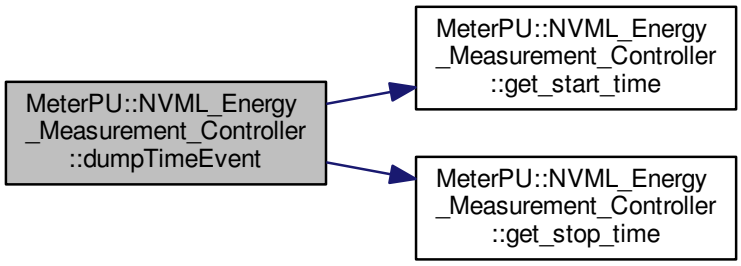


Here is the caller graph for this function:

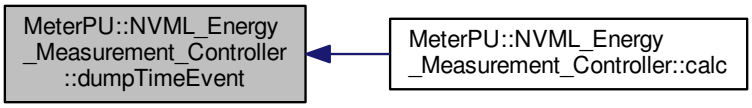


9.16.2.10 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::dumpTimeEvent ( ) [inline], [private]

Here is the call graph for this function:

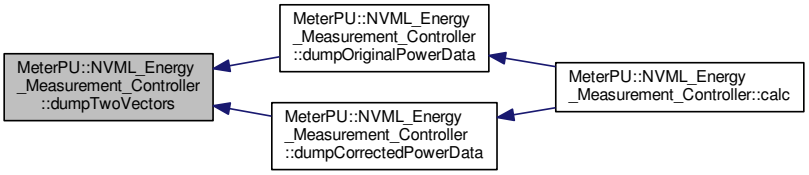


Here is the caller graph for this function:



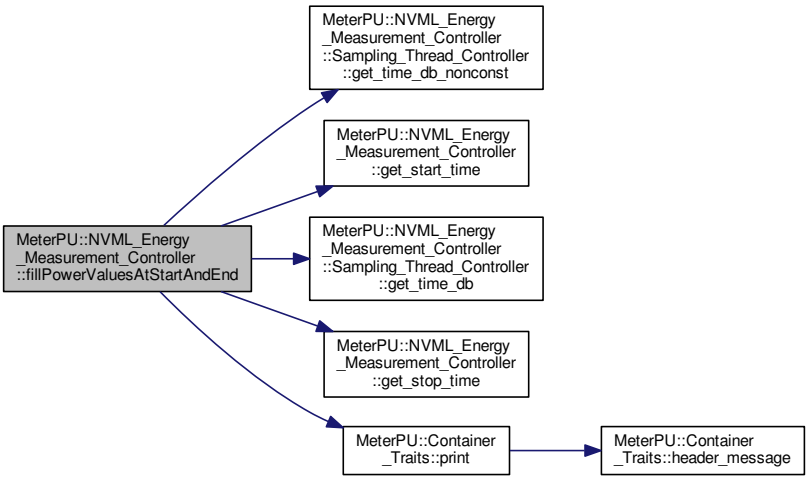
9.16.2.11 template<GPU_Device_Id_Type device_id> template<class T1 , class T2 > void MeterPU::NVML_Energy_-Measurement_Controller< device_id >::dumpTwoVectors (const std::string & *filename,* const std::string & *header,* const T1 & *keys,* const T2 & *values*) const [inline], [private]

Here is the caller graph for this function:

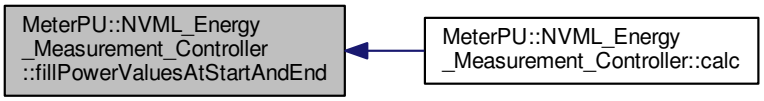


9.16.2.12 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::fillPowerValuesAtStartAndEnd ( ) [inline], [private]

Here is the call graph for this function:

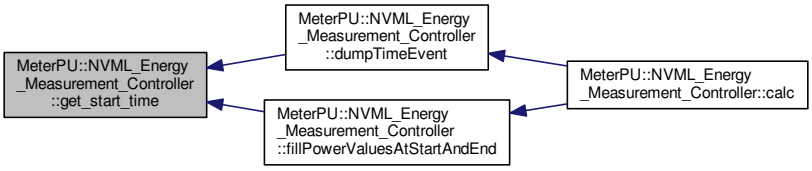


Here is the caller graph for this function:



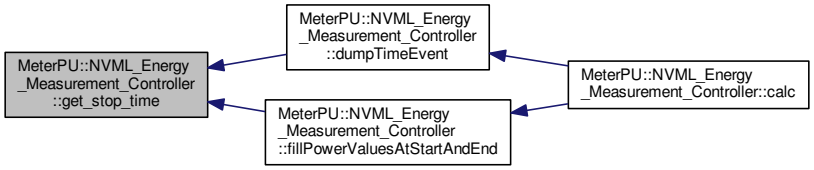
9.16.2.13 template<GPU_Device_Id_Type device_id> const CPU_Time::Time_Unit& MeterPU::NVML_Energy_Measurement_Controller< device_id >::get_start_time ( ) const [inline], [private]

Here is the caller graph for this function:



9.16.2.14 template<GPU_Device_Id_Type device_id> const CPU_Time::Time_Unit& MeterPU::NVML_Energy_Measurement_Controller< device_id >::get_stop_time ( ) const [inline], [private]

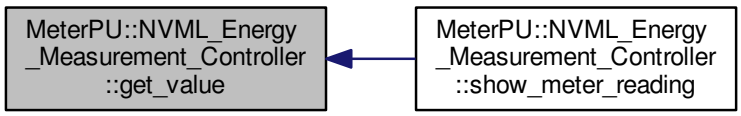
Here is the caller graph for this function:



9.16.2.15 template<GPU_Device_Id_Type device_id> NVML_Energy ::Energy_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::get_value ( ) const [inline]

Get calculated metric value, require calc() to be called already.

Here is the caller graph for this function:

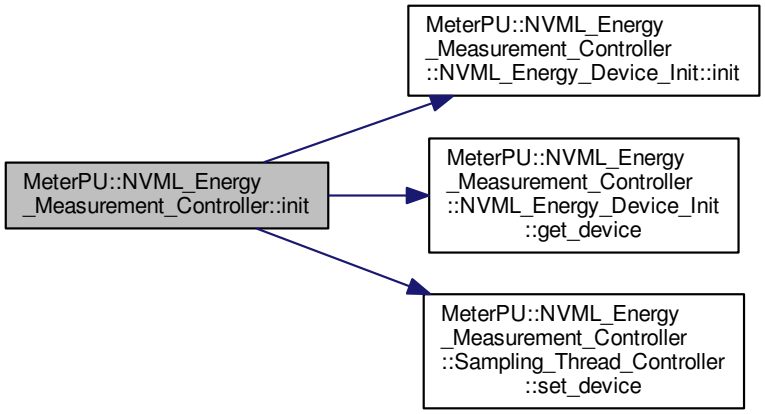


9.16.2.16 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::init ( ) [inline]

Initialize a GPU device.

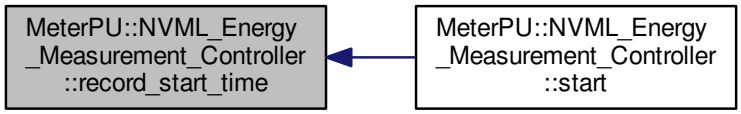
This function make if possible to initialize different CUDA GPUs if many exists.

Here is the call graph for this function:



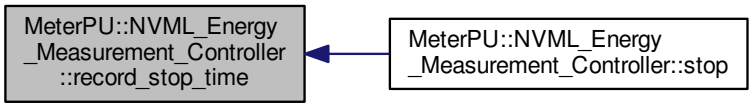
9.16.2.17 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::record_start_time ( ) [inline], [private]

Here is the caller graph for this function:



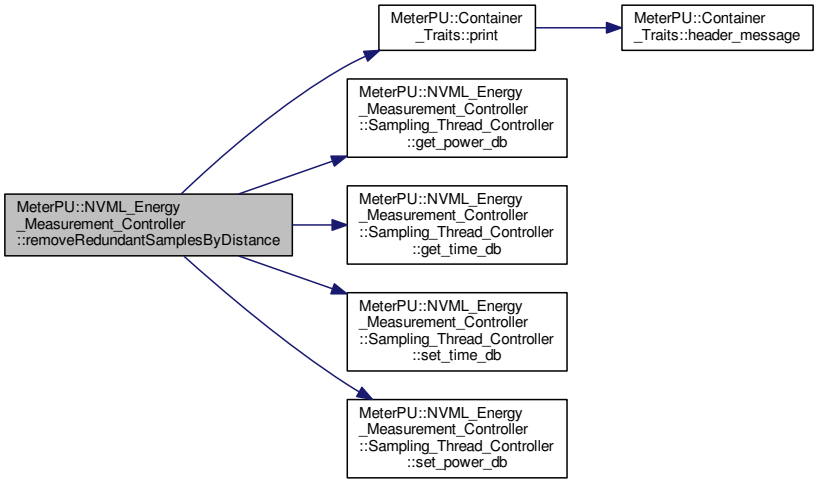
9.16.2.18 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::record_stop_time ( ) [inline], [private]

Here is the caller graph for this function:

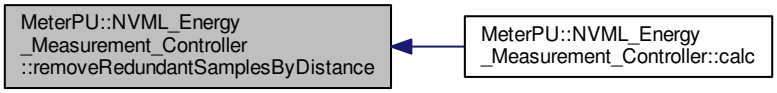


9.16.2.19 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::removeRedundantSamplesByDistance (const CPU_Time::ResultType & *time_distance_ms*) [inline],[private]

Here is the call graph for this function:

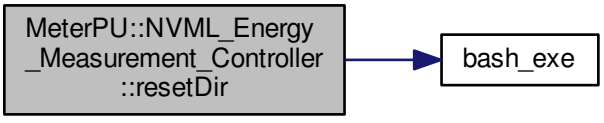


Here is the caller graph for this function:

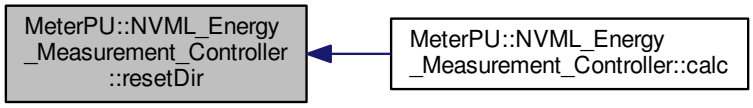


9.16.2.20 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::resetDir ( ) [inline],[private]

Here is the call graph for this function:



Here is the caller graph for this function:

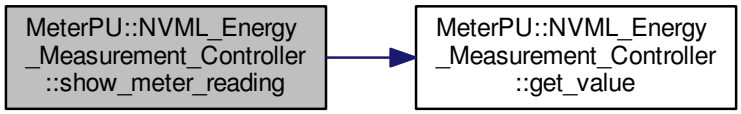


9.16.2.21 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::show_meter_reading ( ) const [inline], [virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:

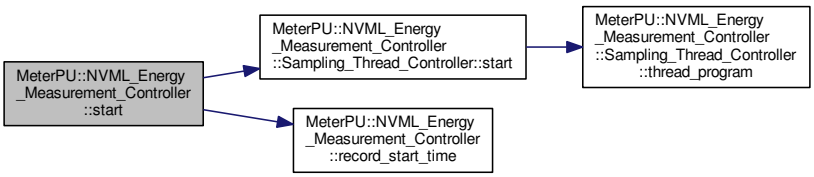


9.16.2.22 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::start ( ) [inline], [virtual]

mark the start of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:

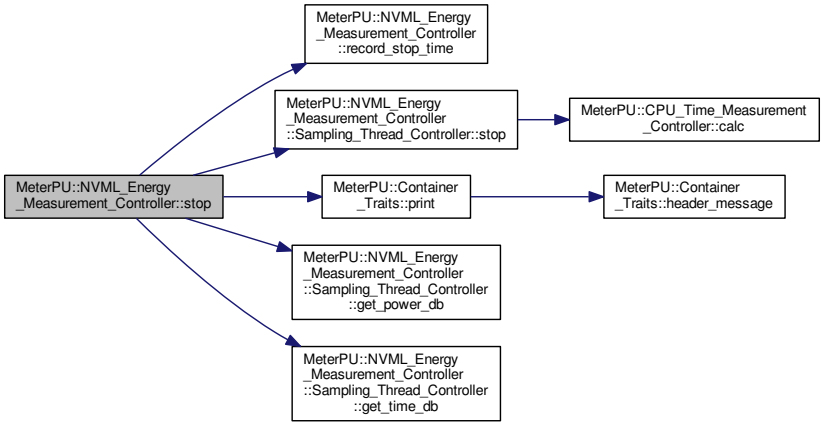


9.16.2.23 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::stop ( ) [inline], [virtual]

mark the end of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



9.16.3 Member Data Documentation

9.16.3.1 template<GPU_Device_Id_Type device_id> NVML_Energy ::Hp_Power_DB_Type MeterPU::NVML_Energy_Measurement_Controller< device_id >::correctedPowerDB [private]

9.16.3.2 template<GPU_Device_Id_Type device_id> nvmlDevice_t MeterPU::NVML_Energy_Measurement_Controller< device_id >::device [private]

9.16.3.3 template<GPU_Device_Id_Type device_id> NVML_Energy ::Energy_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::meter_reading [private]

9.16.3.4 template<GPU_Device_Id_Type device_id> NVML_Energy_Device_Init MeterPU::NVML_Energy_Measurement_Controller< device_id >::nvml_energy_device_init [private]

9.16.3.5 template<GPU_Device_Id_Type device_id> std::string MeterPU::NVML_Energy_Measurement_Controller< device_id >::path [private]

9.16.3.6 template<GPU_Device_Id_Type device_id> Sampling_Thread_Controller MeterPU::NVML_Energy_Measurement_Controller< device_id >::sampling_thread_controller [private]

9.16.3.7 template<GPU_Device_Id_Type device_id> CPU_Time::Time_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::start_time [private]

9.16.3.8 template<GPU_Device_Id_Type device_id> CPU_Time::Time_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::stop_time [private]

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

- MeterPU.h

9.17 MeterPU::NVML_Energy_Environment_Init::NVML_Manager Class Reference

NVML library init and teardown.

Static Public Member Functions

- static void init_NVML ()

Private Member Functions

- NVML_Manager ()
- ~NVML_Manager ()
- NVML_Manager (const NVML_Manager &)
- void operator= (const NVML_Manager &)
- void init ()
- void teardown ()

9.17.1 Detailed Description

NVML library init and teardown.

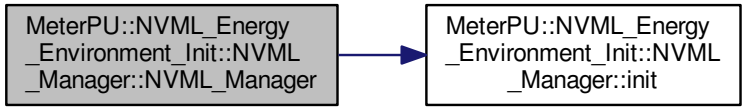
It contains methods for init and teardown of NVML library, the default constructor initialize the NVML library.

9.17.2 Constructor & Destructor Documentation

9.17.2.1 MeterPU::NVML_Energy_Environment_Init::NVML_Manager::NVML_Manager ( ) [inline], [private]

Default constructor, private, applications are forbidden to instantiate an object.

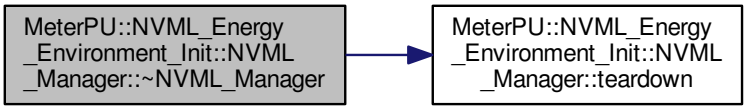
Here is the call graph for this function:



9.17.2.2 MeterPU::NVML_Energy_Environment_Init::NVML_Manager::~NVML_Manager ( ) [inline], [private]

Default constructor, private, if one object is created by calling init_NVML(), the destructor will be called automatically at the end of a program.

Here is the call graph for this function:



9.17.2.3 MeterPU::NVML_Energy_Environment_Init::NVML_Manager::NVML_Manager (const NVML_Manager &) [private]

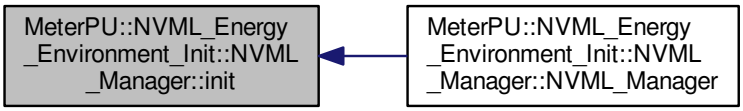
Copy constructor, override default one, and also private.

9.17.3 Member Function Documentation

9.17.3.1 void MeterPU::NVML_Energy_Environment_Init::NVML_Manager::init ( ) [inline], [private]

Initialize the NVML library.

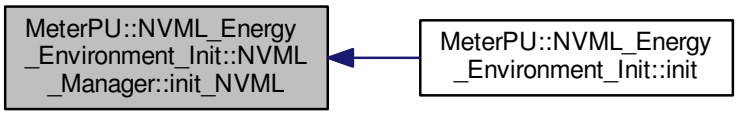
Here is the caller graph for this function:



9.17.3.2 static void MeterPU::NVML_Energy_Environment_Init::NVML_Manager::init_NVML ( ) [inline], [static]

Application only needs to call this function at startup, lazy initialization only when called.

Here is the caller graph for this function:



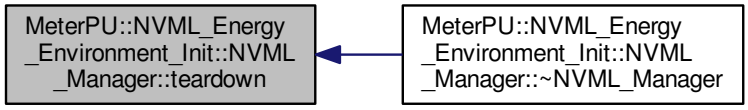
9.17.3.3 void MeterPU::NVML_Energy_Environment_Init::NVML_Manager::operator= (const NVML_Manager &) [inline], [private]

Assignment overloading, override default one, and also private.

9.17.3.4 void MeterPU::NVML_Energy_Environment_Init::NVML_Manager::teardown ( ) [inline], [private]

Shut down the NVML library.

Here is the caller graph for this function:



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

- MeterPU.h

9.18 MeterPU::PCM_Energy Struct Reference

PCM Energy Traits.

`#include <MeterPU.h>`

Public Types

- typedef PCM_Energy_Environment_Init Environment_Init_Type
- typedef double ResultType
- typedef PCM_Energy_Measurement_Controller Measurement_Controller
- typedef double CPU_Energy_Type
- typedef double DRAM_Energy_Type

9.18.1 Detailed Description

PCM Energy Traits.

Describe types relevant to CPU and DRAM Energy measurement.

9.18.2 Member Typedef Documentation

9.18.2.1 typedef double MeterPU::PCM_Energy::CPU_Energy_Type

9.18.2.2 typedef double MeterPU::PCM_Energy::DRAM_Energy_Type

9.18.2.3 typedef PCM_Energy_Environment_Init MeterPU::PCM_Energy::Environment_Init_Type

9.18.2.4 typedef PCM_Energy_Measurement_Controller MeterPU::PCM_Energy::Measurement_Controller

9.18.2.5 typedef double MeterPU::PCM_Energy::ResultType

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

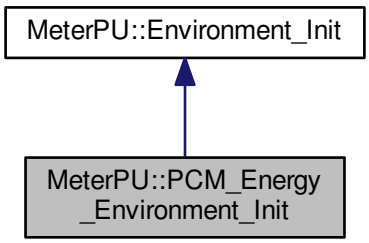
- MeterPU.h

9.19 MeterPU::PCM_Energy_Environment_Init Struct Reference

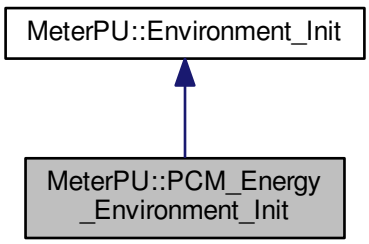
CPU Energy Library Initializer.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::PCM_Energy_Environment_Init:



Collaboration diagram for MeterPU::PCM_Energy_Environment_Init:



Public Member Functions

- void init ()

Private Member Functions

- DECLARE_CLASS_NAME ("PCM_Energy_Environment_Init")

9.19.1 Detailed Description

CPU Energy Library Initializer.

9.19.2 Member Function Documentation

9.19.2.1 MeterPU::PCM_Energy_Environment_Init::DECLARE_CLASS_NAME ("PCM_Energy_Environment_Init") [private]

9.19.2.2 void MeterPU::PCM_Energy_Environment_Init::init ( ) [inline], [virtual]

Implements MeterPU::Environment_Init.

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

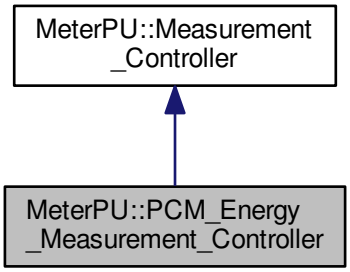
- MeterPU.h

9.20 MeterPU::PCM_Energy_Measurement_Controller Struct Reference

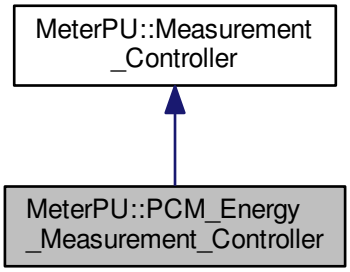
CPU and DRAME Energy Measurement Controller.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::PCM_Energy_Measurement_Controller:



Collaboration diagram for MeterPU::PCM_Energy_Measurement_Controller:



Public Member Functions

- void init ()
- void start ()

  mark the start of a measurement phase/period.
- void stop ()

  mark the end of a measurement phase/period.
- void calc ()

  calculate the metric value between start() and stop().
- PCM_Energy::ResultType const & get_value () const

  Get calculated metric value, require calc() to be called already.
- void show_meter_reading () const

  Print the calculated metric value to standard output, requires an invocation of calc() already done.
- PCM_Energy::ResultType const & get_cpu_energy () const
- PCM_Energy::ResultType const & get_dram_energy () const
- PCM_Energy_Measurement_Controller ()
- ~PCM_Energy_Measurement_Controller ()

Private Member Functions

- DECLARE_CLASS_NAME ("PCM_Energy_Measurement_Controller")
- void update_cpu_energy ()
- void update_dram_energy ()

Private Attributes

- PCM_Energy::ResultType meter_reading
- PCM_Energy::CPU_Energy_Type cpu_energy
- PCM_Energy::DRAM_Energy_Type dram_energy
- PCM * pcm
- SystemCounterState before_sstate
- SystemCounterState after_sstate

9.20.1 Detailed Description

CPU and DRAME Energy Measurement Controller.

It internally uses Intel PCM library.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 MeterPU::PCM_Energy_Measurement_Controller::PCM_Energy_Measurement_Controller ( ) [inline]

9.20.2.2 MeterPU::PCM_Energy_Measurement_Controller::~PCM_Energy_Measurement_Controller ( ) [inline]

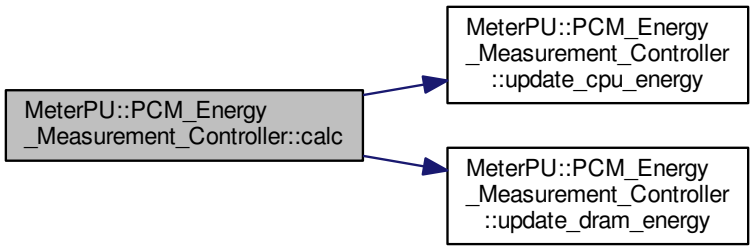
9.20.3 Member Function Documentation

9.20.3.1 void MeterPU::PCM_Energy_Measurement_Controller::calc ( ) [inline], [virtual]

calculate the metric value between start() and stop().

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



9.20.3.2 MeterPU::PCM_Energy_Measurement_Controller::DECLARE_CLASS_NAME ("PCM_Energy_Measurement_Controller") [private]

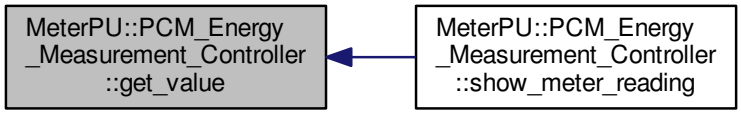
9.20.3.3 PCM_Energy::ResultType const& MeterPU::PCM_Energy_Measurement_Controller::get_cpu_energy ( ) const [inline]

9.20.3.4 PCM_Energy::ResultType const& MeterPU::PCM_Energy_Measurement_Controller::get_dram_energy ( ) const [inline]

9.20.3.5 PCM_Energy::ResultType const& MeterPU::PCM_Energy_Measurement_Controller::get_value ( ) const [inline]

Get calculated metric value, require calc() to be called already.

Here is the caller graph for this function:



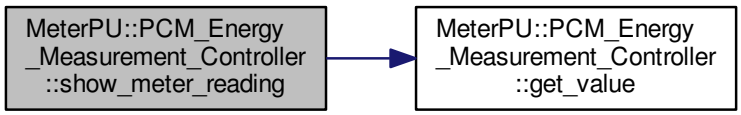
9.20.3.6 void MeterPU::PCM_Energy_Measurement_Controller::init ( ) [inline]

9.20.3.7 void MeterPU::PCM_Energy_Measurement_Controller::show_meter_reading ( ) const [inline], [virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



9.20.3.8 void MeterPU::PCM_Energy_Measurement_Controller::start ( ) [inline], [virtual]

mark the start of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

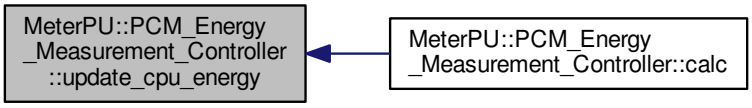
9.20.3.9 void MeterPU::PCM_Energy_Measurement_Controller::stop ( ) [inline], [virtual]

mark the end of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

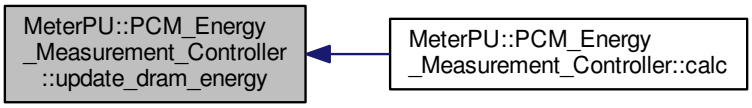
9.20.3.10 void MeterPU::PCM_Energy_Measurement_Controller::update_cpu_energy ( ) [inline], [private]

Here is the caller graph for this function:



9.20.3.11 void MeterPU::PCM_Energy_Measurement_Controller::update_dram_energy ( ) [inline], [private]

Here is the caller graph for this function:



9.20.4 Member Data Documentation

9.20.4.1 SystemCounterState MeterPU::PCM_Energy_Measurement_Controller::after_sstate [private]

9.20.4.2 SystemCounterState MeterPU::PCM_Energy_Measurement_Controller::before_sstate [private]

9.20.4.3 PCM_Energy::CPU_Energy_Type MeterPU::PCM_Energy_Measurement_Controller::cpu_energy [private]

9.20.4.4 PCM_Energy::DRAM_Energy_Type MeterPU::PCM_Energy_Measurement_Controller::dram_energy [private]

9.20.4.5 PCM_Energy::ResultType MeterPU::PCM_Energy_Measurement_Controller::meter_reading [private]

9.20.4.6 PCM* MeterPU::PCM_Energy_Measurement_Controller::pcm [private]

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

- MeterPU.h

9.21 MeterPU::Power_Vector_Container Struct Reference

`#include <MeterPU.h>`

Public Types

- typedef NVML_Energy::Power_DB_Type Data_Type
- typedef NVML_Energy::Power_DB_Const_Iterator_Type Const_Iterator_Type

Static Public Member Functions

- static std::string header_message ()

9.21.1 Member Typedef Documentation

9.21.1.1 typedef NVML_Energy ::Power_DB_Const_Iterator_Type MeterPU::Power_Vector_Container::Const_Iterator_Type

9.21.1.2 typedef NVML_Energy ::Power_DB_Type MeterPU::Power_Vector_Container::Data_Type

9.21.2 Member Function Documentation

9.21.2.1 static std::string MeterPU::Power_Vector_Container::header_message ( ) [inline], [static]

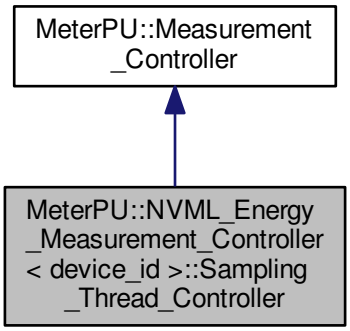
The documentation for this struct was generated from the following file:

- MeterPU.h

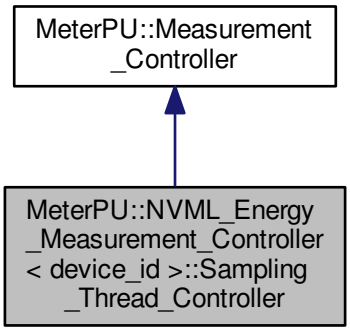
9.22 MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller Class Reference

Sampling Thread Controller.

Inheritance diagram for MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller:



Collaboration diagram for MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller:



Public Member Functions

- Sampling_Thread_Controller ()
- void start ()

  mark the start of a measurement phase/period.
- void stop ()

  mark the end of a measurement phase/period.
- void calc ()

  calculate the metric value between start() and stop().
- void show_meter_reading () const

  Print the calculated metric value to standard output, requires an invocation of calc() already done.
- void set_device (const nvmlDevice_t &device_handle)
- const NVML_Energy::Power_Unit & get_power () const
- const NVML_Energy::Power_DB_Type & get_power_db () const
- const NVML_Energy::Time_DB_Type & get_time_db () const
- NVML_Energy::Power_DB_Type & get_power_db_nonconst ()
- NVML_Energy::Time_DB_Type & get_time_db_nonconst ()
- void set_time_db (const NVML_Energy<>::Time_DB_Type &x)
- void set_power_db (const NVML_Energy<>::Power_DB_Type &x)
- NVML_Energy::Time_Unit const & get_stop_time ()
- NVML_Energy::Time_Unit const & get_start_time ()
- void reset_state ()

Private Member Functions

- DECLARE_CLASS_NAME ("Sampling_Thread_Controller")

Static Private Member Functions

- static NVML_Energy::Power_Unit get_UNREALISTIC_POWER_VALUE ()
- static void * thread_program (void *arg)

Private Attributes

- bool sampling
- nvmlDevice_t device

  Device handle.
- NVML_Energy::Power_Unit power

  Power value for each sample.
- NVML_Energy::Time_Unit time

  Time value for each sample.
- NVML_Energy::Time_DB_Type time_db

  Power data database.
- NVML_Energy::Power_DB_Type power_db
- pthread_t thread
- pthread_attr_t attr

9.22.1 Detailed Description

template<GPU_Device_Id_Type device_id>class MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller

Sampling Thread Controller.

Start a thread sampling power samples, and wait for stop signal.

9.22.2 Constructor & Destructor Documentation

9.22.2.1 template<GPU_Device_Id_Type device_id> MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::Sampling_Thread_Controller ( ) [inline]

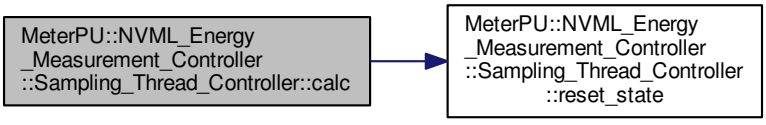
9.22.3 Member Function Documentation

9.22.3.1 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::calc ( ) [inline], [virtual]

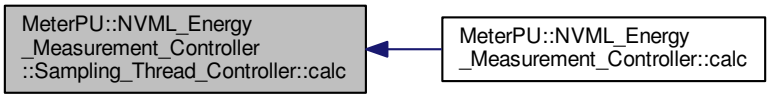
calculate the metric value between start() and stop().

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



Here is the caller graph for this function:

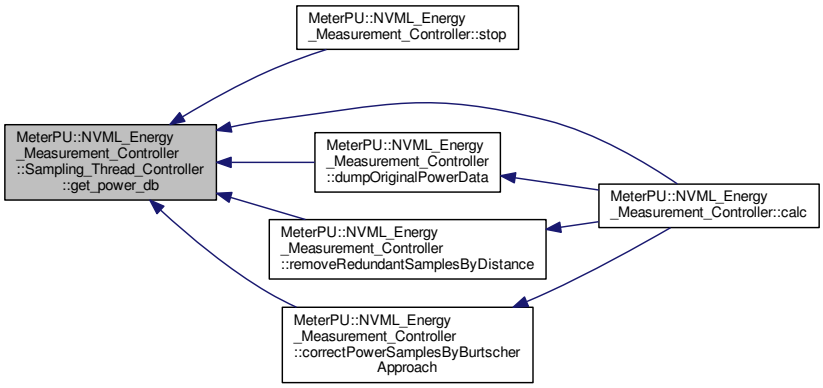


9.22.3.2 template<GPU_Device_Id_Type device_id> MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::DECLARE_CLASS_NAME ("Sampling_Thread_Controller") [private]

9.22.3.3 template<GPU_Device_Id_Type device_id> const NVML_Energy ::Power_Unit& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_power ( ) const [inline]

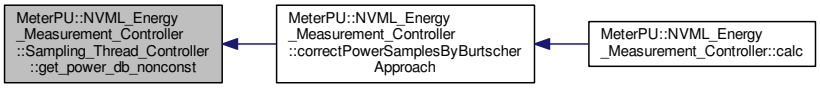
9.22.3.4 template<GPU_Device_Id_Type device_id> const NVML_Energy ::Power_DB_Type& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_power_db ( ) const [inline]

Here is the caller graph for this function:



9.22.3.5 template<GPU_Device_Id_Type device_id> NVML_Energy ::Power_DB_Type& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_power_db_nonconst ( ) [inline]

Here is the caller graph for this function:

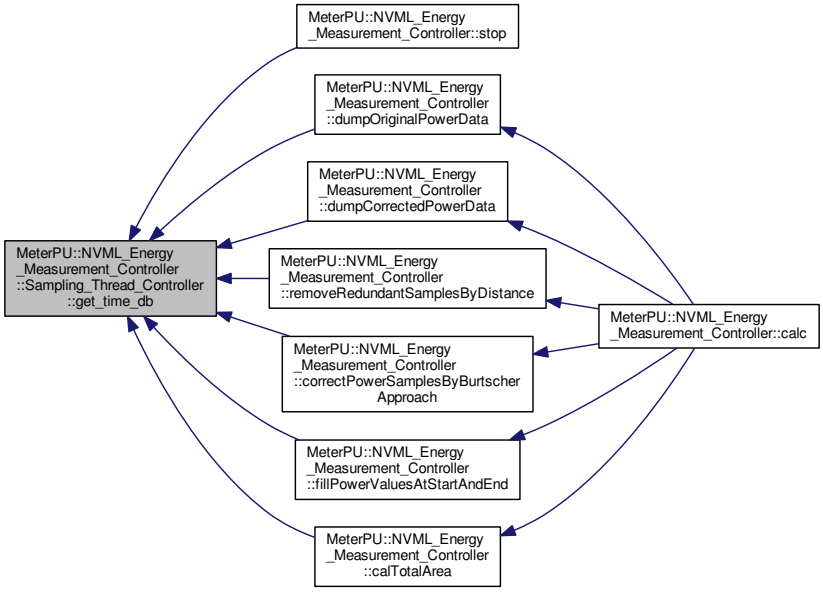


9.22.3.6 template<GPU_Device_Id_Type device_id> NVML_Energy ::Time_Unit const& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_start_time ( ) [inline]

9.22.3.7 template<GPU_Device_Id_Type device_id> NVML_Energy ::Time_Unit const& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_stop_time ( ) [inline]

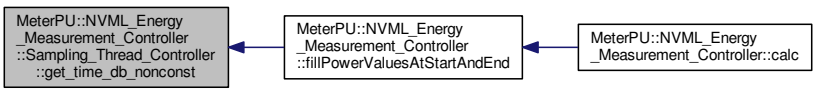
9.22.3.8 template<GPU_Device_Id_Type device_id> const NVML_Energy ::Time_DB_Type& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_time_db ( ) const [inline]

Here is the caller graph for this function:



9.22.3.9 template<GPU_Device_Id_Type device_id> NVML_Energy ::Time_DB_Type& MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::get_time_db_nonconst ( ) [inline]

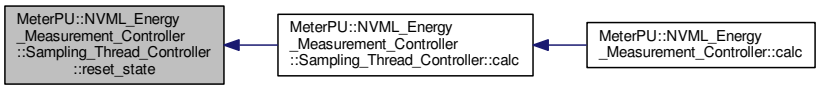
Here is the caller graph for this function:



9.22.3.10 template<GPU_Device_Id_Type device_id> static NVML_Energy ::Power_Unit MeterPU::NVML_Energy_-Measurement_Controller< device_id >::Sampling_Thread_Controller::get_UNREALISTIC_POWER_VALUE ( ) [inline],[static],[private]

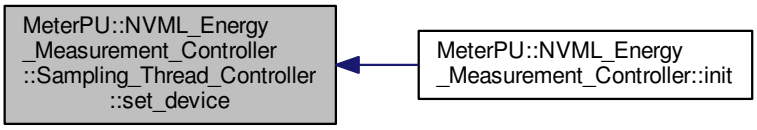
9.22.3.11 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::reset_state ( ) [inline]

Here is the caller graph for this function:



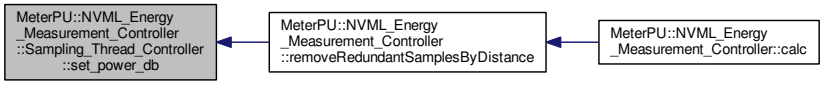
9.22.3.12 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::set_device (const nvmlDevice_t & *device_handle*) [inline]

Here is the caller graph for this function:



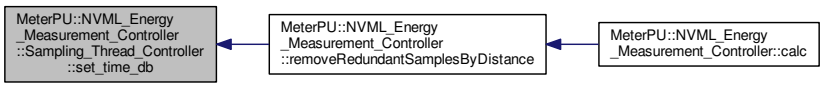
9.22.3.13 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::set_power_db (const NVML_Energy<>::Power_DB_Type & *x*) [inline]

Here is the caller graph for this function:



9.22.3.14 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::set_time_db (const NVML_Energy<>::Time_DB_Type & *x*) [inline]

Here is the caller graph for this function:



9.22.3.15 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::show_meter_reading ( ) const [inline], [virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

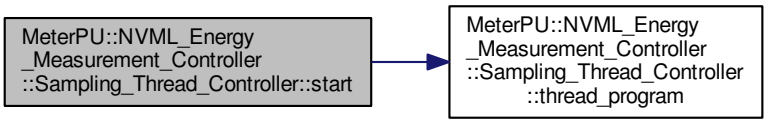
Implements MeterPU::Measurement_Controller.

9.22.3.16 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::start ( ) [inline], [virtual]

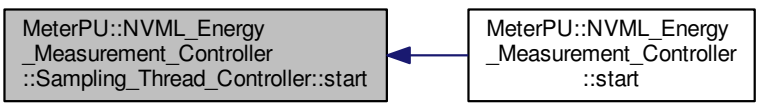
mark the start of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



Here is the caller graph for this function:

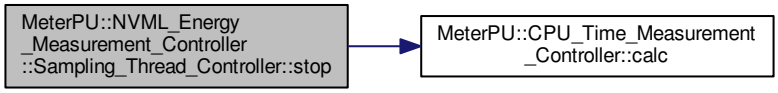


9.22.3.17 template<GPU_Device_Id_Type device_id> void MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::stop ( ) [inline], [virtual]

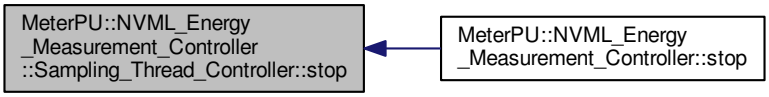
mark the end of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:

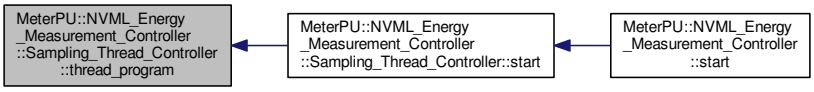


Here is the caller graph for this function:



9.22.3.18 template<GPU_Device_Id_Type device_id> static void* MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::thread_program (void * *arg*) [inline], [static], [private]

Here is the caller graph for this function:



9.22.4 Member Data Documentation

9.22.4.1 template<GPU_Device_Id_Type device_id> pthread_attr_t MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::attr [private]

9.22.4.2 template<GPU_Device_Id_Type device_id> nvmlDevice_t MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::device [private]

Device handle.

9.22.4.3 template<GPU_Device_Id_Type device_id> NVML_Energy ::Power_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::power [private]

Power value for each sample.

9.22.4.4 template<GPU_Device_Id_Type device_id> NVML_Energy ::Power_DB_Type MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::power_db [private]

9.22.4.5 template<GPU_Device_Id_Type device_id> bool MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::sampling [private]

9.22.4.6 template<GPU_Device_Id_Type device_id> pthread_t MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::thread [private]

9.22.4.7 template<GPU_Device_Id_Type device_id> NVML_Energy ::Time_Unit MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::time [private]

Time value for each sample.

9.22.4.8 template<GPU_Device_Id_Type device_id> NVML_Energy ::Time_DB_Type MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller::time_db [private]

Power data database.

Store time and power value pairs.

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

- MeterPU.h

9.23 MeterPU::System_Energy< gpu_ids > Struct Template Reference

System Energy Traits.

`#include <MeterPU.h>`

Public Types

- typedef System_Energy_Environment_Init Environment_Init_Type
- typedef System_Energy_Measurement_Controller < gpu_ids...> Measurement_Controller
- typedef double ResultType

9.23.1 Detailed Description

template<GPU_Device_Id_Type... gpu_ids>struct MeterPU::System_Energy< gpu_ids >

System Energy Traits.

Describe types relevant to energy measurement. System Energy Meter can be specified as a combination of all CPUs and some GPUs.

9.23.2 Member Typedef Documentation

9.23.2.1 template<GPU_Device_Id_Type... gpu_ids> typedef System_Energy_Environment_Init MeterPU::System_Energy< gpu_ids >::Environment_Init_Type

9.23.2.2 template<GPU_Device_Id_Type... gpu_ids> typedef System_Energy_Measurement_Controller<gpu_ids...> MeterPU::System_Energy< gpu_ids >::Measurement_Controller

9.23.2.3 template<GPU_Device_Id_Type... gpu_ids> typedef double MeterPU::System_Energy< gpu_ids >::ResultType

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

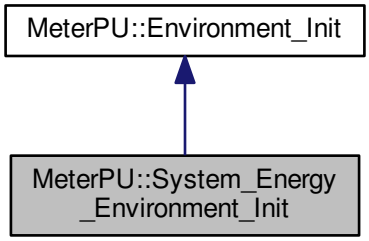
- MeterPU.h

9.24 MeterPU::System_Energy_Environment_Init Struct Reference

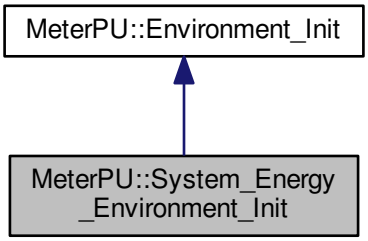
System Energy Library Initializer.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::System_Energy_Environment_Init:



Collaboration diagram for MeterPU::System_Energy_Environment_Init:



Public Member Functions

- void init ()

Private Member Functions

- DECLARE_CLASS_NAME ("PCM_Energy_Environment_Init")

9.24.1 Detailed Description

System Energy Library Initializer.

9.24.2 Member Function Documentation

9.24.2.1 MeterPU::System_Energy_Environment_Init::DECLARE_CLASS_NAME ("PCM_Energy_Environment_Init") [private]

9.24.2.2 void MeterPU::System_Energy_Environment_Init::init ( ) [inline], [virtual]

Implements MeterPU::Environment_Init.

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

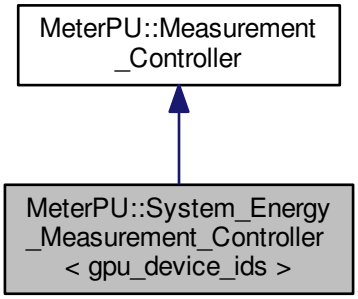
- MeterPU.h

9.25 MeterPU::System_Energy_Measurement_Controller< gpu_device_ids > Struct Template Reference

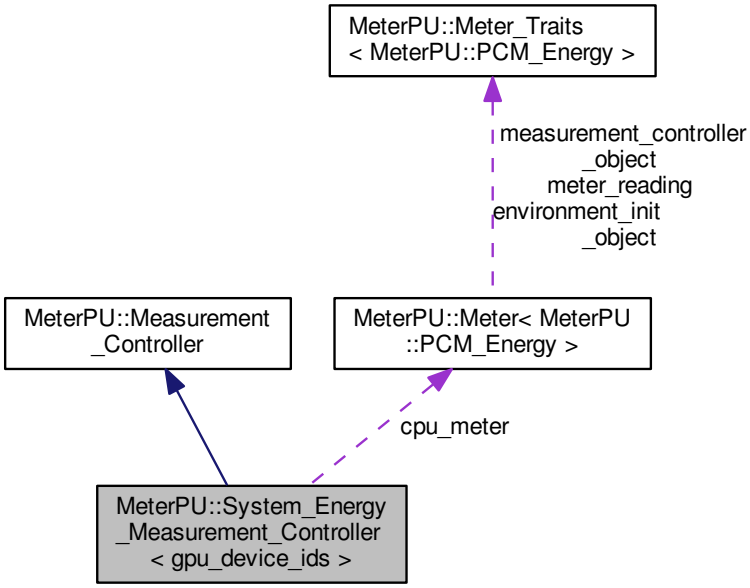
System Energy Measurement Controller.

`#include <MeterPU.h>`

Inheritance diagram for MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >:



Collaboration diagram for MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >:



Public Member Functions

- System_Energy_Measurement_Controller ()
- ~System_Energy_Measurement_Controller ()
- void init ()
- void start ()

  mark the start of a measurement phase/period.
- void stop ()

mark the end of a measurement phase/period.

- void calc ()

  calculate the metric value between start() and stop().

- System_Energy::ResultType const & get_value () const

  Get calculated metric value, require calc() to be called already.

- void show_meter_reading () const

  Print the calculated metric value to standard output, requires an invocation of calc() already done.

Private Types

- enum { N =sizeof...(gpu_device_ids) }

Private Attributes

- Meter< PCM_Energy > cpu_meter
- void * gpu_meters [N]
- NVML_Energy::ResultType meter_reading

9.25.1 Detailed Description

template<GPU_Device_Id_Type... gpu_device_ids>struct MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >

System Energy Measurement Controller.

It initialize a Energy Meter of CPU and DRAM, together with a specified combination of GPU Energy Meters, thus it can measure energy consumption of those hardware components together.

9.25.2 Member Enumeration Documentation

9.25.2.1 template<GPU_Device_Id_Type... gpu_device_ids> anonymous enum [private]

Enumerator

N

9.25.3 Constructor & Destructor Documentation

9.25.3.1 template<GPU_Device_Id_Type... gpu_device_ids> MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::System_Energy_Measurement_Controller ( ) [inline]

9.25.3.2 template<GPU_Device_Id_Type... gpu_device_ids> MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::~System_Energy_Measurement_Controller ( ) [inline]

9.25.4 Member Function Documentation

9.25.4.1 template<GPU_Device_Id_Type... gpu_device_ids> void MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::calc ( ) [inline], [virtual]

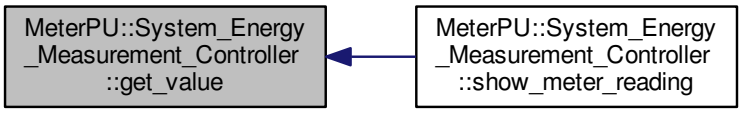
calculate the metric value between start() and stop().

Implements MeterPU::Measurement_Controller.

9.25.4.2 template<GPU_Device_Id_Type... gpu_device_ids> System_Energy ::ResultType const& MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::get_value ( ) const [inline]

Get calculated metric value, require calc() to be called already.

Here is the caller graph for this function:



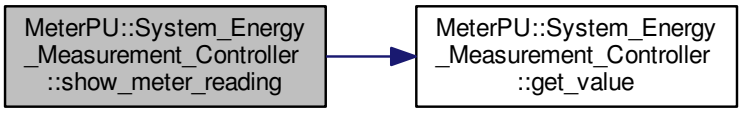
9.25.4.3 template<GPU_Device_Id_Type... gpu_device_ids> void MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::init ( ) [inline]

9.25.4.4 template<GPU_Device_Id_Type... gpu_device_ids> void MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::show_meter_reading ( ) const [inline], [virtual]

Print the calculated metric value to standard output, requires an invocation of calc() already done.

Implements MeterPU::Measurement_Controller.

Here is the call graph for this function:



9.25.4.5 template<GPU_Device_Id_Type... gpu_device_ids> void MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::start ( ) [inline], [virtual]

mark the start of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

9.25.4.6 template<GPU_Device_Id_Type... gpu_device_ids> void MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::stop ( ) [inline], [virtual]

mark the end of a measurement phase/period.

Implements MeterPU::Measurement_Controller.

9.25.5 Member Data Documentation

9.25.5.1 template<GPU_Device_Id_Type... gpu_device_ids> Meter<PCM_Energy> MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::cpu_meter [private]

9.25.5.2 template<GPU_Device_Id_Type... gpu_device_ids> void* MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::gpu_meters[N] [private]

9.25.5.3 template<GPU_Device_Id_Type... gpu_device_ids> NVML_Energy ::ResultType MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >::meter_reading [private]

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

- MeterPU.h

9.26 MeterPU::Time_Vector_Container Struct Reference

`#include <MeterPU.h>`

Public Types

- typedef NVML_Energy::Time_DB_Type Data_Type
- typedef NVML_Energy::Time_DB_Const_Iterator_Type Const_Iterator_Type

Static Public Member Functions

- static std::string header_message ()

9.26.1 Member Typedef Documentation

9.26.1.1 typedef NVML_Energy ::Time_DB_Const_Iterator_Type MeterPU::Time_Vector_Container::Const_Iterator_Type

9.26.1.2 typedef NVML_Energy ::Time_DB_Type MeterPU::Time_Vector_Container::Data_Type

9.26.2 Member Function Documentation

9.26.2.1 static std::string MeterPU::Time_Vector_Container::header_message ( ) [inline], [static]

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

- MeterPU.h

Chapter 10

File Documentation

10.1 examples/4.GpuEnergyMeasurement-2/cuda_call.h File Reference

Functions

- __global__ void hello (char *a, int *b, double *d)

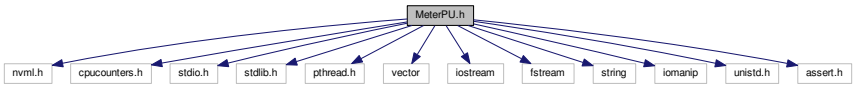
10.1.1 Function Documentation

10.1.1.1 __global__ void hello (char * *a*, int * *b*, double * *d*)

10.2 MeterPU.h File Reference

```
#include <nvml.h>
#include <cpucounters.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <unistd.h>
#include <assert.h>
```

Include dependency graph for MeterPU.h:



Classes

- struct MeterPU::Meter_Traits< Type >

  Traits Interface.
- struct MeterPU::NVML_Energy< device_id >

GPU Energy Traits.

- struct MeterPU::NVML_Energy_Measurement_Controller< device_id >

  CUDA-enabled GPU Energy Measurement Controller.
- struct MeterPU::System_Energy< gpu_ids >

  System Energy Traits.
- struct MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >

  System Energy Measurement Controller.
- class MeterPU::Meter< Type >

  The software multi-meters.
- struct MeterPU::Meter_Traits< Type >

  Traits Interface.
- struct MeterPU::CPU_Time

  Time Traits.
- struct MeterPU::CUDA_Time

  Cuda Timer Traits.
- struct MeterPU::PCM_Energy

  PCM Energy Traits.
- struct MeterPU::NVML_Energy< device_id >

  GPU Energy Traits.
- struct MeterPU::System_Energy< gpu_ids >

  System Energy Traits.
- struct MeterPU::Container_Traits< Type >
- struct MeterPU::Power_Vector_Container
- struct MeterPU::Hp_Power_Vector_Container
- struct MeterPU::Time_Vector_Container
- struct MeterPU::Environment_Init

  Library Initializer Interface.
- struct MeterPU::CPU_Time_Environment_Init

  CPU Time Library Initializer.
- struct MeterPU::CUDA_Time_Environment_Init

  GPU Time Library Initializer.
- struct MeterPU::PCM_Energy_Environment_Init

  CPU Energy Library Initializer.
- struct MeterPU::NVML_Energy_Environment_Init

  GPU Energy Library Initializer.
- class MeterPU::NVML_Energy_Environment_Init::NVML_Manager

  NVML library init and teardown.
- struct MeterPU::System_Energy_Environment_Init

  System Energy Library Initializer.
- struct MeterPU::Measurement_Controller

  Measurement Controller Interface.
- struct MeterPU::CPU_Time_Measurement_Controller

  CPU Time Measurement Controller.
- struct MeterPU::CUDA_Time_Measurement_Controller

  CUDA-enabled GPU Time Measurement Controller.
- struct MeterPU::PCM_Energy_Measurement_Controller

  CPU and DRAME Energy Measurement Controller.
- struct MeterPU::NVML_Energy_Measurement_Controller< device_id >

  CUDA-enabled GPU Energy Measurement Controller.
- struct MeterPU::NVML_Energy_Measurement_Controller< device_id >::NVML_Energy_Device_Init< device_id_value >

- class MeterPU::NVML_Energy_Measurement_Controller< device_id >::Sampling_Thread_Controller

  Sampling Thread Controller.
- struct MeterPU::System_Energy_Measurement_Controller< gpu_device_ids >

  System Energy Measurement Controller.

Namespaces

- MeterPU

Macros

- #define PRINT_FUNC_NAME(message) std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<__func__<<"():] "<<message<<std::endl
- #define PRINT_FUNC_NAME_CONT std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<__func__<<"():] "
- #define DECLARE_CLASS_NAME(name) static std::string class_name(){return name;}
- #define PRINT_CLASS_FUNC_NAME(message) std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<class_name()<<" "<<__func__<<"():] "<<message<<std::endl
- #define PRINT_CLASS_FUNC_NAME_CONT std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<class_name()<<" "<<__func__<<"():] "
- #define METERPU_TIME_MEASURE(x) clock_gettime(CLOCK_MONOTONIC, x)
- #define POINTER ((Meter<NVML_Energy<first_device_id> >*)x[array_index])

  A macro for downcast a void pointer to a NVML Energy Meter.
- #define NOTHING
- #define LOOPER(name, return_type, base_code, general_code)

  A macro to build variadic template to recursively apply code snippets.

Typedefs

- typedef unsigned int MeterPU::GPU_Device_Id_Type

Functions

- bool bash_exe (const std::string &cmd, std::vector< std::string > &out)
- CPU_Time::ResultType MeterPU::operator- (CPU_Time::Time_Unit const &stop_time, CPU_Time::Time_Unit const &start_time)

  Calculate elapsed time between two time stamp.
- bool MeterPU::operator< (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)

  Check if a time stamp is earlier than another.
- bool MeterPU::operator== (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)

  Check if two time stamps are the same.
- bool MeterPU::operator<= (CPU_Time::Time_Unit const &small_time, CPU_Time::Time_Unit const &large_time)

  Check if a time stamp is earlier or equal to another.
- std::ostream & MeterPU::operator<< (std::ostream &out, NVML_Energy<>::Time_Unit ts)

10.2.1 Macro Definition Documentation

10.2.1.1 #define DECLARE_CLASS_NAME(*name*) static std::string class_name(){return name;}

10.2.1.2 #define METERPU_TIME_MEASURE(*x*) clock_gettime(CLOCK_MONOTONIC, x)

10.2.1.3 #define PRINT_CLASS_FUNC_NAME(*message*) std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<class_name()<<" "<<__func__<<"():] "<<message<<std::endl

10.2.1.4 #define PRINT_CLASS_FUNC_NAME_CONT std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<class_name()<<" "<<__func__<<"():] "

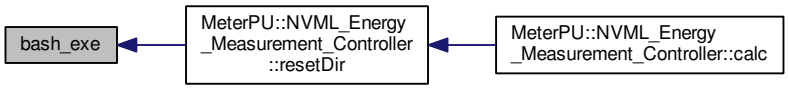
10.2.1.5 #define PRINT_FUNC_NAME(*message*) std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<__func__<<"():] "<<message<<std::endl

10.2.1.6 #define PRINT_FUNC_NAME_CONT std::cout<<"["<<__FILE__<<"#"<<__LINE__<<" "<<__func__<<"():] "

10.2.2 Function Documentation

10.2.2.1 bool bash_exe (const std::string & *cmd,* std::vector< std::string > & *out*)

Here is the caller graph for this function:



Index