M3P SCoPE XML Plug-in

1.0.0

Generated by Doxygen 1.5.5

Wed Dec 17 19:05:24 2008

Contents

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

2 Data Structure Index

2.1 Class Hierarchy

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

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 expatmm Namespace Reference

Data Structures

- class ExpatXMLParser

Functions

- std::string getExpatMMVersion (void)

5.1.1 Function Documentation

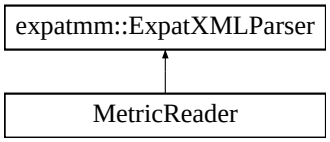
5.1.1.1 std::string EXPATMM_LIBEXPORT expatmm::getExpatMMVersion (void)

6 Data Structure Documentation

6.1 expatmm::ExpatXMLParser Class Reference

`#include <ExpatXMLParser.h>`

Inheritance diagram for expatmm::ExpatXMLParser::



Public Member Functions

- ExpatXMLParser (void)
- ExpatXMLParser (size_t chunk_size)
- virtual ~ExpatXMLParser (void)
- virtual bool Parse (void)
- virtual bool Ready (void)
- virtual XML_Error getLastError (void)
- virtual XML_Status getStatus (void)
- ExpatXMLParser (void)
- ExpatXMLParser (size_t chunk_size)
- virtual ~ExpatXMLParser (void)
- virtual bool Parse (void)
- virtual bool Ready (void)
- virtual XML_Error getLastError (void)
- virtual XML_Status getStatus (void)

Protected Member Functions

- virtual XML_Char * getBuffer (void)
- virtual size_t getBlockSize (void)
- virtual ssize_t read_block (void)
- virtual void setReadiness (bool ready)
- virtual void setStatus (XML_Status new_status)
- virtual void setLastError (XML_Error new_last_error)
- virtual void StartElement (const XML_Char *name, const XML_Char **atts)
- virtual void EndElement (const XML_Char *name)
- virtual void CharacterData (const XML_Char *s, int len)
- virtual void ProcessingInstruction (const XML_Char *target, const XML_Char *data)
- virtual void CommentData (const XML_Char *data)
- virtual void DefaultHandler (const XML_Char *s, int len)
- virtual void CDataStart (void)
- virtual void CDataEnd (void)
- virtual XML_Char * getBuffer (void)
- virtual size_t getBlockSize (void)
- virtual ssize_t read_block (void)
- virtual void setReadiness (bool ready)
- virtual void setStatus (XML_Status new_status)
- virtual void setLastError (XML_Error new_last_error)
- virtual void StartElement (const XML_Char *name, const XML_Char **atts)
- virtual void EndElement (const XML_Char *name)
- virtual void CharacterData (const XML_Char *s, int len)
- virtual void ProcessingInstruction (const XML_Char *target, const XML_Char *data)
- virtual void CommentData (const XML_Char *data)
- virtual void DefaultHandler (const XML_Char *s, int len)
- virtual void CDataStart (void)
- virtual void CDataEnd (void)

6.1.1 Constructor & Destructor Documentation

6.1.1.1 ExpatXMLParser::ExpatXMLParser (void)

6.1.1.2 ExpatXMLParser::ExpatXMLParser (size_t *chunk_size*)

6.1.1.3 ExpatXMLParser::~ExpatXMLParser (void) [virtual]

6.1.1.4 expatmm::ExpatXMLParser::ExpatXMLParser (void)

6.1.1.5 expatmm::ExpatXMLParser::ExpatXMLParser (size_t *chunk_size*)

6.1.1.6 virtual expatmm::ExpatXMLParser::~ExpatXMLParser (void) [virtual]

6.1.2 Member Function Documentation

6.1.2.1 virtual XML_Char* expatmm::ExpatXMLParser::getBuffer (void) [inline, protected, virtual]

6.1.2.2 virtual size_t expatmm::ExpatXMLParser::getBlockSize (void) [inline, protected, virtual]

6.1.2.3 ssize_t ExpatXMLParser::read_block (void) [protected, virtual]

Reimplemented in MetricReader.

6.1.2.4 virtual void expatmm::ExpatXMLParser::setReadiness (bool *ready*) [inline, protected, virtual]

6.1.2.5 virtual void expatmm::ExpatXMLParser::setStatus (XML_Status *new_status*) [inline, protected, virtual]

6.1.2.6 virtual void expatmm::ExpatXMLParser::setLastError (XML_Error *new_last_error*) [inline, protected, virtual]

6.1.2.7 void ExpatXMLParser::StartElement (const XML_Char * *name*, const XML_Char ** *atts*) [protected, virtual]

Reimplemented in MetricReader.

6.1.2.8 void ExpatXMLParser::EndElement (const XML_Char * *name*) [protected, virtual]

Reimplemented in MetricReader.

6.1.2.9 void ExpatXMLParser::CharacterData (const XML_Char * *s*, int *len*) [protected, virtual]

6.1.2.10 void ExpatXMLParser::ProcessingInstruction (const XML_Char * *target*, const XML_Char * *data*) [protected, virtual]

6.1.2.11 void ExpatXMLParser::CommentData (const XML_Char * *data*) [protected, virtual]

6.1.2.12 void ExpatXMLParser::DefaultHandler (const XML_Char * *s*, int *len*) [protected, virtual]

6.1.2.13 void ExpatXMLParser::CDataStart (void) [protected, virtual]

6.1.2.14 void ExpatXMLParser::CDataEnd (void) [protected, virtual]

6.1.2.15 bool ExpatXMLParser::Parse (void) [virtual]

6.1.2.16 virtual bool expatmm::ExpatXMLParser::Ready (void) [inline, virtual]

6.1.2.17 virtual XML_Error expatmm::ExpatXMLParser::getLastError (void) [inline, virtual]

6.1.2.18 virtual XML_Status expatmm::ExpatXMLParser::getStatus (void) [inline, virtual]

6.1.2.19 virtual XML_Char* expatmm::ExpatXMLParser::getBuffer (void) [inline, protected, virtual]

6.1.2.20 virtual size_t expatmm::ExpatXMLParser::getBlockSize (void) [inline, protected, virtual]

6.1.2.21 virtual ssize_t expatmm::ExpatXMLParser::read_block (void) [protected, virtual]

Reimplemented in MetricReader.

6.1.2.22 virtual void expatmm::ExpatXMLParser::setReadiness (bool *ready*) [inline, protected, virtual]

6.1.2.23 virtual void expatmm::ExpatXMLParser::setStatus (XML_Status *new_status*) [inline, protected, virtual]

6.1.2.24 virtual void expatmm::ExpatXMLParser::setLastError (XML_Error *new_last_error*) [inline, protected, virtual]

6.1.2.25 virtual void expatmm::ExpatXMLParser::StartElement (const XML_Char * *name*, const XML_Char ** *atts*) [protected, virtual]

Reimplemented in MetricReader.

6.1.2.26 virtual void expatmm::ExpatXMLParser::EndElement (const XML_Char * *name*) [protected, virtual]

Reimplemented in MetricReader.

6.1.2.27 virtual void expatmm::ExpatXMLParser::CharacterData (const XML_Char * *s*, int *len*) [protected, virtual]

6.1.2.28 virtual void expatmm::ExpatXMLParser::ProcessingInstruction (const XML_Char * *target*, const XML_Char * *data*) [protected, virtual]

6.1.2.29 virtual void expatmm::ExpatXMLParser::CommentData (const XML_Char * *data*) [protected, virtual]

6.1.2.30 virtual void expatmm::ExpatXMLParser::DefaultHandler (const XML_Char * *s*, int *len*) [protected, virtual]

6.1.2.31 virtual void expatmm::ExpatXMLParser::CDataStart (void) [protected, virtual]

6.1.2.32 virtual void expatmm::ExpatXMLParser::CDataEnd (void) [protected, virtual]

6.1.2.33 virtual bool expatmm::ExpatXMLParser::Parse (void) [virtual]

6.1.2.34 virtual bool expatmm::ExpatXMLParser::Ready (void) [inline, virtual]

6.1.2.35 virtual XML_Error expatmm::ExpatXMLParser::getLastError (void) [inline, virtual]

6.1.2.36 virtual XML_Status expatmm::ExpatXMLParser::getStatus (void) [inline, virtual]

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

- src/ExpatXMLParser.h
- expatmm/ExpatXMLParser.h
- ExpatXMLParser.cpp

6.2 hw_component Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- char * type
- char * class_name
- char * activation_type
- char * freq
- char * mem_size
- char * type_specific_1
- char * type_specific_2
- char * area
- char * mean_power
- char * read_energy
- char * read_size_energy

- char * write_energy
- char * write_size_energy
- char * latency
- char * width

6.2.1 Field Documentation

6.2.1.1 char* hw_component::name

6.2.1.2 char* hw_component::path

6.2.1.3 char* hw_component::type

6.2.1.4 char* hw_component::class_name

6.2.1.5 char* hw_component::activation_type

6.2.1.6 char* hw_component::freq

6.2.1.7 char* hw_component::mem_size

6.2.1.8 char* hw_component::type_specific_1

6.2.1.9 char* hw_component::type_specific_2

6.2.1.10 char* hw_component::area

6.2.1.11 char* hw_component::mean_power

6.2.1.12 char* hw_component::read_energy

6.2.1.13 char* hw_component::read_size_energy

6.2.1.14 char* hw_component::write_energy

6.2.1.15 char* hw_component::write_size_energy

6.2.1.16 char* hw_component::latency

6.2.1.17 char* hw_component::width

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

- xml_if.h

6.3 hw_connection Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- char * activation_type
- char * speed
- char * latency
- char * mem_size
- char * type_specific_1
- char * type_specific_2
- char * start_addr
- char * port
- char * irq
- char * rec_irq
- char * local_id
- char * instance_name
- struct hw_instance * instance
- int offset

6.3.1 Field Documentation

6.3.1.1 char* hw_connection::name

6.3.1.2 char* hw_connection::path

6.3.1.3 char* hw_connection::activation_type

6.3.1.4 char* hw_connection::speed

6.3.1.5 char* hw_connection::latency

6.3.1.6 char* hw_connection::mem_size

6.3.1.7 char* hw_connection::type_specific_1

6.3.1.8 char* hw_connection::type_specific_2

6.3.1.9 char* hw_connection::start_addr

6.3.1.10 char* hw_connection::port

6.3.1.11 char* hw_connection::irq

6.3.1.12 char* hw_connection::rec_irq

6.3.1.13 char* hw_connection::local_id

6.3.1.14 char* hw_connection::instance_name

6.3.1.15 struct hw_instance* hw_connection::instance [read]

6.3.1.16 int hw_connection::offset

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

- xml_if.h

6.4 hw_group Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- struct list_elem * hw_names
- struct list_elem * list

6.4.1 Field Documentation

6.4.1.1 char* hw_group::name

6.4.1.2 char* hw_group::path

6.4.1.3 struct list_elem* hw_group::hw_names [read]

6.4.1.4 struct list_elem* hw_group::list [read]

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

- xml_if.h

6.5 hw_instance Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- char * type
- char * activation_type
- char * class_name
- char * latency
- char * freq
- char * mem_size
- char * type_specific_1
- char * type_specific_2
- char * start_addr
- char * irq
- char * local_id
- char * area
- char * mean_power
- char * read_energy
- char * read_size_energy
- char * write_energy
- char * write_size_energy
- char * width
- char * component_name
- struct hw_component * component
- int offset
- struct list_elem * contain
- struct list_elem * connections
- void * scope_data

6.5.1 Field Documentation

6.5.1.1 char* hw_instance::name

6.5.1.2 char* hw_instance::path

6.5.1.3 char* hw_instance::type

6.5.1.4 char* hw_instance::activation_type

6.5.1.5 char* hw_instance::class_name

6.5.1.6 char* hw_instance::latency

6.5.1.7 char* hw_instance::freq

6.5.1.8 char* hw_instance::mem_size

6.5.1.9 char* hw_instance::type_specific_1

6.5.1.10 char* hw_instance::type_specific_2

6.5.1.11 char* hw_instance::start_addr

6.5.1.12 char* hw_instance::irq

6.5.1.13 char* hw_instance::local_id

6.5.1.14 char* hw_instance::area

6.5.1.15 char* hw_instance::mean_power

6.5.1.16 char* hw_instance::read_energy

6.5.1.17 char* hw_instance::read_size_energy

6.5.1.18 char* hw_instance::write_energy

6.5.1.19 char* hw_instance::write_size_energy

6.5.1.20 char* hw_instance::width

6.5.1.21 char* hw_instance::component_name

6.5.1.22 struct hw_component* hw_instance::component [read]

6.5.1.23 int hw_instance::offset

6.5.1.24 struct list_elem* hw_instance::contain [read]

6.5.1.25 struct list_elem* hw_instance::connections [read]

6.5.1.26 void* hw_instance::scope_data

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

- xml_if.h

6.6 list_elem Struct Reference

`#include <xml_list.h>`

Data Fields

- struct list_elem * next
- void * elem

6.6.1 Field Documentation

6.6.1.1 struct list_elem* list_elem::next [read]

6.6.1.2 void* list_elem::elem

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

- xml_list.h

6.7 Metric Class Reference

Metric Class which captures the design metric concept.

`#include <metric.h>`

Public Member Functions

- Metric ()
  Metric default constructor.
- Metric (const string &rname, const string &rtype, const string &runit)
  Metric constructor.
- Metric (Metric const &metric)
  Metric copy constructor.
- ~Metric ()
  Metric default destructor.
- string getName () const
  Metric name acccesor.
- string getType () const

Metric type acccesor.

- string getUnit () const
  Metric unit acccesor.
- unsigned long int getIntegerValue () const
  Metric integer value acccesor.
- long double getFloatValue () const
  Metric real value acccesor.
- void nameToLower ()
  Metric object name property character case conversion function to lower case.
- long double multiplier ()
- void setName (const string &rname)
  Metric name setting method.
- void setType (const string &rtype)
  Metric type setting method.
- void setUnit (const string &runit)
  Metric unit setting method.
- void setValue (const unsigned long int &rvalue)
  Metric integer value setting method.
- void setValue (const long double &rvalue)
  Metric floating point (real) value setting method.
- Metric & operator= (const Metric &m)
  Metric assignment operator.

Friends

- ostream & operator<< (ostream &os, const Metric &m)
  XML output stream operator (Metric object XML serializer).
- ostream & operator<< (ostream &os, const Metric *m)
  XML output stream operator (Metric object XML serializer).

Data Structures

- union **Value**

6.7.1 Detailed Description

Metric Class which captures the design metric concept.

Author:

Gerardo de Miguel González

Version:

1.0

Date:

June 2008

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Metric::Metric ()

Metric default constructor.

Returns:

A Metric Object

6.7.2.2 Metric::Metric (const string & *rname*, const string & *rtype*, const string & *runit*)

Metric constructor.

Parameters:

rname metric name (p.e execution_time)

rtype metric type (p.e float)

runit metric unit (p.e seconds)

Returns:

A Metric object

6.7.2.3 Metric::Metric (Metric const & *m*)

Metric copy constructor.

Parameters:

m a reference to a Metric object

Returns:

A Metric object

6.7.2.4 Metric::~Metric ()

Metric default destructor.

6.7.3 Member Function Documentation

6.7.3.1 string Metric::getName () const

Metric name acccesor.

Returns:

A string object which holds the name property content of the Metric object

6.7.3.2 string Metric::getType () const

Metric type acccesor.

Returns:

A string object which holds the type property content of the Metric object

6.7.3.3 string Metric::getUnit () const

Metric unit acccesor.

Returns:

A string object which holds the unit property content of the Metric object

6.7.3.4 unsigned long int Metric::getIntegerValue () const

Metric integer value acccesor.

Returns:

An unsigned integer number which holds the integer value of the Metric object

6.7.3.5 long double Metric::getFloatValue () const

Metric real value acccesor.

Returns:

A floating point number which holds the real value of the Metric object

6.7.3.6 void Metric::nameToLower ()

Metric object name property character case conversion function to lower case.

6.7.3.7 long double Metric::multiplier ()

6.7.3.8 void Metric::setName (const string & *rname*)

Metric name setting method.

Parameters:

rname a constant reference to a string object holding a name to set

6.7.3.9 void Metric::setType (const string & *rtype*)

Metric type setting method.

Parameters:

rtype a constant reference to a string object holding a type to set

6.7.3.10 void Metric::setUnit (const string & *runit*)

Metric unit setting method.

Parameters:

runit a constant reference to a string object holding a unit to set

6.7.3.11 void Metric::setValue (const unsigned long int & *rvalue*)

Metric integer value setting method.

Parameters:

rvalue a constant reference to an integer number holding a value to be set

6.7.3.12 void Metric::setValue (const long double & *rvalue*)

Metric floating point (real) value setting method.

Parameters:

rvalue a constant reference to floating-point number holding a value to be set

6.7.3.13 Metric & Metric::operator= (const Metric & *m*)

Metric assignment operator.

Parameters:

m a constant reference to a Metric object (rvalue)

Returns:

a reference to a Metric object (lvalue)

6.7.4 Friends And Related Function Documentation

6.7.4.1 ostream& operator<< (ostream & *os*, const Metric & *m*) [friend]

XML output stream operator (Metric object XML serializer).

Parameters:

os an output stream object reference

m a constant Metric object reference

Returns:

a reference to an output stream object

6.7.4.2 ostream& operator<< (ostream & *os*, const Metric * *m*) [friend]

XML output stream operator (Metric object XML serializer).

Parameters:

os an output stream object reference

m a constant Metric object pointer

Returns:

a reference to an output stream object

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

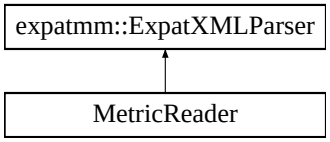
- metric.h
- metric.cpp

6.8 MetricReader Class Reference

Metric (expat + expatmm) Parser Class.

`#include <parse.h>`

Inheritance diagram for MetricReader::



Public Member Functions

- MetricReader ()

  MetricReader default constructor.

- MetricReader (const char *xmlFile)

  MetricReader constructor (from a XML file source).

- ~MetricReader ()

  Metric default destructor.

- bool writeXMLFile (const char *xmlOutputFile)

  writes an XML file with the Metric objects as specified in MULTICUBE

- bool writeXMLErrorFile (const char *xmlOutputFile, const char *reason, const char *kind)

  writes an XML error file as specified in MULTICUBE

- Metric getMetric ()

  Metric object acccesor Destructive retrieval of a Metric object from the Stack.

- void setMetric (Metric &m)

  Metric object setting method Insertion of a Metric object on the Stack.

- bool isMetricLeft () const

  checks whether there is some Metric object left in the Stack

- int size () const

  size of the Stack holding the Metric objects

Protected Member Functions

- virtual ssize_t read_block (void)

  expatmm (expat C++ wrapper) read_block method implementation Read the XML source (i.e a file) getting blocks of characters from a predefined block size which are then parsed and reported as events (SAX parsing style) which are handled with the start and end tag event handlers

- virtual void StartElement (const XML_Char *name, const XML_Char **attrs)

  StartElement Handler (SAX Parsing style) which triggers after parsing a starting tag in the XML file The data got from the attribute/value pairs is used to build a Metric object and the Metric object is pushed into a Stack (queued) container.

- virtual void EndElement (const XML_Char *name)

  EndElement Handler (SAX Parsing style) which triggers after parsing an ending tag in the XML file.

6.8.1 Detailed Description

Metric (expat + expatmm) Parser Class.

Author:

Gerardo de Miguel González

Version:

1.0

Date:

June 2008

6.8.2 Constructor & Destructor Documentation

6.8.2.1 MetricReader::MetricReader ()

MetricReader default constructor.

Returns:

A MetricReader Object

Warning:

The default constructor is disabled declaring it private

6.8.2.2 MetricReader::MetricReader (const char * *xmlFile*)

MetricReader constructor (from a XML file source).

Parameters:

xmlfile XML file name where the MetricReader object is built from

Returns:

A MetricReader object

6.8.2.3 MetricReader::~MetricReader ()

Metric default destructor.

6.8.3 Member Function Documentation

6.8.3.1 bool MetricReader::writeXMLFile (const char * *xmlOutputFile*)

writes an XML file with the Metric objects as specified in MULTICUBE

Parameters:

xmlOutputFile constant char array holding the name of the XML output file

Returns:

A boolean result 'True' if the writing process has been successful

6.8.3.2 bool MetricReader::writeXMLErrorFile (const char * *xmlOutputFile*, const char * *reason*, const char * *kind*)

writes an XML error file as specified in MULTICUBE

Parameters:

xmlOutputFile constant char array holding the name of the XML output file

reason a word description of the exception or error reported

kind the severity grade of the error reported (i.e fatal)

Returns:

A boolean result 'True' if the writing process has been successful

6.8.3.3 Metric MetricReader::getMetric ()

Metric object acccesor Destructive retrieval of a Metric object from the Stack.

Returns:

A Metric object

6.8.3.4 void MetricReader::setMetric (Metric & *m*)

Metric object setting method Insertion of a Metric object on the Stack.

Parameters:

m the Metric object that is going to be pushed into the Stack

6.8.3.5 bool MetricReader::isMetricLeft () const

checks whether there is some Metric object left in the Stack

Returns:

A boolean result 'True' if there is some Metric object left

6.8.3.6 int MetricReader::size () const

size of the Stack holding the Metric objects

Returns:

the number of Metric objects which are in the Stack

6.8.3.7 ssize_t MetricReader::read_block (void) [protected, virtual]

expatmm (expat C++ wrapper) read_block method implementation Read the XML source (i.e a file) getting blocks of characters from a predefined block size which are then parsed and reported as events (SAX parsing style) which are handled with the start and end tag event handlers

Returns:

the number of blocks read or '-1' if there is some error

Reimplemented from expatmm::ExpatXMLParser.

6.8.3.8 void MetricReader::StartElement (const XML_Char * *name*, const XML_Char ** *attrs*) [protected, virtual]

StartElement Handler (SAX Parsing style) which triggers after parsing a starting tag in the XML file The data got from the attribute/value pairs is used to build a Metric object and the Metric object is pushed into a Stack (queued) container.

Parameters:

name constant char array which holds the tag's name which is parsed

attrs constant Nx2 array which holds the attribute/value pairs within the XML tag which is parsed

Returns:

Reimplemented from expatmm::ExpatXMLParser.

6.8.3.9 void MetricReader::EndElement (const XML_Char * *name*) [protected, virtual]

EndElement Handler (SAX Parsing style) which triggers after parsing an ending tag in the XML file.

Parameters:

name constant char array which holds the tag's name which is parsed

Returns:

Reimplemented from expatmm::ExpatXMLParser.

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

- parse.h
- parse.cpp

6.9 sw_allocation Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- char * priority
- char * policy
- char * args
- char * task_name
- char * os_name
- char * resource_name
- struct sw_task * task

- struct hw_instance * hw_resource
- struct hw_group * hw_group
- struct sw_instance * sw_resource
- int offset
- void * scope_data

6.9.1 Field Documentation

6.9.1.1 char* sw_allocation::name

6.9.1.2 char* sw_allocation::path

6.9.1.3 char* sw_allocation::priority

6.9.1.4 char* sw_allocation::policy

6.9.1.5 char* sw_allocation::args

6.9.1.6 char* sw_allocation::task_name

6.9.1.7 char* sw_allocation::os_name

6.9.1.8 char* sw_allocation::resource_name

6.9.1.9 struct sw_task* sw_allocation::task [read]

6.9.1.10 struct hw_instance* sw_allocation::hw_resource [read]

6.9.1.11 struct hw_group* sw_allocation::hw_group [read]

6.9.1.12 struct sw_instance* sw_allocation::sw_resource [read]

6.9.1.13 int sw_allocation::offset

6.9.1.14 void* sw_allocation::scope_data

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

- xml_if.h

6.10 sw_component Struct Reference

```
#include <xml_if.h>
```

Data Fields

- char * name
- char * path
- char * type

6.10.1 Field Documentation

6.10.1.1 char* sw_component::name

6.10.1.2 char* sw_component::path

6.10.1.3 char* sw_component::type

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

- xml_if.h

6.11 sw_instance Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- char * type
- char * component_name
- char * resource_name
- struct sw_component * component
- struct hw_instance * hw_resource
- struct hw_group * hw_group
- int offset
- void * scope_data

6.11.1 Field Documentation

6.11.1.1 char* sw_instance::name

6.11.1.2 char* sw_instance::path

6.11.1.3 char* sw_instance::type

6.11.1.4 char* sw_instance::component_name

6.11.1.5 char* sw_instance::resource_name

6.11.1.6 struct sw_component* sw_instance::component [read]

6.11.1.7 struct hw_instance* sw_instance::hw_resource [read]

6.11.1.8 struct hw_group* sw_instance::hw_group [read]

6.11.1.9 int sw_instance::offset

6.11.1.10 void* sw_instance::scope_data

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

- xml_if.h

6.12 sw_task Struct Reference

`#include <xml_if.h>`

Data Fields

- char * name
- char * path
- char * type
- char * function
- char * file
- char * class_name
- char * compute_time
- char * period
- char * data_size
- char * init_time
- char * fin_time

6.12.1 Field Documentation

6.12.1.1 char* sw_task::name

6.12.1.2 char* sw_task::path

6.12.1.3 char* sw_task::type

6.12.1.4 char* sw_task::function

6.12.1.5 char* sw_task::file

6.12.1.6 char* sw_task::class_name

6.12.1.7 char* sw_task::compute_time

6.12.1.8 char* sw_task::period

6.12.1.9 char* sw_task::data_size

6.12.1.10 char* sw_task::init_time

6.12.1.11 char* sw_task::fin_time

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

- xml_if.h

6.13 xml_basic_info Struct Reference

`#include <xml_hierarchy.h>`

Data Fields

- char * name
- char ** args
- struct xml_basic_info * parent
- struct list_elem * children_header
- char index
- char * init_index
- char * repeat

6.13.1 Field Documentation

6.13.1.1 char* xml_basic_info::name

6.13.1.2 char** xml_basic_info::args

6.13.1.3 struct xml_basic_info* xml_basic_info::parent [read]

6.13.1.4 struct list_elem* xml_basic_info::children_header [read]

6.13.1.5 char xml_basic_info::index

6.13.1.6 char* xml_basic_info::init_index

6.13.1.7 char* xml_basic_info::repeat

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

- xml_hierarchy.h

6.14 xml_parameter_info Struct Reference

`#include <xml_configuration_file.h>`

Data Fields

- char * name
- char * value

6.14.1 Field Documentation

6.14.1.1 char* xml_parameter_info::name

6.14.1.2 char* xml_parameter_info::value

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

- xml_configuration_file.h

6.15 xml_scope_simulation_info_t Struct Reference

`#include <uc_load_xml.h>`

Data Fields

- long long simulation_time
- enum sc_time_unit sim_time_unit
- int debug_level
- int warnings

6.15.1 Field Documentation

6.15.1.1 long long xml_scope_simulation_info_t::simulation_time

6.15.1.2 enum sc_time_unit xml_scope_simulation_info_t::sim_time_unit

6.15.1.3 int xml_scope_simulation_info_t::debug_level

6.15.1.4 int xml_scope_simulation_info_t::warnings

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

- uc_load_xml.h

7 File Documentation

7.1 expatmm-libdef.h File Reference

7.2 ExpatMM-version.cpp File Reference

```
#include <cstdio>
#include <string>
#include "expat.h"
#include "expatmm.h"
```

Functions

- std::string expatmm::getExpatMMVersion (void)

7.3 expatmm.h File Reference

```
#include "expatmm-libdef.h"
#include "ExpatXMLParser.h"
```

Namespaces

- namespace expatmm

Defines

- #define EXPATMM_LIBRARY_MAJOR 1
- #define EXPATMM_LIBRARY_MINOR 0
- #define EXPATMM_LIBRARY_REVISION 0
- #define EXPATMM_LIBRARY_VERSION "1.0.0"

Functions

- std::string expatmm::getExpatMMVersion (void)

7.3.1 Define Documentation

7.3.1.1 #define EXPATMM_LIBRARY_MAJOR 1

7.3.1.2 #define EXPATMM_LIBRARY_MINOR 0

7.3.1.3 #define EXPATMM_LIBRARY_REVISION 0

7.3.1.4 #define EXPATMM_LIBRARY_VERSION "1.0.0"

7.4 ExpatXMLParser.cpp File Reference

```
#include <cstddef>
#include <cstring>
#include <iostream>
#include "expat.h"
#include "expat_external.h"
#include "ExpatXMLParser.h"
```

7.5 ExpatXMLParser.h File Reference

```
#include "expatmm-libdef.h"
```

Namespaces

- namespace expatmm

Data Structures

- class expatmm::ExpatXMLParser

Defines

- #define XML_CHUNK_SIZE 10240

7.5.1 Define Documentation

7.5.1.1 #define XML_CHUNK_SIZE 10240

7.6 ExpatXMLParser.h File Reference

```
#include "expatmm-libdef.h"
```

Namespaces

- namespace expatmm

Data Structures

- class expatmm::ExpatXMLParser

Defines

- #define XML_CHUNK_SIZE 10240

7.6.1 Define Documentation

7.6.1.1 #define XML_CHUNK_SIZE 10240

7.7 metric.cpp File Reference

```
#include "metric.h"
```

Functions

- ostream & operator<< (ostream &os, const Metric &m)
  XML output stream operator (Metric object XML serializer).
- ostream & operator<< (ostream &os, const Metric *m)
  XML output stream operator (Metric object XML serializer).

7.7.1 Function Documentation

7.7.1.1 ostream& operator<< (ostream & *os*, const Metric * *m*)

XML output stream operator (Metric object XML serializer).

Parameters:

os an output stream object reference

m a constant Metric object pointer

Returns:

a reference to an output stream object

7.7.1.2 ostream& operator<< (ostream & *os*, const Metric & *m*)

XML output stream operator (Metric object XML serializer).

Parameters:

os an output stream object reference

m a constant Metric object reference

Returns:

a reference to an output stream object

7.8 metric.h File Reference

```
#include <iostream>
#include <string>
#include <iomanip>
```

Data Structures

- class Metric

  Metric Class which captures the design metric concept.

- union **Metric::Value**

7.9 parse.cpp File Reference

```
#include <cstddef>
#include <cstring>
#include <sys/types.h>
#include <strings.h>
#include "metric.h"
#include "expat.h"
#include "expatmm.h"
#include "parse.h"
```

7.10 parse.h File Reference

```
#include <fstream>
#include <queue>
#include "ExpatXMLParser.h"
```

Data Structures

- class MetricReader

  Metric (expat + expatmm) Parser Class.

7.11 uc_create_xml_platform.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
#include <ctype.h>
#include <vector>
#include "uc_load_xml.h"
#include "uc_create_xml_platform.h"
#include "xml_input.h"
#include "xml_if.h"
#include "xml_list.h"
```

```
#include "sc_scope.h"
#include "uc_unistd.h"
#include "uc_hal_sw.h"
```

Functions

- void * get_dynamic_function (char *file, char *function)

  Obtain the pointer to a function from a dynamic library, knowing the function name and the file name. The function has to be 'extern "C"'.

- void * get_static_function (char *function)

  Obtain the pointer to a function from a the executable itself, knowing the function name and the file name. Tries with the name in C and C++ for gcc 3.x-4.x.

- void load_task_info (struct _xml_intermediate_info *info, struct sw_allocation *alloc)

  Load the information required to execute a function from the sw_allocation struct, where all are strings, to the _xml_intermediate_info, as function poineter, priority, policy, arguments, ...

- void load_taskload_info (struct _xml_taskload_info *info, struct sw_allocation *alloc)

  Create the data structure to create a task load from the xml structure containing xml strings.

- void * _xml_intermediate_function (void *data)

  Intermediate function used to launch a SW task. Read the data struct and starts the function indicated in the function pointer.

- void * _xml_taskload_function (void *data)

  Function executed to simulate a task load.

- void instanciate_sw_components ()

  Instanciate the SW components (OS) described in the XML System Description file.

- void instanciate_hw_components ()

  Instanciate the HW components described in the XML System Description file: processors, buses, memories, ...

- void connect_hw_components ()

  Connect the HW components of the system as described in the XML System Description file.

- void allocate_sw_tasks ()

  Function executed to instanciate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. It if is a driver, it is adequately load.

Variables

- char * executable_name
- long long uc_segment_time
- std::vector< UC_TLM_bus_class * > system_buses

7.11.1 Function Documentation

7.11.1.1 void* _xml_intermediate_function (void * *data*)

Intermediate function used to launch a SW task. Read the data struct and starts the function indicated in the function pointer.

Parameters:

data Structure where the infomation is stored

Returns:

Return value of the called function

7.11.1.2 void* _xml_taskload_function (void * *data*)

Function executed to simulate a task load.

Parameters:

data Structure where the taskload infomation is stored

Returns:

NULL, used for compatibility

7.11.1.3 void allocate_sw_tasks ()

Function executed to instanciate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. It if is a driver, it is adequately load.

Parameters:

***return**

7.11.1.4 void connect_hw_components ()

Connect the HW components of the system as described in the XML System Description file.

Current version only connections to connect the network interface to the network

7.11.1.5 void* get_dynamic_function (char * *file*, char * *function*)

Obtain the pointer to a function from a dynamic library, knowing the function name and the file name. The function has to be 'extern "C"'.

Parameters:

file Name of the dynamic library where the function is

function Name of the function

Returns:

The function pointer

7.11.1.6 void* get_static_function (char * *function*)

Obtain the pointer to a function from a the executable itself, knowing the function name and the file name. Tries with the name in C and C++ for gcc 3.x-4.x.

Parameters:

function Name of the function

Returns:

The function pointer

7.11.1.7 void instanciate_hw_components ()

Instanciate the HW components described in the XML System Description file: processors, buses, memories, ...

Returns:

7.11.1.8 void instanciate_sw_components ()

Instanciate the SW components (OS) described in the XML System Description file.

Returns:

7.11.1.9 void load_task_info (struct _xml_intermediate_info * *info*, struct sw_allocation * *alloc*)

Load the information required to execute a function from the sw_allocation struct, where all are strings, to the _xml_intermediate_info, as function poineter, priority, policy, arguments, ...

Parameters:

info Structure where the infomation is stored

alloc Structure from where the information is read

Returns:

7.11.1.10 void load_taskload_info (struct _xml_taskload_info * *info*, struct sw_allocation * *alloc*)

Create the data structure to create a task load from the xml structure containing xml strings.

Parameters:

info Structure where the infomation is stored

alloc Structure from where the information is read

Returns:

7.11.2 Variable Documentation

7.11.2.1 char* executable_name

7.11.2.2 std::vector<UC_TLM_bus_class*> system_buses

7.11.2.3 long long uc_segment_time

7.12 uc_create_xml_platform.h File Reference

Functions

- void instanciate_sw_components ()

  Instanciate the SW components (OS) described in the XML System Description file.

- void instanciate_hw_components ()

  Instanciate the HW components described in the XML System Description file: processors, buses, memories, ...

- void connect_hw_components ()

  Connect the HW components of the system as described in the XML System Description file.

- void allocate_sw_tasks ()

  Function executed to instanciate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. It if is a driver, it is adequately load.

7.12.1 Function Documentation

7.12.1.1 void allocate_sw_tasks ()

Function executed to instanciate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. It if is a driver, it is adequately load.

Parameters:

***return**

7.12.1.2 void connect_hw_components ()

Connect the HW components of the system as described in the XML System Description file.

Current version only connections to connect the network interface to the network

7.12.1.3 void instanciate_hw_components ()

Instanciate the HW components described in the XML System Description file: processors, buses, memories, ...

Returns:

7.12.1.4 void instanciate_sw_components ()

Instanciate the SW components (OS) described in the XML System Description file.

Returns:

7.13 uc_load_xml.cpp File Reference

```
#include "sc_scope.h"
#include "xml_main.h"
#include "uc_load_xml.h"
#include "uc_create_xml_platform.h"
```

Functions

- int sc_main (int argc, char **argv)

  SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.

- void destroy_objects ()

  Destroy the dynamically allocated simulation objects and print out the results on screen.

- void usage (char *programa)

  Prints out the program command-line help.

- void load_xml_platform_file (char *file)

  Execute all the functions required to read the XML System Description file and create the system model.

- void load_xml_configuration_file (char *xml_file)

  Load the XML System Configuration file.

Variables

- UC_gui_connector * gui_connector
- xml_scope_simulation_info_t xml_scope_simulation_info = {-1, SC_NS, 0}
- char * executable_name
- int activate_backtrace
- vector< UC_rtos_class * > rtos_list
- vector< UC_NoC_Interface * > simulator_list

7.13.1 Function Documentation

7.13.1.1 void destroy_objects ()

Destroy the dynamically allocated simulation objects and print out the results on screen.

7.13.1.2 void load_xml_configuration_file (char * *xml_file*)

Load the XML System Configuration file.

Parameters:

file Name of the XML System Configuration file

7.13.1.3 void load_xml_platform_file (char * *file*)

Execute all the functions required to read the XML System Description file and create the system model.

Parameters:

file Name of the XML System Description file

7.13.1.4 int sc_main (int *argc*, char ** *argv*)

SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.

Parameters:

argc Number of arguments received from the terminal

argv Arguments received from the terminal

Returns:

Program result

7.13.1.5 void usage (char * *programa*)

Prints out the program command-line help.

7.13.2 Variable Documentation

7.13.2.1 int activate_backtrace

7.13.2.2 char* executable_name

7.13.2.3 UC_gui_connector* gui_connector

7.13.2.4 vector<UC_rtos_class *> rtos_list

7.13.2.5 vector<UC_NoC_Interface *> simulator_list

7.13.2.6 xml_scope_simulation_info_t xml_scope_simulation_info = {-1, SC_NS, 0}

7.14 uc_load_xml.h File Reference

```
#include "sc_scope.h"
#include <vector>
#include "expat.h"
#include "expatmm.h"
#include "metric.h"
#include "parse.h"
```

Data Structures

- struct xml_scope_simulation_info_t

Functions

- int sc_main (int argc, char **argv)

  SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.

- void usage (char *program)

  Prints out the program command-line help.

- void load_xml_platform_file (char *xml_config)

  Execute all the functions required to read the XML System Description file and create the system model.

- void load_xml_configuration_file (char *xml_file)

  Load the XML System Configuration file.

- int obtain_xml_metrics (MetricReader &mr)
- void destroy_objects ()

  Destroy the dynamically allocated simulation objects and print out the results on screen.

Variables

- xml_scope_simulation_info_t xml_scope_simulation_info
- vector< UC_rtos_class * > rtos_list
- vector< UC_NoC_Interface * > simulator_list
- UC_gui_connector * gui_connector

7.14.1 Function Documentation

7.14.1.1 void destroy_objects ()

Destroy the dynamically allocated simulation objects and print out the results on screen.

7.14.1.2 void load_xml_configuration_file (char * *xml_file*)

Load the XML System Configuration file.

Parameters:

file Name of the XML System Configuration file

7.14.1.3 void load_xml_platform_file (char * *file*)

Execute all the functions required to read the XML System Description file and create the system model.

Parameters:

file Name of the XML System Description file

7.14.1.4 int obtain_xml_metrics (MetricReader & *mr*)

7.14.1.5 int sc_main (int *argc*, char ** *argv*)

SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.

Parameters:

argc Number of arguments received from the terminal

argv Arguments received from the terminal

Returns:

Program result

7.14.1.6 void usage (char * *program*)

Prints out the program command-line help.

7.14.2 Variable Documentation

7.14.2.1 UC_gui_connector* gui_connector

7.14.2.2 vector<UC_rtos_class *> rtos_list

7.14.2.3 vector<UC_NoC_Interface *> simulator_list

7.14.2.4 xml_scope_simulation_info_t xml_scope_simulation_info

7.15 xml_configuration_file.c File Reference

```
#include "xml_configuration_file.h"
#include "xml_list.h"
#include "xml_if.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

Functions

- const char * xml_load_configuration_parameter (const char *param)
  Get the corresponding value for a configuration parameter.
- void xml_add_parameter (char *userData, const char *name, const char **atts)
  Decode the attributes of a configuration parameter xml clause.
- void xml_end_parameter (char *userData, const char *name)
  Close a configuration xml clause.

Variables

- struct list_elem * xml_config_parameter_header = 0
- struct xml_parameter_info * prev_xml_parameter_info = NULL

7.15.1 Function Documentation

7.15.1.1 void xml_add_parameter (char * *userData*, const char * *name*, const char ** *atts*)

Decode the attributes of a configuration parameter xml clause.

Parameters:

userData Pointer to the user defined buffer

name Name of the parameter xml clause

atts Attributes of the xml clause

Returns:

7.15.1.2 void xml_end_parameter (char * *userData*, const char * *name*)

Close a configuration xml clause.

Parameters:

userData Pointer to the user defined buffer

name Name of the parameter xml clause

Returns:

7.15.1.3 const char* xml_load_configuration_parameter (const char * *param*)

Get the corresponding value for a configuration parameter.

Parameters:

param Name of the parameter

Returns:

Parameter value

7.15.2 Variable Documentation

7.15.2.1 struct xml_parameter_info* prev_xml_parameter_info = NULL

7.15.2.2 struct list_elem* xml_config_parameter_header = 0

7.16 xml_configuration_file.h File Reference

Data Structures

- struct xml_parameter_info

Functions

- const char * xml_load_configuration_parameter (const char *param)
  Get the corresponding value for a configuration parameter.
- void xml_add_parameter (char *userData, const char *name, const char **atts)
  Decode the attributes of a configuration parameter xml clause.
- void xml_end_parameter (char *userData, const char *name)
  Close a configuration xml clause.

7.16.1 Function Documentation

7.16.1.1 void xml_add_parameter (char * *userData*, const char * *name*, const char ** *atts*)

Decode the attributes of a configuration parameter xml clause.

Parameters:

userData Pointer to the user defined buffer

name Name of the parameter xml clause

atts Attributes of the xml clause

Returns:

7.16.1.2 void xml_end_parameter (char * *userData*, const char * *name*)

Close a configuration xml clause.

Parameters:

userData Pointer to the user defined buffer

name Name of the parameter xml clause

Returns:

7.16.1.3 const char* xml_load_configuration_parameter (const char * *param*)

Get the corresponding value for a configuration parameter.

Parameters:

param Name of the parameter

Returns:

Parameter value

7.17 xml_hierarchy.c File Reference

```
#include "xml_hierarchy.h"
#include "xml_configuration_file.h"
#include "xml_list.h"
#include "xml_if.h"
#include "uc_load_xml.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

Functions

- void xml_add_hierarchy (char *path, const char *name, const char **args)

  Add a new struct with the info of a clause to the correct point of the xml data hierarchy.

- void xml_up_hierarchy (char *path, const char *name)

  Close a clause moving the current hierarchy node to the upper node.

- void xml_increase_path (char *path, struct xml_basic_info *elem, int index)

  Increase the path when entering a new clause.

- void xml_decrease_path (char *path, const char *name)

  Decrease the path when exiting a new clause.

- struct xml_basic_info * xml_search_path (const char *path)

  Return the information struct of the clause indicated by the input path.

- char * xml_check_name (const char *name, struct xml_basic_info *prev_elem, int index)

  Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.

- void xml_display_hierarchy_node (struct xml_basic_info *info, char *path, int index)

  Create a node for a xml clause.

- void xml_display_hierarchy (char *path)

  Create a node for a certain path.

- const char * xml_string_index_convert (const char *data_in)

  Check if the argument is a configuratino parameter (start with '_') or contain a "repeat" index and fix the string if required.

- int xml_chech_instance (int index, const char **atts)

  Check if the clause has been selected in the implementation clause or must be discarded.

- void xml_select_implementation (const char **atts)

  Decode the attributes of an implementation clause.

- void xml_simulation_parameters (const char **atts)

  Decode the attributes of a simulation clause.

Variables

- struct xml_basic_info * xml_hierarchy_header = 0
- struct xml_basic_info * xml_hierarchy_current = 0
- int xml_indexes [30]
- char * xml_select_implementation_array [11] = {NULL}

7.17.1 Function Documentation

7.17.1.1 void xml_add_hierarchy (char * *path*, const char * *name*, const char ** *args*)

Add a new struct with the info of a clause to the correct point of the xml data hierarchy.

Parameters:

path Clause path

name Clause name

atts Clause attributes

Returns:

7.17.1.2 int xml_chech_instance (int *index*, const char ** *atts*)

Check if the clause has been selected in the implementation clause or must be discarded.

Parameters:

index type of index

atts arguments of the clause

Returns:

1 if valid, 0 otherwise

7.17.1.3 char* xml_check_name (const char * *name*, struct xml_basic_info * *prev_elem*, int *index*)

Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.

Parameters:

name Clause name

prev_elem Clause info list

index used when creating several copies in a repeat clause

Returns:

The new clause name

7.17.1.4 void xml_decrease_path (char * *path*, const char * *name*)

Decrease the path when exiting a new clause.

Parameters:

path Clause path

elem Clause info

Returns:

7.17.1.5 void xml_display_hierarchy (char * *path*)

Create a node for a certain path.

Parameters:

path Clause path

Returns:

The new clause name

7.17.1.6 void xml_display_hierarchy_node (struct xml_basic_info * *info*, char * *path*, int *index*)

Create a node for a xml clause.

Parameters:

info Clause info list

name Clause info

index used when creating several copies in a repeat clause

Returns:

The new clause name

7.17.1.7 void xml_increase_path (char * *path*, struct xml_basic_info * *elem*, int *index*)

Increase the path when entering a new clause.

Parameters:

path Clause path

elem Clause info

index used when creating several copies in a repeat clause

Returns:

7.17.1.8 struct xml_basic_info* xml_search_path (const char * *path*) [read]

Return the information struct of the clause indicated by the input path.

Parameters:

path Clause path

Returns:

The clause info

7.17.1.9 void xml_select_implementation (const char ** *atts*)

Decode the attributes of an implementation clause.

Parameters:

atts Clause attributes

Returns:

7.17.1.10 void xml_simulation_parameters (const char ** *atts*)

Decode the attributes of a simulation clause.

Parameters:

atts Clause attributes

Returns:

7.17.1.11 const char* xml_string_index_convert (const char * *data_in*)

Check if the argument is a configuratino parameter (start with '_') or contain a "repeat" index and fix the string if required.

Parameters:

data_in Input argument

Returns:

The fixed argument

7.17.1.12 void xml_up_hierarchy (char * *path*, const char * *name*)

Close a clause moving the current hierarchy node to the upper node.

Parameters:

path Clause path

name Clause name

Returns:

7.17.2 Variable Documentation

7.17.2.1 struct xml_basic_info* xml_hierarchy_current = 0

7.17.2.2 struct xml_basic_info* xml_hierarchy_header = 0

7.17.2.3 int xml_indexes[30]

7.17.2.4 char* xml_select_implementation_array[11] = {NULL}

7.18 xml_hierarchy.h File Reference

Data Structures

- struct xml_basic_info

Functions

- void xml_display_hierarchy (char *path)

  Create a node for a certain path.

- void xml_add_hierarchy (char *path, const char *name, const char **args)

  Add a new struct with the info of a clause to the correct point of the xml data hierarchy.

- void xml_up_hierarchy (char *path, const char *name)

  Close a clause moving the current hierarchy node to the upper node.

- void xml_decrease_path (char *path, const char *name)

  Decrease the path when exiting a new clause.

- void xml_increase_path (char *path, struct xml_basic_info *elem, int index)

  Increase the path when entering a new clause.

- struct xml_basic_info * xml_search_path (const char *path)

  Return the information struct of the clause indicated by the input path.

- char * xml_check_name (const char *name, struct xml_basic_info *prev_elem, int index)

  Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.

- const char * xml_string_index_convert (const char *data_in)

  Check if the argument is a configuratino parameter (start with '_') or contain a "repeat" index and fix the string if required.

- int xml_chech_instance (int i, const char **atts)

  Check if the clause has been selected in the implementation clause or must be discarded.

- void xml_select_implementation (const char **atts)

  Decode the attributes of an implementation clause.

- void xml_simulation_parameters (const char **args)

  Decode the attributes of a simulation clause.

7.18.1 Function Documentation

7.18.1.1 void xml_add_hierarchy (char * *path*, const char * *name*, const char ** *args*)

Add a new struct with the info of a clause to the correct point of the xml data hierarchy.

Parameters:

path Clause path
name Clause name
atts Clause attributes

Returns:

7.18.1.2 int xml_chech_instance (int *index*, const char ** *atts*)

Check if the clause has been selected in the implementation clause or must be discarded.

Parameters:

index type of index
atts arguments of the clause

Returns:

1 if valid, 0 otherwise

7.18.1.3 char* xml_check_name (const char * *name*, struct xml_basic_info * *prev_elem*, int *index*)

Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.

Parameters:

name Clause name
prev_elem Clause info list
index used when creating several copies in a repeat clause

Returns:

The new clause name

7.18.1.4 void xml_decrease_path (char * *path*, const char * *name*)

Decrease the path when exiting a new clause.

Parameters:

path Clause path
elem Clause info

Returns:

7.18.1.5 void xml_display_hierarchy (char * *path*)

Create a node for a certain path.

Parameters:

path Clause path

Returns:

The new clause name

7.18.1.6 void xml_increase_path (char * *path*, struct xml_basic_info * *elem*, int *index*)

Increase the path when entering a new clause.

Parameters:

path Clause path

elem Clause info

index used when creating several copies in a repeat clause

Returns:

7.18.1.7 struct xml_basic_info* xml_search_path (const char * *path*) [read]

Return the information struct of the clause indicated by the input path.

Parameters:

path Clause path

Returns:

The clause info

7.18.1.8 void xml_select_implementation (const char ** *atts*)

Decode the attributes of an implementation clause.

Parameters:

atts Clause attributes

Returns:

7.18.1.9 void xml_simulation_parameters (const char ** *atts*)

Decode the attributes of a simulation clause.

Parameters:

atts Clause attributes

Returns:

7.18.1.10 const char* xml_string_index_convert (const char * *data_in*)

Check if the argument is a configuratino parameter (start with '_') or contain a "repeat" index and fix the string if required.

Parameters:

data_in Input argument

Returns:

The fixed argument

7.18.1.11 void xml_up_hierarchy (char * *path*, const char * *name*)

Close a clause moving the current hierarchy node to the upper node.

Parameters:

path Clause path

name Clause name

Returns:

7.19 xml_if.c File Reference

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "xml_if.h"
#include "xml_input.h"
#include "xml_list.h"
#include "xml_hierarchy.h"
```

Functions

- int strcasecmp_null (const char *str1, const char *str2)
  Compares two strings checking if any is NULL.
- unsigned long int atoi_null (const char *str)
  Obtain a integer value from a string, checking if the string is null.
- int xml_declare_function (char *path, const char *name, const char **atts)
  Check the type of clause, and call the corresponding function to decode the arguments.
- void xml_close_declare (char *path, const char *name)
  Close a xml clause.
- void * xml_load_hw_component (char *path, const char **attr)
  Decode the arguments of a HW_Component clause.
- void * xml_load_hw_instance (char *path, const char **attr)
  Decode the arguments of a HW_Instance clause.
- void * xml_load_hw_connection (char *path, const char **attr)
  Decode the arguments of a HW_Connection clause.
- void * xml_load_hw_group (char *path, const char **attr)
  Decode the arguments of a Computing_Group clause.
- void * xml_add_to_hw_group (char *path, const char **attr)
  Decode the arguments of a Computing_Resource clause.
- void * xml_load_sw_component (char *path, const char **attr)
  Decode the arguments of a SW_Component clause.
- void * xml_load_sw_instance (char *path, const char **attr)
  Decode the arguments of a SW instance clause.
- void * xml_load_exec_component (char *path, const char **attr)
  Decode the arguments of a Exec_Component clause.
- void * xml_load_exec_instance (char *path, const char **attr)
  Decode the arguments of a Exec_Instance clause.
- void xml_add_process_argument (char *path, const char **attr)
  Add an argument to the previous sw allocation.

7.19.1 Function Documentation

7.19.1.1 unsigned long int atoi_null (const char * *str*)

Obtain a integer value from a string, checking if the string is null.

Parameters:

str The string to be converted

Returns:

the value, or 0

7.19.1.2 int strcasecmp_null (const char * *str1*, const char * *str2*)

Compares two strings checking if any is NULL.

Parameters:

str1 String to be compared
str2 String to be compared

Returns:

1 if equal, 0 otherwise.

7.19.1.3 void xml_add_process_argument (char * *path*, const char ** *attr*)

Add an argument to the previous sw allocation.

Parameters:

path Clause path
name Clause name

7.19.1.4 void* xml_add_to_hw_group (char * *path*, const char ** *attr*)

Decode the arguments of a Computing_Resource clause.

Parameters:

path Clause path
name Clause name

Returns:

the created struct

7.19.1.5 void xml_close_declare (char * *path*, const char * *name*)

Close a xml clause.

Parameters:

path Clause path
name Clause name

Returns:

7.19.1.6 int xml_declare_function (char * *path*, const char * *name*, const char ** *atts*)

Check the type of clause, and call the corresponding function to decode the arguments.

Parameters:

path Clause path

name Clause name

atts Clause attributes

Returns:

0 if ok, otherwise error

7.19.1.7 void* xml_load_exec_component (char * *path*, const char ** *attr*)

Decode the arguments of a Exec_Component clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.8 void* xml_load_exec_instance (char * *path*, const char ** *attr*)

Decode the arguments of a Exec_Instance clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.9 void* xml_load_hw_component (char * *path*, const char ** *attr*)

Decode the arguments of a HW_Component clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.10 void* xml_load_hw_connection (char * *path*, const char ** *attr*)

Decode the arguments of a HW_Connection clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.11 void* xml_load_hw_group (char * *path*, const char ** *attr*)

Decode the arguments of a Computing_Group clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.12 void* xml_load_hw_instance (char * *path*, const char ** *attr*)

Decode the arguments of a HW_Instance clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.13 void* xml_load_sw_component (char * *path*, const char ** *attr*)

Decode the arguments of a SW_Component clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.19.1.14 void* xml_load_sw_instance (char * *path*, const char ** *attr*)

Decode the arguments of a SW instance clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20 xml_if.h File Reference

Data Structures

- struct hw_component
- struct hw_instance
- struct hw_connection
- struct hw_group
- struct sw_component
- struct sw_instance
- struct sw_task
- struct sw_allocation

Functions

- int strcasecmp_null (const char *str1, const char *str2)

  Compares two strings checking if any is NULL.

- unsigned long int atoi_null (const char *str)

  Obtain a integer value from a string, checking if the string is null.

- void xml_close_declare (char *path, const char *name)

  Close a xml clause.

- int xml_declare_function (char *path, const char *name, const char **atts)

  Check the type of clause, and call the corresponding function to decode the arguments.

- void * xml_load_hw_component (char *path, const char **atts)

  Decode the arguments of a HW_Component clause.

- void * xml_load_hw_instance (char *path, const char **atts)

  Decode the arguments of a HW_Instance clause.

- void * xml_load_hw_connection (char *path, const char **atts)

  Decode the arguments of a HW_Connection clause.

- void * xml_load_hw_group (char *path, const char **atts)

  Decode the arguments of a Computing_Group clause.

- void * xml_load_sw_component (char *path, const char **atts)
  Decode the arguments of a SW_Component clause.
- void * xml_load_sw_instance (char *path, const char **atts)
  Decode the arguments of a SW instance clause.
- void * xml_load_exec_component (char *path, const char **atts)
  Decode the arguments of a Exec_Component clause.
- void * xml_load_exec_instance (char *path, const char **atts)
  Decode the arguments of a Exec_Instance clause.
- void * xml_add_to_hw_group (char *path, const char **atts)
  Decode the arguments of a Computing_Resource clause.
- void xml_add_process_argument (char *path, const char **attr)
  Add an argument to the previous sw allocation.
- void xml_hw_component_string_convert (struct hw_component *component)
- void xml_hw_instance_string_convert (struct hw_instance *instance)
- void xml_hw_connection_string_convert (struct hw_connection *connection)
- void xml_hw_group_string_convert (struct hw_group *group)
- void xml_sw_instance_string_convert (struct sw_instance *instance)
- void xml_sw_task_string_convert (struct sw_task *task)
- void xml_sw_allocation_string_convert (struct sw_allocation *allocation)
- void xml_connect_hw_instance (struct hw_instance *instance)
- void xml_connect_hw_connection (struct hw_connection *connection)
- void xml_connect_hw_group (struct hw_group *group)
- void xml_connect_sw_instance (struct sw_instance *instance)
- int xml_connect_sw_allocation (struct sw_allocation *allocation)

Variables

- int xml_info_level

7.20.1 Function Documentation

7.20.1.1 unsigned long int atoi_null (const char * *str*)

Obtain a integer value from a string, checking if the string is null.

Parameters:

str The string to be converted

Returns:

the value, or 0

7.20.1.2 int strcasecmp_null (const char * *str1*, const char * *str2*)

Compares two strings checking if any is NULL.

Parameters:

str1 String to be compared

str2 String to be compared

Returns:

1 if equal, 0 otherwise.

7.20.1.3 void xml_add_process_argument (char * *path*, const char ** *attr*)

Add an argument to the previous sw allocation.

Parameters:

path Clause path

name Clause name

7.20.1.4 void* xml_add_to_hw_group (char * *path*, const char ** *attr*)

Decode the arguments of a Computing_Resource clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.5 void xml_close_declare (char * *path*, const char * *name*)

Close a xml clause.

Parameters:

path Clause path

name Clause name

Returns:

7.20.1.6 void xml_connect_hw_connection (struct hw_connection * *connection*)

7.20.1.7 void xml_connect_hw_group (struct hw_group * *group*)

7.20.1.8 void xml_connect_hw_instance (struct hw_instance * *instance*)

7.20.1.9 int xml_connect_sw_allocation (struct sw_allocation * *allocation*)

7.20.1.10 void xml_connect_sw_instance (struct sw_instance * *instance*)

7.20.1.11 int xml_declare_function (char * *path*, const char * *name*, const char ** *atts*)

Check the type of clause, and call the corresponding function to decode the arguments.

Parameters:

path Clause path
name Clause name
atts Clause attributes

Returns:

0 if ok, otherwise error

7.20.1.12 void xml_hw_component_string_convert (struct hw_component * *component*)

7.20.1.13 void xml_hw_connection_string_convert (struct hw_connection * *connection*)

7.20.1.14 void xml_hw_group_string_convert (struct hw_group * *group*)

7.20.1.15 void xml_hw_instance_string_convert (struct hw_instance * *instance*)

7.20.1.16 void* xml_load_exec_component (char * *path*, const char ** *attr*)

Decode the arguments of a Exec_Component clause.

Parameters:

path Clause path
name Clause name

Returns:

the created struct

7.20.1.17 void* xml_load_exec_instance (char * *path*, const char ** *attr*)

Decode the arguments of a Exec_Instance clause.

Parameters:

path Clause path
name Clause name

Returns:

the created struct

7.20.1.18 void* xml_load_hw_component (char * *path*, const char ** *attr*)

Decode the arguments of a HW_Component clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.19 void* xml_load_hw_connection (char * *path*, const char ** *attr*)

Decode the arguments of a HW_Connection clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.20 void* xml_load_hw_group (char * *path*, const char ** *attr*)

Decode the arguments of a Computing_Group clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.21 void* xml_load_hw_instance (char * *path*, const char ** *attr*)

Decode the arguments of a HW_Instance clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.22 void* xml_load_sw_component (char * *path*, const char ** *attr*)

Decode the arguments of a SW_Component clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.23 void* xml_load_sw_instance (char * *path*, const char ** *attr*)

Decode the arguments of a SW instance clause.

Parameters:

path Clause path

name Clause name

Returns:

the created struct

7.20.1.24 void xml_sw_allocation_string_convert (struct sw_allocation * *allocation*)

7.20.1.25 void xml_sw_instance_string_convert (struct sw_instance * *instance*)

7.20.1.26 void xml_sw_task_string_convert (struct sw_task * *task*)

7.20.2 Variable Documentation

7.20.2.1 int xml_info_level

7.21 xml_input.c File Reference

```
#include "xml_input.h"
#include "xml_list.h"
#include "xml_if.h"
#include <stdlib.h>
#include <string.h>
```

Functions

- char * xml_convert_path (const char *path)
  Adjust a searching path by ensuring the path does not contain a ".0".
- struct list_elem * get_hw_component_header ()
  Return the list of HW components.
- struct list_elem * get_hw_connection_header ()
  Return the list of HW connections.
- struct list_elem * get_sw_component_header ()
  Return the list of SW components.
- struct list_elem * get_sw_task_header ()
  Return the list of SW tasks.
- struct list_elem * get_hw_instance_header ()
  Return the list of HW instances.
- struct list_elem * get_sw_instance_header ()
  Return the list of SW instances.
- struct list_elem * get_sw_allocation_header ()
  Return the list of SW allocations.
- struct list_elem * get_hw_group_header ()
  Return the list of coomputing groups.
- void create_hw_component (struct hw_component *elem)
  Insert a new HW component into the corresponding list.
- struct hw_component * get_hw_component_by_name (const char *name)
  Return the component indicated by the input name.
- struct hw_component * get_hw_component_by_path (const char *path)
  Return the component indicated by the input path.
- void create_hw_instance (struct hw_instance *elem)
  Insert a new HW component into the corresponding list.
- struct hw_instance * get_hw_instance_by_name (const char *name)
  Return the component indicated by the input name.
- struct hw_instance * get_hw_instance_by_path (const char *path)
  Return the component indicated by the input path.
- void create_hw_connection (struct hw_connection *elem)
  Insert a new HW component into the corresponding list.

- void create_hw_group (struct hw_group *elem)
  Insert a new HW component into the corresponding list.
- struct hw_group * get_hw_group_by_name (const char *name)
  Return the component indicated by the input name.
- struct hw_group * get_hw_group_by_path (const char *path)
  Return the component indicated by the input path.
- void add_instance_to_hw_group (struct hw_group *group, struct hw_instance *elem)
  Add a HW instance to a computing group previously created.
- void create_sw_component (struct sw_component *elem)
  Insert a new HW component into the corresponding list.
- struct sw_component * get_sw_component_by_name (const char *name)
  Return the component indicated by the input name.
- struct sw_component * get_sw_component_by_path (const char *path)
  Return the component indicated by the input path.
- void create_sw_instance (struct sw_instance *elem)
  Insert a new HW component into the corresponding list.
- struct sw_instance * get_sw_instance_by_name (const char *name)
  Return the component indicated by the input name.
- struct sw_instance * get_sw_instance_by_path (const char *path)
  Return the component indicated by the input path.
- void create_sw_task (struct sw_task *elem)
  Insert a new HW component into the corresponding list.
- struct sw_task * get_sw_task_by_name (const char *name)
  Return the component indicated by the input name.
- struct sw_task * get_sw_task_by_path (const char *path)
  Return the component indicated by the input path.
- void create_sw_allocation (struct sw_allocation *elem)
  Insert a new HW component into the corresponding list.

Variables

- struct list_elem * hw_component_header = 0
- struct list_elem * hw_instance_header = 0
- struct list_elem * hw_connection_header = 0
- struct list_elem * hw_group_header = 0
- struct list_elem * sw_component_header = 0

- struct list_elem * sw_instance_header = 0
- struct list_elem * sw_task_header = 0
- struct list_elem * sw_allocation_header = 0

7.21.1 Function Documentation

7.21.1.1 void add_instance_to_hw_group (struct hw_group * *group*, struct hw_instance * *elem*)

Add a HW instance to a computing group previously created.

Parameters:

group The computing group

elem The component instance to be added

Returns:

7.21.1.2 void create_hw_component (struct hw_component * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.3 void create_hw_connection (struct hw_connection * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.4 void create_hw_group (struct hw_group * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.5 void create_hw_instance (struct hw_instance * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.6 void create_sw_allocation (struct sw_allocation * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.7 void create_sw_component (struct sw_component * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.8 void create_sw_instance (struct sw_instance * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.9 void create_sw_task (struct sw_task * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.21.1.10 struct hw_component* get_hw_component_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.21.1.11 struct hw_component* get_hw_component_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.21.1.12 struct list_elem* get_hw_component_header () [read]

Return the list of HW components.

Returns:

The list pointer

7.21.1.13 struct list_elem* get_hw_connection_header () [read]

Return the list of HW connections.

Returns:

The list pointer

7.21.1.14 struct hw_group* get_hw_group_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.21.1.15 struct hw_group* get_hw_group_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.21.1.16 struct list_elem* get_hw_group_header () [read]

Return the list of coomputing groups.

Returns:

The list pointer

7.21.1.17 struct hw_instance* get_hw_instance_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.21.1.18 struct hw_instance* get_hw_instance_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.21.1.19 struct list_elem* get_hw_instance_header () [read]

Return the list of HW instances.

Returns:

The list pointer

7.21.1.20 struct list_elem* get_sw_allocation_header () [read]

Return the list of SW allocations.

Returns:

The list pointer

7.21.1.21 struct sw_component* get_sw_component_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.21.1.22 struct sw_component* get_sw_component_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.21.1.23 struct list_elem* get_sw_component_header () [read]

Return the list of SW components.

Returns:

The list pointer

7.21.1.24 struct sw_instance* get_sw_instance_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.21.1.25 struct sw_instance* get_sw_instance_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.21.1.26 struct list_elem* get_sw_instance_header () [read]

Return the list of SW instances.

Returns:

The list pointer

7.21.1.27 struct sw_task* get_sw_task_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.21.1.28 struct sw_task* get_sw_task_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.21.1.29 struct list_elem* get_sw_task_header () [read]

Return the list of SW tasks.

Returns:

The list pointer

7.21.1.30 char* xml_convert_path (const char * *path*)

Adjust a searching path by ensuring the path does not contain a ".0".

Parameters:

path The original path

Returns:

The fixed path

7.21.2 Variable Documentation

7.21.2.1 struct list_elem* hw_component_header = 0

7.21.2.2 struct list_elem* hw_connection_header = 0

7.21.2.3 struct list_elem* hw_group_header = 0

7.21.2.4 struct list_elem* hw_instance_header = 0

7.21.2.5 struct list_elem* sw_allocation_header = 0

7.21.2.6 struct list_elem* sw_component_header = 0

7.21.2.7 struct list_elem* sw_instance_header = 0

7.21.2.8 struct list_elem* sw_task_header = 0

7.22 xml_input.h File Reference

Functions

- struct list_elem * get_hw_component_header ()
  Return the list of HW components.
- struct list_elem * get_hw_connection_header ()
  Return the list of HW connections.
- struct list_elem * get_sw_component_header ()
  Return the list of SW components.
- struct list_elem * get_sw_task_header ()
  Return the list of SW tasks.
- struct list_elem * get_hw_instance_header ()

Return the list of HW instances.

- struct list_elem * get_sw_instance_header ()
  Return the list of SW instances.
- struct list_elem * get_sw_allocation_header ()
  Return the list of SW allocations.
- struct list_elem * get_hw_group_header ()
  Return the list of coomputing groups.
- void create_hw_component (struct hw_component *elem)
  Insert a new HW component into the corresponding list.
- struct hw_component * get_hw_component_by_name (const char *name)
  Return the component indicated by the input name.
- struct hw_component * get_hw_component_by_path (const char *path)
  Return the component indicated by the input path.
- void create_hw_instance (struct hw_instance *elem)
  Insert a new HW component into the corresponding list.
- struct hw_instance * get_hw_instance_by_name (const char *name)
  Return the component indicated by the input name.
- struct hw_instance * get_hw_instance_by_path (const char *path)
  Return the component indicated by the input path.
- void create_hw_connection (struct hw_connection *elem)
  Insert a new HW component into the corresponding list.
- void create_hw_group (struct hw_group *elem)
  Insert a new HW component into the corresponding list.
- struct hw_group * get_hw_group_by_name (const char *name)
  Return the component indicated by the input name.
- struct hw_group * get_hw_group_by_path (const char *path)
  Return the component indicated by the input path.
- void add_instance_to_hw_group (struct hw_group *group, struct hw_instance *elem)
  Add a HW instance to a computing group previously created.
- void create_sw_component (struct sw_component *elem)
  Insert a new HW component into the corresponding list.
- struct sw_component * get_sw_component_by_name (const char *name)
  Return the component indicated by the input name.

- struct sw_component * get_sw_component_by_path (const char *path)
  Return the component indicated by the input path.
- void create_sw_instance (struct sw_instance *elem)
  Insert a new HW component into the corresponding list.
- struct sw_instance * get_sw_instance_by_name (const char *name)
  Return the component indicated by the input name.
- struct sw_instance * get_sw_instance_by_path (const char *path)
  Return the component indicated by the input path.
- void create_sw_task (struct sw_task *elem)
  Insert a new HW component into the corresponding list.
- struct sw_task * get_sw_task_by_name (const char *name)
  Return the component indicated by the input name.
- struct sw_task * get_sw_task_by_path (const char *path)
  Return the component indicated by the input path.
- void create_sw_allocation (struct sw_allocation *elem)
  Insert a new HW component into the corresponding list.

7.22.1 Function Documentation

7.22.1.1 void add_instance_to_hw_group (struct hw_group * *group*, struct hw_instance * *elem*)

Add a HW instance to a computing group previously created.

Parameters:

group The computing group

elem The component instance to be added

Returns:

7.22.1.2 void create_hw_component (struct hw_component * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.3 void create_hw_connection (struct hw_connection * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.4 void create_hw_group (struct hw_group * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.5 void create_hw_instance (struct hw_instance * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.6 void create_sw_allocation (struct sw_allocation * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.7 void create_sw_component (struct sw_component * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.8 void create_sw_instance (struct sw_instance * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.9 void create_sw_task (struct sw_task * *elem*)

Insert a new HW component into the corresponding list.

Parameters:

elem The new component

Returns:

7.22.1.10 struct hw_component* get_hw_component_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.22.1.11 struct hw_component* get_hw_component_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.22.1.12 struct list_elem* get_hw_component_header () [read]

Return the list of HW components.

Returns:

The list pointer

7.22.1.13 struct list_elem* get_hw_connection_header () [read]

Return the list of HW connections.

Returns:

The list pointer

7.22.1.14 struct hw_group* get_hw_group_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.22.1.15 struct hw_group* get_hw_group_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.22.1.16 struct list_elem* get_hw_group_header () [read]

Return the list of coomputing groups.

Returns:

The list pointer

7.22.1.17 struct hw_instance* get_hw_instance_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.22.1.18 struct hw_instance* get_hw_instance_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.22.1.19 struct list_elem* get_hw_instance_header () [read]

Return the list of HW instances.

Returns:

The list pointer

7.22.1.20 struct list_elem* get_sw_allocation_header () [read]

Return the list of SW allocations.

Returns:

The list pointer

7.22.1.21 struct sw_component* get_sw_component_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.22.1.22 struct sw_component* get_sw_component_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.22.1.23 struct list_elem* get_sw_component_header () [read]

Return the list of SW components.

Returns:

The list pointer

7.22.1.24 struct sw_instance* get_sw_instance_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.22.1.25 struct sw_instance* get_sw_instance_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.22.1.26 struct list_elem* get_sw_instance_header () [read]

Return the list of SW instances.

Returns:

The list pointer

7.22.1.27 struct sw_task* get_sw_task_by_name (const char * *name*) [read]

Return the component indicated by the input name.

Parameters:

name The component name

Returns:

the component

7.22.1.28 struct sw_task* get_sw_task_by_path (const char * *path*) [read]

Return the component indicated by the input path.

Parameters:

name The component path

Returns:

the component

7.22.1.29 struct list_elem* get_sw_task_header () [read]

Return the list of SW tasks.

Returns:

The list pointer

7.23 xml_list.c File Reference

```
#include "xml_list.h"
#include <stdlib.h>
```

Functions

- int xml_list_size (struct list_elem **header)
  Return the size of the list.
- void xml_list_add_element (struct list_elem **header, void *data)
  Add an new element to the end of the list.
- void * xml_list_last_element (struct list_elem *header)
  Peek the last element of the list.

7.23.1 Function Documentation

7.23.1.1 void xml_list_add_element (struct list_elem ** *header*, void * *data*)

Add an new element to the end of the list.

Parameters:

header Pointer to the list (pointer to the first element)

data Pointer to the new data to be stored

Returns:

7.23.1.2 void* xml_list_last_element (struct list_elem * *header*)

Peek the last element of the list.

Parameters:

header Pointer to the list (pointer to the first element)

return pointer to the last element of the list

7.23.1.3 int xml_list_size (struct list_elem ** *header*)

Return the size of the list.

Parameters:

header Pointer to the list (pointer to the first element)

Returns:

The list size

7.24 xml_list.h File Reference

Data Structures

- struct list_elem

Functions

- int xml_list_size (struct list_elem **)
  Return the size of the list.
- void xml_list_add_element (struct list_elem **, void *)
  Add an new element to the end of the list.
- void * xml_list_last_element (struct list_elem *)
  Peek the last element of the list.

7.24.1 Function Documentation

7.24.1.1 void xml_list_add_element (struct list_elem ** *header*, void * *data*)

Add an new element to the end of the list.

Parameters:

header Pointer to the list (pointer to the first element)

data Pointer to the new data to be stored

Returns:

7.24.1.2 void* xml_list_last_element (struct list_elem * *header*)

Peek the last element of the list.

Parameters:

header Pointer to the list (pointer to the first element)

return pointer to the last element of the list

7.24.1.3 int xml_list_size (struct list_elem ** *header*)

Return the size of the list.

Parameters:

header Pointer to the list (pointer to the first element)

Returns:

The list size

7.25 xml_main.c File Reference

```
#include "xml_main.h"
#include "xml_configuration_file.h"
#include "xml_hierarchy.h"
#include "xml_if.h"
#include <stdio.h>
#include <string.h>
#include "expat.h"
```

Functions

- int read_xml_file (char *file)

  Function read the XML System Description file.

Variables

- int xml_info_level = 0
- int xml_has_platform_description = 0

7.25.1 Function Documentation

7.25.1.1 int read_xml_file (char * *file*)

Function read the XML System Description file.

Parameters:

file Name of the file

Returns:

0 on success, otherwise error

7.25.2 Variable Documentation

7.25.2.1 int xml_has_platform_description = 0

7.25.2.2 int xml_info_level = 0

7.26 xml_main.h File Reference

Functions

- int read_xml_file (char *file)

  Function read the XML System Description file.

7.26.1 Function Documentation

7.26.1.1 int read_xml_file (char * *file*)

Function read the XML System Description file.

Parameters:

file Name of the file

Returns:

0 on success, otherwise error

7.27 xml_obtain_metrics.cpp File Reference

```
#include "xml_if.h"
#include "xml_input.h"
#include "xml_list.h"
#include <algorithm>
#include "xml_obtain_metrics.h"
#include "scuc_noc_sim_if.h"
```

Defines

- #define GET_INSTANCE_PARAMETER(instance, param)

Functions

- long double xml_get_tlm_power_consumption (UC_tlm_statistics *statistics, struct hw_instance *instance)
- long double calculate_latency ()
- unsigned long int calculate_ec ()
- unsigned long int calculate_ic ()
- unsigned long int calculate_mips ()
- long double calculate_hr ()
- unsigned long int calculate_msc ()
- long double calculate_amat ()
- int obtain_xml_metrics (MetricReader &mr)

7.27.1 Define Documentation

7.27.1.1 #define GET_INSTANCE_PARAMETER(instance, param)

Value:

```
(instance->param != NULL) ? atoi_null(instance->param) : \
((instance->component->param != NULL) ? atoi_null(instance->component->param) : 0)
```

7.27.2 Function Documentation

7.27.2.1 long double calculate_amat ()

7.27.2.2 unsigned long int calculate_ec ()

7.27.2.3 long double calculate_hr ()

7.27.2.4 unsigned long int calculate_ic ()

7.27.2.5 long double calculate_latency ()

7.27.2.6 unsigned long int calculate_mips ()

7.27.2.7 unsigned long int calculate_msc ()

7.27.2.8 int obtain_xml_metrics (MetricReader & *mr*)

7.27.2.9 long double xml_get_tlm_power_consumption (UC_tlm_statistics * *statistics*, struct hw_instance * *instance*)

7.28 xml_obtain_metrics.h File Reference

```
#include "sc_scope.h"
#include "xml_main.h"
#include "uc_load_xml.h"
```

Functions

- int obtain_xml_metrics (MetricReader &mr)

7.28.1 Function Documentation

7.28.1.1 int obtain_xml_metrics (MetricReader & *mr*)

Index