SUPPLIER DOCUMENT Cover Page

idm@F4E UID / VERSION
32SNQE / 2.3

VERSION CREATED ON / STATUS
09 January 2017 / Approved

EXTERNAL REFERENCE

Supplier Document

D4 MARTe Coding Standard

Under the Specific Contract no. F4E-OFC-361-06, with subject “Fast plant controller prototype”, this document is part of the deliverableD4 MARTe Coding Standard.

This document list the main rules applicable to the source code development in C++ for the artifacts linked to the MARTe framework. The main goal of these rules is to assure the coherence among the code developed by the different ...

Approval Process			
	Name	*Action*	*Affiliation*
Author	**Herrero I.**	**09 January 2017:signed**	
Co-Authors			
Reviewers			
Approver	**Cabrita Neto A.**	**09-Jan-2017: approved**	**ITERD**
RO: Cabrita Neto Andre (F4E)			
Read Access	**LG: GTD team, LG: OFC-361-06-CCFE, AD: IDM_F4E, AD: F4E-A40_HEAD, AD: I-CODAC, AD: IDM IE-TS-CO-00 CODAC, GG: IAC, GG: IAS Audit on Document Management, project administrator, RO**		

Orig. Document MD5#: 8DE5DEBAC08A8114ED72884E1E6626C5

Change Log			
D4 MARTe Coding Standard (32SNQE)			
Version	***Latest Status***	***Issue Date***	***Description of Change***
v1.0	Approved	27 February 2015	
v2.0	Revision Required	05 June 2015	This major release of the document is structured according to the following points: 0. Introduction: Introduces the document objective to the reader. 1. Coding rules: Explains how coding rules will be applied to MARTe framework development. 2. Documenting rules: Explains how documenting rules will be applied to MARTe framework development. 3. Naming rules: Explains how naming rules will be applied to MARTe framework development. 4. Formatting rules: Explains how formatting rules will be applied to MARTe framework development. 5. Code & editor templates: Explains how templates will be applied to MARTe framework development. 6. Annex. Coding rules compliance matrix: Lists the coding rules. 7. Annex. Coding rules deviation procedure: Lists the coding rules deviations. 8. Annex. Subset of tags for Doxygen: Lists the tags with their intended use for documenting code. 9. Annex. Configuration file for Doxygen: Lists the Doxygen configuration for MARTE framework. 10. Annex. Naming rules for Eclipse: Lists the naming rules for MARTE framework. 11. Annex. Formatting rules for Eclipse: Lists the formatting rules for MARTE framework. 12. Annex. Code templates for Eclipse: Lists the code templates for MARTE framework. 13. Annex. Editor templates for Eclipse: Lists the editor templates for MARTE framework.
v2.1	Approved	12 June 2015	Annex 8. Checked small typo by which @date format will be DD/MM/YYYY and not MM/DD/YYY.
v2.2	Approved	04 January 2017	Added rules for git commit messages. Updated list of doxygen tags on section “Annex. Subset of tags for Doxygen”. Updated templates on sections “Configuration file for Doxygen”, “Code templates for Eclipse”, and “Editor templates for Eclipse”.
v2.3	Approved	09 January 2017	Fix variable fields not properly refreshed at headers.

Identification of the document			
Document Reference	01.ES-F4E-OFC-361-06 D4	Revision	2.3
F4E Reference	F4E_D_32SNQE	F4E TRO	André C. Neto
F4E Customer Reference	N/A		
Date	2017-01-09		
Supplier	GTD SISTEMAS DE INFORMACION		
Graded Quality level	Class 3 – Any safety related item (SR) or non safety related item (NSR) whose failure could result in MODERATE impact.		

MARTe Coding Standard

Fast plant controller prototype (F4E-OFC-361-06)

Summary

Under the Specific Contract no. F4E-OFC-361-06, with subject “Fast plant controller prototype”, this document is part of the deliverable **D4.a MARTe Coding Standard**.

This document list the main rules applicable to the source code development in C++ for the artifacts linked to the MARTe framework. The main goal of these rules is to assure the coherence among the code developed by the different team members and also assure its quality and maintainability.

	Written by	Revised by	Approved by
Name	Ivan Herrero	Ivan Herrero	Javier Varas
Signature			
Date	2017-01-09	2017-01-09	2017-01-09

Contact person GTD: javier.varas@gtd.eu
Phone: +34 93 493 93 00

DISSEMINATION

Distributed to	Copies	Means
Project Team	1	e-mail
Fusion for Energy	1	Electronic

SUMMARY OF MODIFICATIONS

Edition	Date	Chapter	Modification	Author/s
1.0	23/02/2015	All	Document creation	JT
2.0	05/06/2015	All	Document evolution	IH
2.1	12/06/2015	Annex 8	Checked small typo by which @date format will be DD/MM/YYYY and not MM/DD/YYY	HN
2.2	04/01/2017	5. Commit rules Annex 8 Annex 9 Annex 12 Annex 13	Added rules for git commit messages. Updated list of doxygen tags on section “Annex. Subset of tags for Doxygen”. Updated templates on sections “Configuration file for Doxygen”, “Code templates for Eclipse”, and “Editor templates for Eclipse”.	IH
2.3	09/01/2017	Headers	Fix variable fields not properly refreshed.	IH

CONTENTS

INDEX OF TABLES

0 INTRODUCTION

0.1 Purpose and scope of the document

Under the Specific Contract no. F4E-OFC-361-06, with subject “Fast plant controller prototype”, this document is part of the deliverable **D4.a MARTe Coding Standard (version 1)**.

This document list the main rules applicable to the source code development in C++ for the artifacts linked to the MARTe framework. The main goal of these rules is to assure the coherence among the code developed by the different team members and also assure its quality and maintainability.

0.2 Applicable and reference documents

Ref.	Code	Date	Ver.	Description

Table 1 Applicable documents

Ref.	Code	Date	Ver.	Description
RD1	ISO/IEC 14882:2003	2003		ISO/IEC. (2003). *ISO International Standard ISO/IEC 14882:2003(E) – Programming Languages -- C++*. Geneva, Switzerland: International Organization for Standardization (ISO).
RD2	MISRA C++:2008	2008		MISRA C++:2008 *Guidelines for the Use of the C++ Language in Critical Systems*, ISBN 978-906400-03-3 (paperback), ISBN 978-906400-04-0 (PDF),

Table 2 Reference documents

0.3 Document structure

This document is structured according to the following points:

0. **Introduction**: Introduces the document objective to the reader.
1. **Coding rules**: Explains how coding rules will be applied to MARTe framework development.
2. **Documenting rules**: Explains how documenting rules will be applied to MARTe framework development.
3. **Naming rules:** Explains how naming rules will be applied to MARTe framework development.
4. **Formatting rules:** Explains how formatting rules will be applied to MARTe framework development.
5. **Code & editor templates:** Explains how templates will be applied to MARTe framework development.
6. **Annex. Coding rules compliance matrix:** Lists the coding rules.
7. **Annex. Coding rules deviation procedure:** Lists the coding rules deviations.
8. **Annex. Subset of tags for Doxygen:** Lists the tags with their intended use for documenting code.
9. **Annex. Configuration file for Doxygen:** Lists the Doxygen configuration for MARTE framework.
10. **Annex. Naming rules for Eclipse:** Lists the naming rules for MARTE framework.
11. **Annex. Formatting rules for Eclipse:** Lists the formatting rules for MARTE framework.
12. **Annex. Code templates for Eclipse:** Lists the code templates for MARTE framework.
13. **Annex. Editor templates for Eclipse:** Lists the editor templates for MARTE framework.

1 CODING RULES

1.1 Description

In order to develop robust code and to avoid common errors and pitfalls, a controlled subset of the C++ language must be defined for the MARTe framework. This subset will be defined by means of a list of coding rules, which will address all dangerous aspects of the C++ language for critical systems. Thus, the C++ version used on MARTe will be that defined by the standard ISO/IEC 14882:2003 aka as C++03, while the coding rules will be those defined by the standard MISRA C++:2008.

The MISRA C++:2008 is a set of software development guidelines for the C++ language targeted towards critical systems, which applies to the C++ language defined by the standard ISO/IEC 14882:2003. MISRA C++:2008 has emerged from the automotive industry, and is widely accepted as a model for best practices in sectors like aerospace, telecom, medical devices, defense, railway and others. MISRA C++:2008 also takes into account all those features of the language that are unspecified, undefined, etc in the ISO/IEC 14882:2003.

Note 1: The definition of the C++03 language is large and subject to copyright, so it cannot be reproduced here. For a complete specification of C++03, refer directly to the standard ISO/IEC 14882:2003 [RD1].

Note 2: The definition of the of MISRA C++:2008 rules is large and also subject to copyright, so it cannot neither be reproduced here. For a full listing of MISRA C++:2008 rules refer to [2].

1.2 Compliance

All the coding rules must be followed by developers at coding time, but formal reviews are also imperative to ensure that all written code does conform to them. It is strongly recommended to use static checking tools for detecting rules' violations on code in an automatic fashion, but for those which are not automatable a manual review will be needed.

In order to ensure that all rules are covered by the reviews; it is advisable to produce a compliance matrix which will list all rules indicating for each one how it is to be checked. At annex §9 "Annex. Coding rules compliance matrix" there is the full compliance matrix for MARTe, which follows the style shown in the following example:

Rule	Compiler 1	Compiler 2	Checking tool 1	Checking tool 2	Manual review
Rule 0–1–1	Warning 347				
Rule 0–1–2		Error 25			
Rule 0–1–3			Message 28		
Rule 0–1–4				Warning 97	
Rule 0–1–5					Procedure X.Y

Table 3 Compliance matrix example

2 NOTE: TO FACILITATE WRITING THE RIGHT TAGS FROM THE BEGINNING, ECLIPSE'S CODE AND EDITOR TEMPLATES WILL PROVIDE THE

INITIAL VERSION OF SOURCE CODE FILES WITH SKELETONS FOR COMMENTS. SEE CHAPTER §7 “COMMIT RULES

2.1 Description

In order to make more readable the repository's history, the following guidelines for commit messages are given.

Always write, at least, the subject of the commit on the first line. The subject must be capitalised and written in imperative mood. Moreover, it is advisable to use no more than 50 characters approximately. The imperative mood helps reading a list of commits as a sequence of commands to be applied to the repository. The subject can optionally be prefixed by the identifier of the user story, if applicable.

If more details are needed, and the subject is not enough, then a free from body of multiple lines can be added, separating it from the subject by a blank line. The body can use markdown elements for creating lists, etc.

The body can explain aspects like: why a change was necessary, how the change addresses the issue, what changed, why you made the change the way you did, consequences of the change, etc.

```
#id - Subject max. 50 chars.

Line 1 of body.
Line 2 of body.
…
Line N of body.
```

2.2 Examples

```
#230 - Fix namespace dependency propagation on Object.h macros

- Move macros CLASS_REGISTER_DECLARATION and CLASS_REGISTER out of MARTe
namespace declaration.
- Add "MARTe::" prefix on all uses of MARTe symbols inside macros
CLASS_REGISTER_DECLARATION and CLASS_REGISTER.
```

Code & editor templates”.

2.3 Exceptions

Although all rules are a good advice, a strict adherence to all of them is unlikely for many reasons. For example, the rule 0–1–5 is not applicable to a framework like MARTe, because it says that a project shall not contain unused type declarations, but this is incompatible with a framework because one of its main purposes is to offer types to users of the framework, not to itself.

To address these issues, a deviation procedure must be created, providing for each rule a detailed justification of its violation. These deviations can also be categorized as project deviation and specific deviations. At annex §0 “**Table 5** Coding rules compliance matrix

Annex. Coding rules' deviations" there is the full deviation procedure for MARTe, which follows the style shown in the following example:

Rule	Classification	Deviation
Rule 0–1–5	(Required)	N/A because it is a framework

Table 4 Project deviation table example

3 DOCUMENTING RULES

3.1 Description

In order to make code more readable and consistent among all source files, it is not only necessary to document them, but to define documentation rules for them. For example, a rule could be that each class shall have a brief and a detailed description or that each file shall have the project's copyright statement, author's name, and so on.

Because C++ does not define a structured format for comments, we will use the well-known Doxygen syntax, which is used to document source code files in C++ and many other languages, using a set of reserved tags prefixed with the "@" symbol. These tags will be put into C++ compatible comments, so C++ files will be compiled with no interferences, but at the same time these comments will be detachable, allowing the automated generation of framework's documentation in many formats like HTML, PDF, etc.

The subset of Doxygen's tags expected in the source code files is defined in the annex §11 "**Table 7** Specific deviations Annex. Subset of tags for Doxygen", which has a table with all these tags along with an explanation of their intended use and order. Remember that, by convention, Doxygen tags have to be put into C++ multi-line comments beginning with "/**" instead of "/*" and that these comments will have to precede the target to document (class, method, attribute ...).

3.2 Compliance

All the documenting rules must be followed by developers at coding time, but formal reviews are also imperative to ensure that all written code does conform to them. All these reviews will be manual and will ensure not only the appearance of the tags in the right order at the right place, but that their contents explain what it is supposed to explain.

4 NOTE: TO FACILITATE WRITING THE RIGHT TAGS FROM THE BEGINNING, ECLIPSE'S CODE AND EDITOR TEMPLATES WILL PROVIDE THE INITIAL VERSION OF SOURCE CODE FILES WITH SKELETONS FOR COMMENTS. SEE CHAPTER §7 "COMMIT RULES

4.1 Description

In order to make more readable the repository's history, the following guidelines for commit messages are given.

Always write, at least, the subject of the commit on the first line. The subject must be capitalised and written in imperative mood. Moreover, it is advisable to use no more than 50 characters approximately. The imperative mood helps reading a list of commits as a sequence of commands to be applied to the repository. The subject can optionally be prefixed by the identifier of the user story, if applicable.

If more details are needed, and the subject is not enough, then a free from body of multiple lines can be added, separating it from the subject by a blank line. The body can use markdown elements for creating lists, etc.

The body can explain aspects like: why a change was necessary, how the change addresses the issue, what changed, why you made the change the way you did, consequences of the change, etc.

```
#id - Subject max. 50 chars.

Line 1 of body.
Line 2 of body.
…
Line N of body.
```

4.2 Examples

```
#230 - Fix namespace dependency propagation on Object.h macros

- Move macros CLASS_REGISTER_DECLARATION and CLASS_REGISTER out of MARTe
namespace declaration.
- Add "MARTe::" prefix on all uses of MARTe symbols inside macros
CLASS_REGISTER_DECLARATION and CLASS_REGISTER.
```

Code & editor templates".

4.3 Exceptions

It is expected to put all the comments as close as possible to the code they comment, always preceding the target to document (before the class, before the attribute, before the method, etc). Nevertheless, for namespaces is better to define their comments in file apart named *namespaces.dox* and let namespace declarations on source code without comments, otherwise it would be necessary to repeat the namespace's comment on each appearance of the namespace.

Note: When using tags far away from its target, remember that is mandatory to use a special tag to link the subsequent tags with the expected target (in the case of namespaces, precede the tag @brief with the tag @namespace).

4.4 Generation

The automated generation of documentation from source code comments is made by means of the doxygen program, which can be parameterized with a configuration file suffixed with ".doxyfile" by convention. This file will be named *marte.doxyfile* and it will contain a sequence of properties where each one is a configuration aspect which will be used by the generator. The annex §12 "**Table 8** Tags for Doxygen

Annex. Configuration file for Doxygen" has the contents of such doxyfile file, which will be considered the baseline for the generator configuration. It is out of scope of this document how to handle it into a configuration management system.

4.5 Examples

Next, an example of a C++ class with doxygen tags, where it is implicit that tags belong to the class because they appear right before it:

```
/**
 * @brief Unbounded stack of doubles
 * @details This class represents a stack of doubles, that is, a data
 * structure which follows a LIFO protocol. It offers methods to query
 * its count and top element as well as methods to add and remove
 * elements on its top.
 */
class Stack: public Streamable {
…
}
```

Next, an example of a C++ method class with doxygen tags, where it is implicit that tags belong to the method because they appear right before it:

```
/**
 * @brief Retrieves the element on the top of the stack
 * @details This method retrieves a copy of the element on
 * the top of the stack. It does not modify the stack.
 * @return A copy of the element on top of the stack
 */
double Top(void) const;
```

Next, an example of another C++ method class with doxygen tags, where it is implicit that tags belong to the method because they appear right before it:

```
/**
 * @brief Adds a new element on top of the stack
 * @details This method adds a new element on the top
 * of the stack and maintains the existing elements
 * with the same order.
 * @param[in] e The element to put on top of the stack
 */
void Push(const double e);
```

Finally, an example of doxygen tags for a couple of namespaces, which are defined separately from the namespaces themselves, so they need to refer to the target namespace with the special tag "@namespace":

```
/**
```

```
 * @namespace marte
 * @brief The root namespace for MARTe
 */

/**
 * @namespace marte::core
 * @brief Namespace for core services
 * @details This namespace groups all core related classes with a focus on
 * high performance and small footprint.
 */
```

5 NAMING RULES

5.1 Description

In order to make code more readable and consistent among all source files, it is necessary to define naming rules for them. Examples of rules can be letter case for class names or extensions for files.

The Eclipse platform, and more specifically Eclipse CDT, offers customizable name style settings for two categories: code and files, being code name style settings applied to constants, variables, etc, and file name style settings to header and source files.

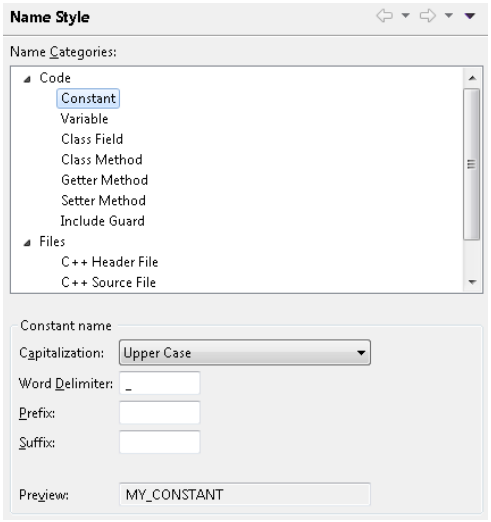
Eclipse allows defining a fixed list of settings for each name, which can only be defined at workspace level. These name style settings can define capitalization (CamelCase, …), prefixes, suffixes (.cpp, …), and so on.

Given that there is no option to define these settings with a configuration file, they must be configured by hand using the Eclipse settings screen. At annex §0 "

Annex. Naming rules for Eclipse" there is a table with the code name style settings and another with the file name style ones.

5.2 Examples

Next, an example of a name style setting for C++ configured through Eclipse settings screen:



6 FORMATTING RULES

6.1 Description

In order to make code more readable and consistent among all source files, it is necessary to define formatting rules for them. Examples of rules are tab size, maximum line width or opening brace position. If the file editor used is capable of auto format the source code, then it is advisable to define the formatting rules in a language compatible with the editor, so developers can format and reformat files effortless at any time.

The Eclipse platform, and more specifically Eclipse CDT, offers a customizable formatter which offers four formatting profiles: “K&R”, “BSD/Allman”, “GNU”, and “Whitesmiths”. The default is “K&R” which seems to be the preferred style among Linux programmers. The main feature of this style is to write the opening brace on the same line as the expression or key word that introduces the block. Another popular style is “BSD/Allman” which in the case of braces writes opening ones on the next line.

Eclipse allows defining many profiles, but only one active formatting profile can be active at any time, which is the one that will be applied when formatting a file. These profiles can be defined at workspace and project level.

All these formatting rules must be defined into a single XML file which can be imported into Eclipse. This file will be named *marte_cpp_formatting_rules.xml* and it will contain the identification of the profile as well as a list with all the settings’ values to be used by the formatter. The annex §0 “**Table 10** Eclipse: filename styles

Annex. Formatting rules for Eclipse" has the contents of such XML file, which will be considered the baseline for the formatting rules. It is out of scope of this document how to handle it into a configuration management system.

Note 1: The user must execute the formatter file-by-file and save them. If the formatting rules change, then it is mandatory to execute the formatter again for all files.
Note 2: The formatting rules are stored on workspace/project metadata. If these rules are changed, remember that there is no automatic sync between workspace/project metadata and the XML file with the formatting rules (a manual import or export will be needed).

6.2 Examples

Next, an example of a formatting setting for C++:

```
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_block" value="next_line"/>
```

With "next_line_value" the code will be formatted as shown next:

```
double Point::distance(const Point& other) const
{
    double dx = x - other.x;
    double dy = y - other.y;
    return sqrt(dx * dx + dy * dy);
}
```

But if we change the value of the setting as follows:

```
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_method_declaration" value="end_of_line"/>
```

Then, with "end_of_line" the code will be formatted as shown next:

```
double Point::distance(const Point& other) const {
    double dx = x - other.x;
    double dy = y - other.y;
    return sqrt(dx * dx + dy * dy);
}
```

7 COMMIT RULES

7.1 Description

In order to make more readable the repository's history, the following guidelines for commit messages are given.

Always write, at least, the subject of the commit on the first line. The subject must be capitalised and written in imperative mood. Moreover, it is advisable to use no more than 50 characters approximately. The imperative mood helps reading a list of commits as a sequence of commands to be applied to the repository. The subject can optionally be prefixed by the identifier of the user story, if applicable.

If more details are needed, and the subject is not enough, then a free from body of multiple lines can be added, separating it from the subject by a blank line. The body can use markdown elements for creating lists, etc.

The body can explain aspects like: why a change was necessary, how the change addresses the issue, what changed, why you made the change the way you did, consequences of the change, etc.

```
#id - Subject max. 50 chars.
```

```
Line 1 of body.
Line 2 of body.
…
Line N of body.
```

7.2 Examples

```
#230 - Fix namespace dependency propagation on Object.h macros

- Move macros CLASS_REGISTER_DECLARATION and CLASS_REGISTER out of MARTe
namespace declaration.
- Add "MARTe::" prefix on all uses of MARTe symbols inside macros
CLASS_REGISTER_DECLARATION and CLASS_REGISTER.
```

8 CODE & EDITOR TEMPLATES

8.1 Description

In order to ease the creation of well-formed and well-structured source files from the beginning, it is advisable to supply file templates from which new files will be created. If the file editor allows generating fragments of code by means of wizards or keyboard shortcuts it is also advisable to supply templates at fragment level.

The Eclipse platform, and more specifically Eclipse CDT, allows defining templates for files (code templates) and fragments (editor templates) which are used by actions that generate code and/or source files.

Eclipse allows defining one template for each file type, which are the ones that will be used when creating new files. These templates can only be defined at workspace level.

All the file templates are defined into a single XML file which can be imported into Eclipse. This file will be named *marte_cpp_code_templates.xml* and it will contain a sequence of templates where each one can contain fixed text and variables which will be substituted by an actual value when the template will be applied. The annex §0 “

Annex. Code templates for Eclipse" has the contents of such XML file, which will be considered the baseline for the file templates. It is out of scope of this document how to handle it into a configuration management system.

All the fragment templates are defined into a single XML file which can also be imported into Eclipse. This file will be named *marte_cpp_editor_templates.xml* and it will contain a sequence of templates where each one can contain fixed text and variables which will be substituted by an actual value when the template will be applied. The annex §0 "

Annex. Editor templates for Eclipse" has the contents of such XML file, which will be considered the baseline for the fragment templates. It is out of scope of this document how to handle it into a configuration management system.

Note: The code and editor templates are stored on workspace metadata. If these templates are changed, remember that there is no automatic sync between workspace metadata and the XML file with the code templates (a manual import or export will be needed).

8.2 Examples

Next, an example of a template for a C++ source file:

```
<template autoinsert="true"
        context="org.eclipse.cdt.core.cxxHeader.contenttype_context" deleted="false"
        description="Default template for newly created C++ header files"
        enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.cppheaderfile"
        name="Default C++ header template">
        ${filecomment}
        #ifndef ${include_guard_symbol}
        #define ${include_guard_symbol}
        ${includes}
        ${namespace_begin}
        ${declarations}
        ${namespace_end}
        #endif /* ${include_guard_symbol} */
</template>
```

This template, called "Default C++ header template", uses other templates like filecomment, which is shown next:

```
<template autoinsert="true"
        context="org.eclipse.cdt.ui.text.codetemplates.filecomment_context"
        deleted="false" description="Comment for created C/C++ files" enabled="true"
        id="org.eclipse.cdt.ui.text.codetemplates.filecomment" name="filecomment">
        /*
        * ${file_name}
        *
        * Created on: ${date}
        * Author: ${user}
        */
</template>
```

If we create a new file named *DemoSourceFile.h* based on "Default C++ header template", we will get the file shown next:

```
/*
```

```
 * DemoSourceFile.h
 *
 * Created on: 26/2/2015
 * Author: John Smith
 */

#ifndef DEMOSOURCEFILE_H_
#define DEMOSOURCEFILE_H_
…
#endif /* DEMOSOURCEFILE_H_ */
```

8.3 Remarks

- The syntax of variables is a dollar followed by the name of the variable between braces (e.g. ${user}).
- If it is needed to insert a literal dollar into a template, one has to escape it with another dollar (i.e. “$$”).
- Some variables may be substituted with values formatted with the user locale rules (e.g. ${date}).

9 ANNEX. CODING RULES COMPLIANCE MATRIX

Rule	GCC	MSC	PC-Lint	Manual review
Rule 0–1–1				
Rule 0–1–2				
Rule 0–1–3				
Rule 0–1–4				
Rule 0–1–5				
Rule 0–1–6				
Rule 0–1–7				
Rule 0–1–8				
Rule 0–1–9				
Rule 0–1–10				
Rule 0–1–11				
Rule 0–1–12				
Rule 0–2–1				
Rule 0–3–1				
Rule 0–3–2				
Rule 0–4–1				
Rule 0–4–2				
Rule 0–4–3				
Rule 1–0–1				
Rule 1–0–2				
Rule 1–0–3				
Rule 2–2–1				
Rule 2–3–1				
Rule 2–5–1				
Rule 2–7–1				
Rule 2–7–2				
Rule 2–7–3				
Rule 2–10–1				
Rule 2–10–2				
Rule 2–10–3				
Rule 2–10–4				
Rule 2–10–5				
Rule 2–10–6				
Rule 2–13–1				
Rule 2–13–2				
Rule 2–13–3				
Rule 2–13–4				
Rule 2–13–5				

Rule	GCC	MSC	PC-Lint	Manual review
Rule 3–1–1				
Rule 3–1–2				
Rule 3–1–3				
Rule 3–2–1				
Rule 3–2–2				
Rule 3–2–3				
Rule 3–2–4				
Rule 3–3–1				
Rule 3–3–2				
Rule 3–4–1				
Rule 3–9–1				
Rule 3–9–2				
Rule 3–9–3				
Rule 4–5–1				
Rule 4–5–2				
Rule 4–5–3				
Rule 4–10–1				
Rule 4–10–2				
Rule 5–0–1				
Rule 5–0–2				
Rule 5–0–3				
Rule 5–0–4				
Rule 5–0–5				
Rule 5–0–6				
Rule 5–0–7				
Rule 5–0–8				
Rule 5–0–9				
Rule 5–0–10				
Rule 5–0–11				
Rule 5–0–12				
Rule 5–0–13				
Rule 5–0–14				
Rule 5–0–15				
Rule 5–0–16				
Rule 5–0–17				
Rule 5–0–18				
Rule 5–0–19				
Rule 5–0–20				
Rule 5–0–21				
Rule 5–2–1				
Rule 5–2–2				

Rule	GCC	MSC	PC-Lint	Manual review
Rule 5–2–3				
Rule 5–2–4				
Rule 5–2–5				
Rule 5–2–6				
Rule 5–2–7				
Rule 5–2–8				
Rule 5–2–9				
Rule 5–2–10				
Rule 5–2–11				
Rule 5–2–12				
Rule 5–3–1				
Rule 5–3–2				
Rule 5–3–3				
Rule 5–3–4				
Rule 5–8–1				
Rule 5–14–1				
Rule 5–17–1				
Rule 5–18–1				
Rule 5–19–1				
Rule 6–2–1				
Rule 6–2–2				
Rule 6–2–3				
Rule 6–3–1				
Rule 6–4–1				
Rule 6–4–2				
Rule 6–4–3				
Rule 6–4–4				
Rule 6–4–5				
Rule 6–4–6				
Rule 6–4–7				
Rule 6–4–8				
Rule 6–5–1				
Rule 6–5–2				
Rule 6–5–3				
Rule 6–5–4				
Rule 6–5–5				
Rule 6–5–6				
Rule 6–6–1				
Rule 6–6–2				
Rule 6–6–3				
Rule 6–6–4				

Rule	GCC	MSC	PC-Lint	Manual review
Rule 6–6–5				
Rule 7–1–1				
Rule 7–1–2				
Rule 7–2–1				
Rule 7–3–1				
Rule 7–3–2				
Rule 7–3–3				
Rule 7–3–4				
Rule 7–3–5				
Rule 7–3–6				
Rule 7–4–1				
Rule 7–4–2				
Rule 7–4–3				
Rule 7–5–1				
Rule 7–5–2				
Rule 7–5–3				
Rule 7–5–4				
Rule 8–0–1				
Rule 8–3–1				
Rule 8–4–1				
Rule 8–4–2				
Rule 8–4–3				
Rule 8–4–4				
Rule 8–5–1				
Rule 8–5–2				
Rule 8–5–3				
Rule 9–3–1				
Rule 9–3–2				
Rule 9–3–3				
Rule 9–5–1				
Rule 9–6–1				
Rule 9–6–2				
Rule 9–6–3				
Rule 9–6–4				
Rule 10–1–1				
Rule 10–1–2				
Rule 10–1–3				
Rule 10–2–1				
Rule 10–3–1				
Rule 10–3–2				
Rule 10–3–3				

Rule	GCC	MSC	PC-Lint	Manual review
Rule 11–0–1				
Rule 12–1–1				
Rule 12–1–2				
Rule 12–1–3				
Rule 12–8–1				
Rule 12–8–2				
Rule 14–5–1				
Rule 14–5–2				
Rule 14–5–3				
Rule 14–6–1				
Rule 14–6–2				
Rule 14–7–1				
Rule 14–7–2				
Rule 14–7–3				
Rule 14–8–1				
Rule 14–8–2				
Rule 15–0–1				
Rule 15–0–2				
Rule 15–0–3				
Rule 15–1–1				
Rule 15–1–2				
Rule 15–1–3				
Rule 15–3–1				
Rule 15–3–2				
Rule 15–3–3				
Rule 15–3–4				
Rule 15–3–5				
Rule 15–3–6				
Rule 15–3–7				
Rule 15–4–1				
Rule 15–5–1				
Rule 15–5–2				
Rule 15–5–3				
Rule 16–0–1				
Rule 16–0–2				
Rule 16–0–3				
Rule 16–0–4				
Rule 16–0–5				
Rule 16–0–6				
Rule 16–0–7				
Rule 16–0–8				

Rule	GCC	MSC	PC-Lint	Manual review
Rule 16–1–1				
Rule 16–1–2				
Rule 16–2–1				
Rule 16–2–2				
Rule 16–2–3				
Rule 16–2–4				
Rule 16–2–5				
Rule 16–2–6				
Rule 16–3–1				
Rule 16–3–2				
Rule 16–6–1				
Rule 17–0–1				
Rule 17–0–2				
Rule 17–0–3				
Rule 17–0–4				
Rule 17–0–5				
Rule 18–0–1				
Rule 18–0–2				
Rule 18–0–3				
Rule 18–0–4				
Rule 18–0–5				
Rule 18–2–1				
Rule 18–4–1				
Rule 18–7–1				
Rule 19–3–1				
Rule 27–0–1				

Table 5 Coding rules compliance matrix

10ANNEX. CODING RULES' DEVIATIONS

10.1Project deviations

Rule	Classification	Deviation
Rule 0–1–5	(Required)	N/A because it is a framework
Rule 0–1–10	(Required)	N/A because it is a framework
Rule 0–1–12	(Required)	N/A because it is a framework
Rule 0–4–3	(Document)	IEEE 754 will be used as a floating-point standard.
Rule 1–0–2	(Document)	Justify it (because one compiler for platform is used)
Rule 1–0–3	(Document)	Justify it by documenting gcc and msc do
Rule 2–2–1	(Document)	UTF-8 will be used as character set.
Rule 2–13–3	(Required)	Check if we can comply with it
Rule 3–9–3	(Required)	Check if we need to relax it for printf
Rule 4–5–3	(Required)	We agree rule, but code must be corrected
Rule 5–2–8	(Required)	Perhaps we will need to relax in some places (DDB buffer)
Rule 5–2–9	(Advisory)	Check it and relax if we need
Rule 5–8–1	(Required)	Check if it is used in MARTe
Rule 6–6–5	(Required)	Analyse deeper this
Rule 9–5–1	(Required)	Check it on code
Rule 12–1–3	(Required)	Relax this for AnyType class.
Rule 12–8–1	(Required)	Check or relax for smart pointer
Rule 14–7–1	(Required)	N/A because it is a framework
Rule 14–7–2	(Required)	Check it at GetTokenFromStream
Rule 16–0–4	(Required)	Heavily used in MARTe for object initialisation
Rule 16–2–1	(Required)	Relax it for objects creation
Rule 16–2–2	(Required)	Relax it for objects creation
Rule 16–3–1	(Required)	Check if it occurs at object management
Rule 16–3–2	(Advisory)	Check if it is used at object management
Rule 18–0–1	(Required)	Relax it at operating system layer
Rule 18–0–2	(Required)	Relax it at operating system layer
Rule 18–4–1	(Required)	Not complaint because it is a core feature of MARTe
Rule 19–3–1	(Required)	Relax it for streams
Rule 27–0–1	(Required)	Relax it for streams

Table 6 Project deviations for the coding rules

10.2Specific deviations

Rule	Classification	Scope	Deviation
		Core	
		GAM's	
		…	

Table 7 Specific deviations

11ANNEX. SUBSET OF TAGS FOR DOXYGEN

This annex lists the subset of Doxygen tags that are expected to be used in the source code files of the project.

Note: There are repeated tags because they can be used in different scopes.

Scope	Tag		Place	Description
Header file	@file	Mandatory	First tag	The base name of the header file. Example: “Stack.h”
Header file	@brief	Mandatory	After @file	The purpose of the header file, explained into a single line paragraph. Example: “Header file for class Stack”
Header file	@date	Mandatory	After @brief	The creation date of the header file in format DD/MM/YYY. Remark: Pay attention to locale options in auto generated dates from templates.
Header file	@author	Mandatory	After @date	The name of the original author of the header file. If more authors have contributed to it, then put subsequent @author tags.
Header file	@copyright	Mandatory	After @author	The copyright and license of the header file which can span multiple lines.
Header file	@warning	Mandatory	After @copyright	A warning about warranties and license.
Header file	@details	Mandatory	After @warning	A description of the header file that can span multiple lines of a paragraph. If more paragraphs are needed, then put subsequent @details tags. This description shall explain what the file contains, mainly the declaration of the class.
Source file	@file	Mandatory	First tag	The base name of the source file. Example: “Stack.cpp”
Source file	@brief	Mandatory	After @file	The purpose of the source file, explained into a single line paragraph. Example: “Source file for class Stack”
Source file	@date	Mandatory	After @brief	The creation date of the source file in format DD/MM/YYYY. Remark: Pay attention to locale options in auto generated dates from templates.
Source file	@author	Mandatory	After @date	The name of the original author of the source file. If more authors have contributed to it, then put subsequent @author tags.
Source file	@copyright	Mandatory	After @author	The copyright and license of the source file which can span multiple lines.
Source file	@warning	Mandatory	After @copyright	A warning about warranties and license.
Source file	@details	Mandatory	After @warning	A description of the header file that can span multiple lines of a paragraph. If more paragraphs are needed, then put subsequent @details tags.

Scope	Tag		Place	Description
				This description shall explain what the file contains, mainly the definition of the class' methods.
Namespace	@brief	Mandatory	First tag	The purpose of the namespace, explained into a single line paragraph.
Namespace	@details	Mandatory	After @brief	A description of the namespace that can span multiple lines of a paragraph. If more paragraphs are needed, then put subsequent @details tags. This description shall explain what the namespace is expected to contain.
Class	@brief	Mandatory	First tag	The main concept that the class represents explained into a single line paragraph. It will usually be the name of the class "expanded". Think of it as a tooltip. Example: "Unbounded stack of doubles"
Class	@details	Mandatory	After @brief	A detailed description of the class that can span multiple lines of a paragraph. If more paragraphs are needed, then put subsequent @details tags. This description shall explain the concept which the class represents alongside its responsibilities or services it provides. Example: "This class represents a stack of doubles, that is, a data structure which … "
Class	@tparam	Mandatory if the class is a template with parameters	After @details	Each template parameter must have its own @tparam tag, which have to include its name and a description. The name must be the same of the template parameter. The description must be a brief explanation intended to fit into a single line, but it can span multiple lines if needed. Think of it as a tooltip.
Class	@invariant	Optional	After @tparam	The public invariant of the class. It can be described formally or informally. Example: "Count() >= 0" or "The number of elements on the stack is always greater or equal than zero"
Class	@remark	Optional	After @invariant	A clarifying comment about the class which could have been included in other tags, like details or @tparams, but needs to be explicitly noticed.
Class	@warning	Optional	After @remark	An advisory notice about class restrictions or behavior, which can have undesired or unexpected side effects.
Class	@verbatim / @endverbatim	Optional	Inline in any tag	A text to be verbatim included, delimited by a starting @verbatim and an ending @endverbatim.
Attribute	--	Mandatory	Whole comment	The role of the member in the class explained briefly into a single paragraph, which can span multiple lines. Think of it as a tooltip.
Method	@brief	Mandatory	First tag	The action that the method does, explained into a single line paragraph. Think of it as a tooltip. As methods are usually a query, a modifier or a manager, they can be expressed like the following

Scope	Tag		Place	Description
				examples: • A boolean query: “Answers if the stack is empty” • A non-boolean query: “Retrieves the number of elements of the stack”; “Retrieves the element on the top of the stack” • A modifier: “Adds a new element on top of the stack”; “Removes the element on top the stack” • A manager: “Builds a Stack with no elements on it”; “Releases the resources use by Stack”
Method	@details	Optional	After @brief	A description of the method that can span multiple lines of a paragraph. If more paragraphs are needed, then put subsequent @details tags. This description shall explain what the method does with all the needed details to understand it. Nevertheless, some methods are obvious from its brief description or they can be fully explained with pre and postconditions.
Method	@tparam	Mandatory if the method is a template with parameters	After @details	Each template parameter must have its own @tparam tag, which have to include its name and a description. The name must be the same of the template parameter. The description must be a brief explanation intended to fit into a single line, but it can span multiple lines if needed. Think of it as a tooltip.
Method	@param	Mandatory if it has parameters	After @tparam	Each parameter must have its own @param tag, which have to include its direction, its name, and a description. The value of the direction can be “in”, “out” or “in,out”. The name must be the same of the parameter. The description must be a brief explanation intended to fit into a single line, but it can span multiple lines if needed. Think of it as a tooltip.
Method	@return	Mandatory if it returns something	After last @param	A brief explanation of the value returned by the method, which is intended to fit into a single line, but it can span multiple lines if needed. Think of it as a tooltip.
Method	@pre	Optional	After @return	The precondition of the method. It can be expressed formally or informally, because it is not intended to be used by static analysis tools neither runtime checks.
Method	@post	Optional	After @pre	The postcondition of the method. It can be expressed formally or informally, because it is not intended to be used by static analysis tools neither runtime checks.
Method	@remark	Optional	After @post	A clarifying comment about the method which could have been included in other tags, like details or tparams/params, but needs to be explicitly noticed.

Scope	Tag		Place	Description
Method	@warning	Optional	After @remark	An advisory notice about class restrictions or behavior, which can have undesired or unexpected side effects. Beware that sometimes these kind of comments fit better as preconditions and postconditions.
Method	@see	Optional	Not defined	A cross reference to another method, usually the one overridden by this method. *Note: If no new information is going to be added on this method, then it is fine to put this tag alone and drop the mandatory tags.*
Method	@verbatim / @endverbatim	Optional	Inline in any tag	A text to be verbatim included, delimited by a starting @verbatim and an ending @endverbatim.

Table 8 Tags for Doxygen

12ANNEX. CONFIGURATION FILE FOR DOXYGEN

This annex lists the contents of the file *marte.doxyfile*, which defines the C++ formatting rules for C++ source files that will be used by Eclipse CDT. These rules are based on Eclipse CDT's K&R profile. [Last update on 24/02/2016]

```
# Doxyfile 1.8.3.1

# This file describes the settings to be used by the documentation system
# doxygen (www.doxygen.org) for a project.
#
# All text after a hash (#) is considered a comment and will be ignored.
# The format is:
#       TAG = value [value, ...]
# For lists items can also be appended using:
#       TAG += value [value, ...]
# Values that contain spaces should be placed between quotes (" ").

#---------------------------------------------------------------------------
# Project related configuration options
#---------------------------------------------------------------------------

# This tag specifies the encoding used for all characters in the config file
# that follow. The default is UTF-8 which is also the encoding used for all
# text before the first occurrence of this tag. Doxygen uses libiconv (or the
# iconv built into libc) for the transcoding. See
# http://www.gnu.org/software/libiconv for the list of possible encodings.

DOXYFILE_ENCODING = UTF-8

# The PROJECT_NAME tag is a single word (or sequence of words) that should
# identify the project. Note that if you do not use Doxywizard you need
# to put quotes around the project name if it contains spaces.

PROJECT_NAME = "MARTe2"

# The PROJECT_NUMBER tag can be used to enter a project or revision number.
# This could be handy for archiving the generated documentation or
# if some version control system is used.

PROJECT_NUMBER =
```

```
# Using the PROJECT_BRIEF tag one can provide an optional one line description
# for a project that appears at the top of each page and should give viewer
# a quick idea about the purpose of the project. Keep the description short.

PROJECT_BRIEF = "C++ real-time application development framework"

# With the PROJECT_LOGO tag one can specify an logo or icon that is
# included in the documentation. The maximum height of the logo should not
# exceed 55 pixels and the maximum width should not exceed 200 pixels.
# Doxygen will copy the logo to the output directory.

PROJECT_LOGO =

# The OUTPUT_DIRECTORY tag is used to specify the (relative or absolute)
# base path where the generated documentation will be put.
# If a relative path is entered, it will be relative to the location
# where doxygen was started. If left blank the current directory will be used.

OUTPUT_DIRECTORY =

# If the CREATE_SUBDIRS tag is set to YES, then doxygen will create
# 4096 sub-directories (in 2 levels) under the output directory of each output
# format and will distribute the generated files over these directories.
# Enabling this option can be useful when feeding doxygen a huge amount of
# source files, where putting all generated files in the same directory would
# otherwise cause performance problems for the file system.

CREATE_SUBDIRS = NO

# The OUTPUT_LANGUAGE tag is used to specify the language in which all
# documentation generated by doxygen is written. Doxygen will use this
# information to generate all constant output in the proper language.
# The default language is English, other supported languages are:
# Afrikaans, Arabic, Brazilian, Catalan, Chinese, Chinese-Traditional,
# Croatian, Czech, Danish, Dutch, Esperanto, Farsi, Finnish, French, German,
# Greek, Hungarian, Italian, Japanese, Japanese-en (Japanese with English
# messages), Korean, Korean-en, Lithuanian, Norwegian, Macedonian, Persian,
# Polish, Portuguese, Romanian, Russian, Serbian, Serbian-Cyrillic, Slovak,
# Slovene, Spanish, Swedish, Ukrainian, and Vietnamese.

OUTPUT_LANGUAGE = English

# If the BRIEF_MEMBER_DESC tag is set to YES (the default) Doxygen will
```

```
# include brief member descriptions after the members that are listed in
# the file and class documentation (similar to JavaDoc).
# Set to NO to disable this.

BRIEF_MEMBER_DESC = YES

# If the REPEAT_BRIEF tag is set to YES (the default) Doxygen will prepend
# the brief description of a member or function before the detailed description.
# Note: if both HIDE_UNDOC_MEMBERS and BRIEF_MEMBER_DESC are set to NO, the
# brief descriptions will be completely suppressed.

REPEAT_BRIEF = YES

# This tag implements a quasi-intelligent brief description abbreviator
# that is used to form the text in various listings. Each string
# in this list, if found as the leading text of the brief description, will be
# stripped from the text and the result after processing the whole list, is
# used as the annotated text. Otherwise, the brief description is used as-is.
# If left blank, the following values are used ("$name" is automatically
# replaced with the name of the entity): "The $name class" "The $name widget"
# "The $name file" "is" "provides" "specifies" "contains"
# "represents" "a" "an" "the"

ABBREVIATE_BRIEF =

# If the ALWAYS_DETAILED_SEC and REPEAT_BRIEF tags are both set to YES then
# Doxygen will generate a detailed section even if there is only a brief
# description.

ALWAYS_DETAILED_SEC = NO

# If the INLINE_INHERITED_MEMB tag is set to YES, doxygen will show all
# inherited members of a class in the documentation of that class as if those
# members were ordinary class members. Constructors, destructors and assignment
# operators of the base classes will not be shown.

INLINE_INHERITED_MEMB = NO

# If the FULL_PATH_NAMES tag is set to YES then Doxygen will prepend the full
# path before files name in the file list and in the header files. If set
# to NO the shortest path that makes the file name unique will be used.

FULL_PATH_NAMES = YES
```

```
# If the FULL_PATH_NAMES tag is set to YES then the STRIP_FROM_PATH tag
# can be used to strip a user-defined part of the path. Stripping is
# only done if one of the specified strings matches the left-hand part of
# the path. The tag can be used to show relative paths in the file list.
# If left blank the directory from which doxygen is run is used as the
# path to strip. Note that you specify absolute paths here, but also
# relative paths, which will be relative from the directory where doxygen is
# started.

STRIP_FROM_PATH =

# The STRIP_FROM_INC_PATH tag can be used to strip a user-defined part of
# the path mentioned in the documentation of a class, which tells
# the reader which header file to include in order to use a class.
# If left blank only the name of the header file containing the class
# definition is used. Otherwise one should specify the include paths that
# are normally passed to the compiler using the -I flag.

STRIP_FROM_INC_PATH =

# If the SHORT_NAMES tag is set to YES, doxygen will generate much shorter
# (but less readable) file names. This can be useful if your file system
# doesn't support long names like on DOS, Mac, or CD-ROM.

SHORT_NAMES = NO

# If the JAVADOC_AUTOBRIEF tag is set to YES then Doxygen
# will interpret the first line (until the first dot) of a JavaDoc-style
# comment as the brief description. If set to NO, the JavaDoc
# comments will behave just like regular Qt-style comments
# (thus requiring an explicit @brief command for a brief description.)

JAVADOC_AUTOBRIEF = YES

# If the QT_AUTOBRIEF tag is set to YES then Doxygen will
# interpret the first line (until the first dot) of a Qt-style
# comment as the brief description. If set to NO, the comments
# will behave just like regular Qt-style comments (thus requiring
# an explicit \brief command for a brief description.)

QT_AUTOBRIEF = NO
```

```
# The MULTILINE_CPP_IS_BRIEF tag can be set to YES to make Doxygen
# treat a multi-line C++ special comment block (i.e. a block of //! or ///
# comments) as a brief description. This used to be the default behaviour.
# The new default is to treat a multi-line C++ comment block as a detailed
# description. Set this tag to YES if you prefer the old behaviour instead.

MULTILINE_CPP_IS_BRIEF = NO

# If the INHERIT_DOCS tag is set to YES (the default) then an undocumented
# member inherits the documentation from any documented member that it
# re-implements.

INHERIT_DOCS = YES

# If the SEPARATE_MEMBER_PAGES tag is set to YES, then doxygen will produce
# a new page for each member. If set to NO, the documentation of a member will
# be part of the file/class/namespace that contains it.

SEPARATE_MEMBER_PAGES = NO

# The TAB_SIZE tag can be used to set the number of spaces in a tab.
# Doxygen uses this value to replace tabs by spaces in code fragments.

TAB_SIZE = 4

# This tag can be used to specify a number of aliases that acts
# as commands in the documentation. An alias has the form "name=value".
# For example adding "sideeffect=\par Side Effects:\n" will allow you to
# put the command \sideeffect (or @sideeffect) in the documentation, which
# will result in a user-defined paragraph with heading "Side Effects:".
# You can put \n's in the value part of an alias to insert newlines.

ALIASES =

# This tag can be used to specify a number of word-keyword mappings (TCL only).
# A mapping has the form "name=value". For example adding
# "class=itcl::class" will allow you to use the command class in the
# itcl::class meaning.

TCL_SUBST =

# Set the OPTIMIZE_OUTPUT_FOR_C tag to YES if your project consists of C
# sources only. Doxygen will then generate output that is more tailored for C.
```

```
# For instance, some of the names that are used will be different. The list
# of all members will be omitted, etc.

OPTIMIZE_OUTPUT_FOR_C = NO

# Set the OPTIMIZE_OUTPUT_JAVA tag to YES if your project consists of Java
# sources only. Doxygen will then generate output that is more tailored for
# Java. For instance, namespaces will be presented as packages, qualified
# scopes will look different, etc.

OPTIMIZE_OUTPUT_JAVA = NO

# Set the OPTIMIZE_FOR_FORTRAN tag to YES if your project consists of Fortran
# sources only. Doxygen will then generate output that is more tailored for
# Fortran.

OPTIMIZE_FOR_FORTRAN = NO

# Set the OPTIMIZE_OUTPUT_VHDL tag to YES if your project consists of VHDL
# sources. Doxygen will then generate output that is tailored for
# VHDL.

OPTIMIZE_OUTPUT_VHDL = NO

# Doxygen selects the parser to use depending on the extension of the files it
# parses. With this tag you can assign which parser to use for a given
# extension. Doxygen has a built-in mapping, but you can override or extend it
# using this tag. The format is ext=language, where ext is a file extension,
# and language is one of the parsers supported by doxygen: IDL, Java,
# Javascript, CSharp, C, C++, D, PHP, Objective-C, Python, Fortran, VHDL, C,
# C++. For instance to make doxygen treat .inc files as Fortran files (default
# is PHP), and .f files as C (default is Fortran), use: inc=Fortran f=C. Note
# that for custom extensions you also need to set FILE_PATTERNS otherwise the
# files are not read by doxygen.

EXTENSION_MAPPING =

# If MARKDOWN_SUPPORT is enabled (the default) then doxygen pre-processes all
# comments according to the Markdown format, which allows for more readable
# documentation. See http://daringfireball.net/projects/markdown/ for details.
# The output of markdown processing is further processed by doxygen, so you
# can mix doxygen, HTML, and XML commands with Markdown formatting.
# Disable only in case of backward compatibilities issues.
```

```
MARKDOWN_SUPPORT = YES

# When enabled doxygen tries to link words that correspond to documented classes,
# or namespaces to their corresponding documentation. Such a link can be
# prevented in individual cases by by putting a % sign in front of the word or
# globally by setting AUTOLINK_SUPPORT to NO.

AUTOLINK_SUPPORT = YES

# If you use STL classes (i.e. std::string, std::vector, etc.) but do not want
# to include (a tag file for) the STL sources as input, then you should
# set this tag to YES in order to let doxygen match functions declarations and
# definitions whose arguments contain STL classes (e.g. func(std::string); v.s.
# func(std::string) {}). This also makes the inheritance and collaboration
# diagrams that involve STL classes more complete and accurate.

BUILTIN_STL_SUPPORT = NO

# If you use Microsoft's C++/CLI language, you should set this option to YES to
# enable parsing support.

CPP_CLI_SUPPORT = NO

# Set the SIP_SUPPORT tag to YES if your project consists of sip sources only.
# Doxygen will parse them like normal C++ but will assume all classes use public
# instead of private inheritance when no explicit protection keyword is present.

SIP_SUPPORT = NO

# For Microsoft's IDL there are propget and propput attributes to indicate
# getter and setter methods for a property. Setting this option to YES (the
# default) will make doxygen replace the get and set methods by a property in
# the documentation. This will only work if the methods are indeed getting or
# setting a simple type. If this is not the case, or you want to show the
# methods anyway, you should set this option to NO.

IDL_PROPERTY_SUPPORT = YES

# If member grouping is used in the documentation and the DISTRIBUTE_GROUP_DOC
# tag is set to YES, then doxygen will reuse the documentation of the first
# member in the group (if any) for the other members of the group. By default
# all members of a group must be documented explicitly.
```

```
DISTRIBUTE_GROUP_DOC = NO

# Set the SUBGROUPING tag to YES (the default) to allow class member groups of
# the same type (for instance a group of public functions) to be put as a
# subgroup of that type (e.g. under the Public Functions section). Set it to
# NO to prevent subgrouping. Alternatively, this can be done per class using
# the \nosubgrouping command.

SUBGROUPING = YES

# When the INLINE_GROUPED_CLASSES tag is set to YES, classes, structs and
# unions are shown inside the group in which they are included (e.g. using
# @ingroup) instead of on a separate page (for HTML and Man pages) or
# section (for LaTeX and RTF).

INLINE_GROUPED_CLASSES = NO

# When the INLINE_SIMPLE_STRUCTS tag is set to YES, structs, classes, and
# unions with only public data fields will be shown inline in the documentation
# of the scope in which they are defined (i.e. file, namespace, or group
# documentation), provided this scope is documented. If set to NO (the default),
# structs, classes, and unions are shown on a separate page (for HTML and Man
# pages) or section (for LaTeX and RTF).

INLINE_SIMPLE_STRUCTS = NO

# When TYPEDEF_HIDES_STRUCT is enabled, a typedef of a struct, union, or enum
# is documented as struct, union, or enum with the name of the typedef. So
# typedef struct TypeS {} TypeT, will appear in the documentation as a struct
# with name TypeT. When disabled the typedef will appear as a member of a file,
# namespace, or class. And the struct will be named TypeS. This can typically
# be useful for C code in case the coding convention dictates that all compound
# types are typedef'ed and only the typedef is referenced, never the tag name.

TYPEDEF_HIDES_STRUCT = NO

# The SYMBOL_CACHE_SIZE determines the size of the internal cache use to
# determine which symbols to keep in memory and which to flush to disk.
# When the cache is full, less often used symbols will be written to disk.
# For small to medium size projects (<1000 input files) the default value is
# probably good enough. For larger projects a too small cache size can cause
# doxygen to be busy swapping symbols to and from disk most of the time
```

```
# causing a significant performance penalty.
# If the system has enough physical memory increasing the cache will improve the
# performance by keeping more symbols in memory. Note that the value works on
# a logarithmic scale so increasing the size by one will roughly double the
# memory usage. The cache size is given by this formula:
# 2^(16+SYMBOL_CACHE_SIZE). The valid range is 0..9, the default is 0,
# corresponding to a cache size of 2^16 = 65536 symbols.

#SYMBOL_CACHE_SIZE = 0

# Similar to the SYMBOL_CACHE_SIZE the size of the symbol lookup cache can be
# set using LOOKUP_CACHE_SIZE. This cache is used to resolve symbols given
# their name and scope. Since this can be an expensive process and often the
# same symbol appear multiple times in the code, doxygen keeps a cache of
# pre-resolved symbols. If the cache is too small doxygen will become slower.
# If the cache is too large, memory is wasted. The cache size is given by this
# formula: 2^(16+LOOKUP_CACHE_SIZE). The valid range is 0..9, the default is 0,
# corresponding to a cache size of 2^16 = 65536 symbols.

LOOKUP_CACHE_SIZE = 0

#---------------------------------------------------------------------------
# Build related configuration options
#---------------------------------------------------------------------------

# If the EXTRACT_ALL tag is set to YES doxygen will assume all entities in
# documentation are documented, even if no documentation was available.
# Private class members and static file members will be hidden unless
# the EXTRACT_PRIVATE and EXTRACT_STATIC tags are set to YES

EXTRACT_ALL = NO

# If the EXTRACT_PRIVATE tag is set to YES all private members of a class
# will be included in the documentation.

EXTRACT_PRIVATE = NO

# If the EXTRACT_PACKAGE tag is set to YES all members with package or internal
# scope will be included in the documentation.

EXTRACT_PACKAGE = NO

# If the EXTRACT_STATIC tag is set to YES all static members of a file
```

```
# will be included in the documentation.

EXTRACT_STATIC = NO

# If the EXTRACT_LOCAL_CLASSES tag is set to YES classes (and structs)
# defined locally in source files will be included in the documentation.
# If set to NO only classes defined in header files are included.

EXTRACT_LOCAL_CLASSES = YES

# This flag is only useful for Objective-C code. When set to YES local
# methods, which are defined in the implementation section but not in
# the interface are included in the documentation.
# If set to NO (the default) only methods in the interface are included.

EXTRACT_LOCAL_METHODS = NO

# If this flag is set to YES, the members of anonymous namespaces will be
# extracted and appear in the documentation as a namespace called
# 'anonymous_namespace{file}', where file will be replaced with the base
# name of the file that contains the anonymous namespace. By default
# anonymous namespaces are hidden.

EXTRACT_ANON_NSPACES = NO

# If the HIDE_UNDOC_MEMBERS tag is set to YES, Doxygen will hide all
# undocumented members of documented classes, files or namespaces.
# If set to NO (the default) these members will be included in the
# various overviews, but no documentation section is generated.
# This option has no effect if EXTRACT_ALL is enabled.

HIDE_UNDOC_MEMBERS = NO

# If the HIDE_UNDOC_CLASSES tag is set to YES, Doxygen will hide all
# undocumented classes that are normally visible in the class hierarchy.
# If set to NO (the default) these classes will be included in the various
# overviews. This option has no effect if EXTRACT_ALL is enabled.

HIDE_UNDOC_CLASSES = NO

# If the HIDE_FRIEND_COMPOUNDS tag is set to YES, Doxygen will hide all
# friend (class|struct|union) declarations.
# If set to NO (the default) these declarations will be included in the
```

```
# documentation.

HIDE_FRIEND_COMPOUNDS = NO

# If the HIDE_IN_BODY_DOCS tag is set to YES, Doxygen will hide any
# documentation blocks found inside the body of a function.
# If set to NO (the default) these blocks will be appended to the
# function's detailed documentation block.

HIDE_IN_BODY_DOCS = NO

# The INTERNAL_DOCS tag determines if documentation
# that is typed after a \internal command is included. If the tag is set
# to NO (the default) then the documentation will be excluded.
# Set it to YES to include the internal documentation.

INTERNAL_DOCS = NO

# If the CASE_SENSE_NAMES tag is set to NO then Doxygen will only generate
# file names in lower-case letters. If set to YES upper-case letters are also
# allowed. This is useful if you have classes or files whose names only differ
# in case and if your file system supports case sensitive file names. Windows
# and Mac users are advised to set this option to NO.

CASE_SENSE_NAMES = YES

# If the HIDE_SCOPE_NAMES tag is set to NO (the default) then Doxygen
# will show members with their full class and namespace scopes in the
# documentation. If set to YES the scope will be hidden.

HIDE_SCOPE_NAMES = NO

# If the SHOW_INCLUDE_FILES tag is set to YES (the default) then Doxygen
# will put a list of the files that are included by a file in the documentation
# of that file.

SHOW_INCLUDE_FILES = YES

# If the FORCE_LOCAL_INCLUDES tag is set to YES then Doxygen
# will list include files with double quotes in the documentation
# rather than with sharp brackets.

FORCE_LOCAL_INCLUDES = NO
```

```
# If the INLINE_INFO tag is set to YES (the default) then a tag [inline]
# is inserted in the documentation for inline members.

INLINE_INFO = YES

# If the SORT_MEMBER_DOCS tag is set to YES (the default) then doxygen
# will sort the (detailed) documentation of file and class members
# alphabetically by member name. If set to NO the members will appear in
# declaration order.

SORT_MEMBER_DOCS = YES

# If the SORT_BRIEF_DOCS tag is set to YES then doxygen will sort the
# brief documentation of file, namespace and class members alphabetically
# by member name. If set to NO (the default) the members will appear in
# declaration order.

SORT_BRIEF_DOCS = NO

# If the SORT_MEMBERS_CTORS_1ST tag is set to YES then doxygen
# will sort the (brief and detailed) documentation of class members so that
# constructors and destructors are listed first. If set to NO (the default)
# the constructors will appear in the respective orders defined by
# SORT_MEMBER_DOCS and SORT_BRIEF_DOCS.
# This tag will be ignored for brief docs if SORT_BRIEF_DOCS is set to NO
# and ignored for detailed docs if SORT_MEMBER_DOCS is set to NO.

SORT_MEMBERS_CTORS_1ST = NO

# If the SORT_GROUP_NAMES tag is set to YES then doxygen will sort the
# hierarchy of group names into alphabetical order. If set to NO (the default)
# the group names will appear in their defined order.

SORT_GROUP_NAMES = NO

# If the SORT_BY_SCOPE_NAME tag is set to YES, the class list will be
# sorted by fully-qualified names, including namespaces. If set to
# NO (the default), the class list will be sorted only by class name,
# not including the namespace part.
# Note: This option is not very useful if HIDE_SCOPE_NAMES is set to YES.
# Note: This option applies only to the class list, not to the
# alphabetical list.
```

```
SORT_BY_SCOPE_NAME = NO

# If the STRICT_PROTO_MATCHING option is enabled and doxygen fails to
# do proper type resolution of all parameters of a function it will reject a
# match between the prototype and the implementation of a member function even
# if there is only one candidate or it is obvious which candidate to choose
# by doing a simple string match. By disabling STRICT_PROTO_MATCHING doxygen
# will still accept a match between prototype and implementation in such cases.

STRICT_PROTO_MATCHING = NO

# The GENERATE_TODOLIST tag can be used to enable (YES) or
# disable (NO) the todo list. This list is created by putting \todo
# commands in the documentation.

GENERATE_TODOLIST = YES

# The GENERATE_TESTLIST tag can be used to enable (YES) or
# disable (NO) the test list. This list is created by putting \test
# commands in the documentation.

GENERATE_TESTLIST = YES

# The GENERATE_BUGLIST tag can be used to enable (YES) or
# disable (NO) the bug list. This list is created by putting \bug
# commands in the documentation.

GENERATE_BUGLIST = YES

# The GENERATE_DEPRECATEDLIST tag can be used to enable (YES) or
# disable (NO) the deprecated list. This list is created by putting
# \deprecated commands in the documentation.

GENERATE_DEPRECATEDLIST = YES

# The ENABLED_SECTIONS tag can be used to enable conditional
# documentation sections, marked by \if section-label ... \endif
# and \cond section-label ... \endcond blocks.

ENABLED_SECTIONS =

# The MAX_INITIALIZER_LINES tag determines the maximum number of lines
```

```
# the initial value of a variable or macro consists of for it to appear in
# the documentation. If the initializer consists of more lines than specified
# here it will be hidden. Use a value of 0 to hide initializers completely.
# The appearance of the initializer of individual variables and macros in the
# documentation can be controlled using \showinitializer or \hideinitializer
# command in the documentation regardless of this setting.

MAX_INITIALIZER_LINES = 30

# Set the SHOW_USED_FILES tag to NO to disable the list of files generated
# at the bottom of the documentation of classes and structs. If set to YES the
# list will mention the files that were used to generate the documentation.

SHOW_USED_FILES = YES

# Set the SHOW_FILES tag to NO to disable the generation of the Files page.
# This will remove the Files entry from the Quick Index and from the
# Folder Tree View (if specified). The default is YES.

SHOW_FILES = YES

# Set the SHOW_NAMESPACES tag to NO to disable the generation of the
# Namespaces page.
# This will remove the Namespaces entry from the Quick Index
# and from the Folder Tree View (if specified). The default is YES.

SHOW_NAMESPACES = YES

# The FILE_VERSION_FILTER tag can be used to specify a program or script that
# doxygen should invoke to get the current version for each file (typically from
# the version control system). Doxygen will invoke the program by executing (via
# popen()) the command <command> <input-file>, where <command> is the value of
# the FILE_VERSION_FILTER tag, and <input-file> is the name of an input file
# provided by doxygen. Whatever the program writes to standard output
# is used as the file version. See the manual for examples.

FILE_VERSION_FILTER =

# The LAYOUT_FILE tag can be used to specify a layout file which will be parsed
# by doxygen. The layout file controls the global structure of the generated
# output files in an output format independent way. To create the layout file
# that represents doxygen's defaults, run doxygen with the -l option.
# You can optionally specify a file name after the option, if omitted
```

```
# DoxygenLayout.xml will be used as the name of the layout file.

LAYOUT_FILE =

# The CITE_BIB_FILES tag can be used to specify one or more bib files
# containing the references data. This must be a list of .bib files. The
# .bib extension is automatically appended if omitted. Using this command
# requires the bibtex tool to be installed. See also
# http://en.wikipedia.org/wiki/BibTeX for more info. For LaTeX the style
# of the bibliography can be controlled using LATEX_BIB_STYLE. To use this
# feature you need bibtex and perl available in the search path. Do not use
# file names with spaces, bibtex cannot handle them.

CITE_BIB_FILES =

#---------------------------------------------------------------------------
# configuration options related to warning and progress messages
#---------------------------------------------------------------------------

# The QUIET tag can be used to turn on/off the messages that are generated
# by doxygen. Possible values are YES and NO. If left blank NO is used.

QUIET = NO

# The WARNINGS tag can be used to turn on/off the warning messages that are
# generated by doxygen. Possible values are YES and NO. If left blank
# NO is used.

WARNINGS = YES

# If WARN_IF_UNDOCUMENTED is set to YES, then doxygen will generate warnings
# for undocumented members. If EXTRACT_ALL is set to YES then this flag will
# automatically be disabled.

WARN_IF_UNDOCUMENTED = YES

# If WARN_IF_DOC_ERROR is set to YES, doxygen will generate warnings for
# potential errors in the documentation, such as not documenting some
# parameters in a documented function, or documenting parameters that
# don't exist or using markup commands wrongly.

WARN_IF_DOC_ERROR = YES
```

```
# The WARN_NO_PARAMDOC option can be enabled to get warnings for
# functions that are documented, but have no documentation for their parameters
# or return value. If set to NO (the default) doxygen will only warn about
# wrong or incomplete parameter documentation, but not about the absence of
# documentation.

WARN_NO_PARAMDOC = NO

# The WARN_FORMAT tag determines the format of the warning messages that
# doxygen can produce. The string should contain the $file, $line, and $text
# tags, which will be replaced by the file and line number from which the
# warning originated and the warning text. Optionally the format may contain
# $version, which will be replaced by the version of the file (if it could
# be obtained via FILE_VERSION_FILTER)

WARN_FORMAT = "$file:$line: $text"

# The WARN_LOGFILE tag can be used to specify a file to which warning
# and error messages should be written. If left blank the output is written
# to stderr.

WARN_LOGFILE =

#---------------------------------------------------------------------------
# configuration options related to the input files
#---------------------------------------------------------------------------

# The INPUT tag can be used to specify the files and/or directories that contain
# documented source files. You may enter file names like "myfile.cpp" or
# directories like "/usr/src/myproject". Separate the files or directories
# with spaces.

INPUT = ../../../MARTe2-dev/Source

# This tag can be used to specify the character encoding of the source files
# that doxygen parses. Internally doxygen uses the UTF-8 encoding, which is
# also the default input encoding. Doxygen uses libiconv (or the iconv built
# into libc) for the transcoding. See http://www.gnu.org/software/libiconv for
# the list of possible encodings.

INPUT_ENCODING = UTF-8

# If the value of the INPUT tag contains directories, you can use the
```

```
# FILE_PATTERNS tag to specify one or more wildcard pattern (like *.cpp
# and *.h) to filter out the source-files in the directories. If left
# blank the following patterns are tested:
# *.c *.cc *.cxx *.cpp *.c++ *.d *.java *.ii *.ixx *.ipp *.i++ *.inl *.h *.hh
# *.hxx *.hpp *.h++ *.idl *.odl *.cs *.php *.php3 *.inc *.m *.mm *.dox *.py
# *.f90 *.f *.for *.vhd *.vhdl

FILE_PATTERNS =

# The RECURSIVE tag can be used to turn specify whether or not subdirectories
# should be searched for input files as well. Possible values are YES and NO.
# If left blank NO is used.

RECURSIVE = YES

# The EXCLUDE tag can be used to specify files and/or directories that should be
# excluded from the INPUT source files. This way you can easily exclude a
# subdirectory from a directory tree whose root is specified with the INPUT tag.
# Note that relative paths are relative to the directory from which doxygen is
# run.

EXCLUDE =

# The EXCLUDE_SYMLINKS tag can be used to select whether or not files or
# directories that are symbolic links (a Unix file system feature) are excluded
# from the input.

EXCLUDE_SYMLINKS = NO

# If the value of the INPUT tag contains directories, you can use the
# EXCLUDE_PATTERNS tag to specify one or more wildcard patterns to exclude
# certain files from those directories. Note that the wildcards are matched
# against the file with absolute path, so to exclude all test directories
# for example use the pattern */test/*

EXCLUDE_PATTERNS = */Architecture/* */Environment/*

# The EXCLUDE_SYMBOLS tag can be used to specify one or more symbol names
# (namespaces, classes, functions, etc.) that should be excluded from the
# output. The symbol name can be a fully qualified name, a word, or if the
# wildcard * is used, a substring. Examples: ANamespace, AClass,
# AClass::ANamespace, ANamespace::*Test
```

```
EXCLUDE_SYMBOLS =

# The EXAMPLE_PATH tag can be used to specify one or more files or
# directories that contain example code fragments that are included (see
# the \include command).

EXAMPLE_PATH =

# If the value of the EXAMPLE_PATH tag contains directories, you can use the
# EXAMPLE_PATTERNS tag to specify one or more wildcard pattern (like *.cpp
# and *.h) to filter out the source-files in the directories. If left
# blank all files are included.

EXAMPLE_PATTERNS =

# If the EXAMPLE_RECURSIVE tag is set to YES then subdirectories will be
# searched for input files to be used with the \include or \dontinclude
# commands irrespective of the value of the RECURSIVE tag.
# Possible values are YES and NO. If left blank NO is used.

EXAMPLE_RECURSIVE = NO

# The IMAGE_PATH tag can be used to specify one or more files or
# directories that contain image that are included in the documentation (see
# the \image command).

IMAGE_PATH =

# The INPUT_FILTER tag can be used to specify a program that doxygen should
# invoke to filter for each input file. Doxygen will invoke the filter program
# by executing (via popen()) the command <filter> <input-file>, where <filter>
# is the value of the INPUT_FILTER tag, and <input-file> is the name of an
# input file. Doxygen will then use the output that the filter program writes
# to standard output.
# If FILTER_PATTERNS is specified, this tag will be
# ignored.

INPUT_FILTER =

# The FILTER_PATTERNS tag can be used to specify filters on a per file pattern
# basis.
# Doxygen will compare the file name with each pattern and apply the
# filter if there is a match.
```

```
# The filters are a list of the form:
# pattern=filter (like *.cpp=my_cpp_filter). See INPUT_FILTER for further
# info on how filters are used. If FILTER_PATTERNS is empty or if
# non of the patterns match the file name, INPUT_FILTER is applied.

FILTER_PATTERNS =

# If the FILTER_SOURCE_FILES tag is set to YES, the input filter (if set using
# INPUT_FILTER) will be used to filter the input files when producing source
# files to browse (i.e. when SOURCE_BROWSER is set to YES).

FILTER_SOURCE_FILES = NO

# The FILTER_SOURCE_PATTERNS tag can be used to specify source filters per file
# pattern. A pattern will override the setting for FILTER_PATTERN (if any)
# and it is also possible to disable source filtering for a specific pattern
# using *.ext= (so without naming a filter). This option only has effect when
# FILTER_SOURCE_FILES is enabled.

FILTER_SOURCE_PATTERNS =

# If the USE_MD_FILE_AS_MAINPAGE tag refers to the name of a markdown file that
# is part of the input, its contents will be placed on the main page (index.html).
# This can be useful if you have a project on for instance GitHub and want reuse
# the introduction page also for the doxygen output.

USE_MDFILE_AS_MAINPAGE = Mainpage.md

#---------------------------------------------------------------------------
# configuration options related to source browsing
#---------------------------------------------------------------------------

# If the SOURCE_BROWSER tag is set to YES then a list of source files will
# be generated. Documented entities will be cross-referenced with these sources.
# Note: To get rid of all source code in the generated output, make sure also
# VERBATIM_HEADERS is set to NO.

SOURCE_BROWSER = NO

# Setting the INLINE_SOURCES tag to YES will include the body
# of functions and classes directly in the documentation.

INLINE_SOURCES = NO
```

```
# Setting the STRIP_CODE_COMMENTS tag to YES (the default) will instruct
# doxygen to hide any special comment blocks from generated source code
# fragments. Normal C, C++ and Fortran comments will always remain visible.

STRIP_CODE_COMMENTS = YES

# If the REFERENCED_BY_RELATION tag is set to YES
# then for each documented function all documented
# functions referencing it will be listed.

REFERENCED_BY_RELATION = NO

# If the REFERENCES_RELATION tag is set to YES
# then for each documented function all documented entities
# called/used by that function will be listed.

REFERENCES_RELATION = NO

# If the REFERENCES_LINK_SOURCE tag is set to YES (the default)
# and SOURCE_BROWSER tag is set to YES, then the hyperlinks from
# functions in REFERENCES_RELATION and REFERENCED_BY_RELATION lists will
# link to the source code.
# Otherwise they will link to the documentation.

REFERENCES_LINK_SOURCE = YES

# If the USE_HTAGS tag is set to YES then the references to source code
# will point to the HTML generated by the htags(1) tool instead of doxygen
# built-in source browser. The htags tool is part of GNU's global source
# tagging system (see http://www.gnu.org/software/global/global.html). You
# will need version 4.8.6 or higher.

USE_HTAGS = NO

# If the VERBATIM_HEADERS tag is set to YES (the default) then Doxygen
# will generate a verbatim copy of the header file for each class for
# which an include is specified. Set to NO to disable this.

VERBATIM_HEADERS = YES

#---------------------------------------------------------------------------
# configuration options related to the alphabetical class index
```

```
#---------------------------------------------------------------------------

# If the ALPHABETICAL_INDEX tag is set to YES, an alphabetical index
# of all compounds will be generated. Enable this if the project
# contains a lot of classes, structs, unions or interfaces.

ALPHABETICAL_INDEX = YES

# If the alphabetical index is enabled (see ALPHABETICAL_INDEX) then
# the COLS_IN_ALPHA_INDEX tag can be used to specify the number of columns
# in which this list will be split (can be a number in the range [1..20])

COLS_IN_ALPHA_INDEX = 5

# In case all classes in a project start with a common prefix, all
# classes will be put under the same header in the alphabetical index.
# The IGNORE_PREFIX tag can be used to specify one or more prefixes that
# should be ignored while generating the index headers.

IGNORE_PREFIX =

#---------------------------------------------------------------------------
# configuration options related to the HTML output
#---------------------------------------------------------------------------

# If the GENERATE_HTML tag is set to YES (the default) Doxygen will
# generate HTML output.

GENERATE_HTML = YES

# The HTML_OUTPUT tag is used to specify where the HTML docs will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `html' will be used as the default path.

HTML_OUTPUT = html

# The HTML_FILE_EXTENSION tag can be used to specify the file extension for
# each generated HTML page (for example: .htm,.php,.asp). If it is left blank
# doxygen will generate files with .html extension.

HTML_FILE_EXTENSION = .html

# The HTML_HEADER tag can be used to specify a personal HTML header for
```

```
# each generated HTML page. If it is left blank doxygen will generate a
# standard header. Note that when using a custom header you are responsible
#  for the proper inclusion of any scripts and style sheets that doxygen
# needs, which is dependent on the configuration options used.
# It is advised to generate a default header using "doxygen -w html
# header.html footer.html stylesheet.css YourConfigFile" and then modify
# that header. Note that the header is subject to change so you typically
# have to redo this when upgrading to a newer version of doxygen or when
# changing the value of configuration settings such as GENERATE_TREEVIEW!

HTML_HEADER =

# The HTML_FOOTER tag can be used to specify a personal HTML footer for
# each generated HTML page. If it is left blank doxygen will generate a
# standard footer.

HTML_FOOTER =

# The HTML_STYLESHEET tag can be used to specify a user-defined cascading
# style sheet that is used by each HTML page. It can be used to
# fine-tune the look of the HTML output. If left blank doxygen will
# generate a default style sheet. Note that it is recommended to use
# HTML_EXTRA_STYLESHEET instead of this one, as it is more robust and this
# tag will in the future become obsolete.

HTML_STYLESHEET =

# The HTML_EXTRA_STYLESHEET tag can be used to specify an additional
# user-defined cascading style sheet that is included after the standard
# style sheets created by doxygen. Using this option one can overrule
# certain style aspects. This is preferred over using HTML_STYLESHEET
# since it does not replace the standard style sheet and is therefor more
# robust against future updates. Doxygen will copy the style sheet file to
# the output directory.

HTML_EXTRA_STYLESHEET =

# The HTML_EXTRA_FILES tag can be used to specify one or more extra images or
# other source files which should be copied to the HTML output directory. Note
# that these files will be copied to the base HTML output directory. Use the
# $relpath$ marker in the HTML_HEADER and/or HTML_FOOTER files to load these
# files. In the HTML_STYLESHEET file, use the file name only. Also note that
# the files will be copied as-is; there are no commands or markers available.
```

```
HTML_EXTRA_FILES =

# The HTML_COLORSTYLE_HUE tag controls the color of the HTML output.
# Doxygen will adjust the colors in the style sheet and background images
# according to this color. Hue is specified as an angle on a colorwheel,
# see http://en.wikipedia.org/wiki/Hue for more information.
# For instance the value 0 represents red, 60 is yellow, 120 is green,
# 180 is cyan, 240 is blue, 300 purple, and 360 is red again.
# The allowed range is 0 to 359.

HTML_COLORSTYLE_HUE = 220

# The HTML_COLORSTYLE_SAT tag controls the purity (or saturation) of
# the colors in the HTML output. For a value of 0 the output will use
# grayscales only. A value of 255 will produce the most vivid colors.

HTML_COLORSTYLE_SAT = 100

# The HTML_COLORSTYLE_GAMMA tag controls the gamma correction applied to
# the luminance component of the colors in the HTML output. Values below
# 100 gradually make the output lighter, whereas values above 100 make
# the output darker. The value divided by 100 is the actual gamma applied,
# so 80 represents a gamma of 0.8, The value 220 represents a gamma of 2.2,
# and 100 does not change the gamma.

HTML_COLORSTYLE_GAMMA = 80

# If the HTML_TIMESTAMP tag is set to YES then the footer of each generated HTML
# page will contain the date and time when the page was generated. Setting
# this to NO can help when comparing the output of multiple runs.

HTML_TIMESTAMP = NO

# If the HTML_DYNAMIC_SECTIONS tag is set to YES then the generated HTML
# documentation will contain sections that can be hidden and shown after the
# page has loaded.

HTML_DYNAMIC_SECTIONS = NO

# With HTML_INDEX_NUM_ENTRIES one can control the preferred number of
# entries shown in the various tree structured indices initially; the user
# can expand and collapse entries dynamically later on. Doxygen will expand
```

```
# the tree to such a level that at most the specified number of entries are
# visible (unless a fully collapsed tree already exceeds this amount).
# So setting the number of entries 1 will produce a full collapsed tree by
# default. 0 is a special value representing an infinite number of entries
# and will result in a full expanded tree by default.

HTML_INDEX_NUM_ENTRIES = 100

# If the GENERATE_DOCSET tag is set to YES, additional index files
# will be generated that can be used as input for Apple's Xcode 3
# integrated development environment, introduced with OSX 10.5 (Leopard).
# To create a documentation set, doxygen will generate a Makefile in the
# HTML output directory. Running make will produce the docset in that
# directory and running "make install" will install the docset in
# ~/Library/Developer/Shared/Documentation/DocSets so that Xcode will find
# it at startup.
# See http://developer.apple.com/tools/creatingdocsetswithdoxygen.html
# for more information.

GENERATE_DOCSET = NO

# When GENERATE_DOCSET tag is set to YES, this tag determines the name of the
# feed. A documentation feed provides an umbrella under which multiple
# documentation sets from a single provider (such as a company or product suite)
# can be grouped.

DOCSET_FEEDNAME = "Doxygen generated docs"

# When GENERATE_DOCSET tag is set to YES, this tag specifies a string that
# should uniquely identify the documentation set bundle. This should be a
# reverse domain-name style string, e.g. com.mycompany.MyDocSet. Doxygen
# will append .docset to the name.

DOCSET_BUNDLE_ID = org.doxygen.Project

# When GENERATE_PUBLISHER_ID tag specifies a string that should uniquely
# identify the documentation publisher. This should be a reverse domain-name
# style string, e.g. com.mycompany.MyDocSet.documentation.

DOCSET_PUBLISHER_ID = org.doxygen.Publisher

# The GENERATE_PUBLISHER_NAME tag identifies the documentation publisher.
```

```
DOCSET_PUBLISHER_NAME = Publisher

# If the GENERATE_HTMLHELP tag is set to YES, additional index files
# will be generated that can be used as input for tools like the
# Microsoft HTML help workshop to generate a compiled HTML help file (.chm)
# of the generated HTML documentation.

GENERATE_HTMLHELP = NO

# If the GENERATE_HTMLHELP tag is set to YES, the CHM_FILE tag can
# be used to specify the file name of the resulting .chm file. You
# can add a path in front of the file if the result should not be
# written to the html output directory.

CHM_FILE =

# If the GENERATE_HTMLHELP tag is set to YES, the HHC_LOCATION tag can
# be used to specify the location (absolute path including file name) of
# the HTML help compiler (hhc.exe). If non-empty doxygen will try to run
# the HTML help compiler on the generated index.hhp.

HHC_LOCATION =

# If the GENERATE_HTMLHELP tag is set to YES, the GENERATE_CHI flag
# controls if a separate .chi index file is generated (YES) or that
# it should be included in the master .chm file (NO).

GENERATE_CHI = NO

# If the GENERATE_HTMLHELP tag is set to YES, the CHM_INDEX_ENCODING
# is used to encode HtmlHelp index (hhk), content (hhc) and project file
# content.

CHM_INDEX_ENCODING =

# If the GENERATE_HTMLHELP tag is set to YES, the BINARY_TOC flag
# controls whether a binary table of contents is generated (YES) or a
# normal table of contents (NO) in the .chm file.

BINARY_TOC = NO

# The TOC_EXPAND flag can be set to YES to add extra items for group members
# to the contents of the HTML help documentation and to the tree view.
```

```
TOC_EXPAND = NO

# If the GENERATE_QHP tag is set to YES and both QHP_NAMESPACE and
# QHP_VIRTUAL_FOLDER are set, an additional index file will be generated
# that can be used as input for Qt's qhelpgenerator to generate a
# Qt Compressed Help (.qch) of the generated HTML documentation.

GENERATE_QHP = NO

# If the QHG_LOCATION tag is specified, the QCH_FILE tag can
# be used to specify the file name of the resulting .qch file.
# The path specified is relative to the HTML output folder.

QCH_FILE =

# The QHP_NAMESPACE tag specifies the namespace to use when generating
# Qt Help Project output. For more information please see
# http://doc.trolltech.com/qthelpproject.html#namespace

QHP_NAMESPACE = org.doxygen.Project

# The QHP_VIRTUAL_FOLDER tag specifies the namespace to use when generating
# Qt Help Project output. For more information please see
# http://doc.trolltech.com/qthelpproject.html#virtual-folders

QHP_VIRTUAL_FOLDER = doc

# If QHP_CUST_FILTER_NAME is set, it specifies the name of a custom filter to
# add. For more information please see
# http://doc.trolltech.com/qthelpproject.html#custom-filters

QHP_CUST_FILTER_NAME =

# The QHP_CUST_FILT_ATTRS tag specifies the list of the attributes of the
# custom filter to add. For more information please see
# <a href="http://doc.trolltech.com/qthelpproject.html#custom-filters">
# Qt Help Project / Custom Filters</a>.

QHP_CUST_FILTER_ATTRS =

# The QHP_SECT_FILTER_ATTRS tag specifies the list of the attributes this
# project's
```

```
# filter section matches.
# <a href="http://doc.trolltech.com/qthelpproject.html#filter-attributes">
# Qt Help Project / Filter Attributes</a>.

QHP_SECT_FILTER_ATTRS =

# If the GENERATE_QHP tag is set to YES, the QHG_LOCATION tag can
# be used to specify the location of Qt's qhelpgenerator.
# If non-empty doxygen will try to run qhelpgenerator on the generated
# .qhp file.

QHG_LOCATION =

# If the GENERATE_ECLIPSEHELP tag is set to YES, additional index files
#  will be generated, which together with the HTML files, form an Eclipse help
# plugin. To install this plugin and make it available under the help contents
# menu in Eclipse, the contents of the directory containing the HTML and XML
# files needs to be copied into the plugins directory of eclipse. The name of
# the directory within the plugins directory should be the same as
# the ECLIPSE_DOC_ID value. After copying Eclipse needs to be restarted before
# the help appears.

GENERATE_ECLIPSEHELP = NO

# A unique identifier for the eclipse help plugin. When installing the plugin
# the directory name containing the HTML and XML files should also have
# this name.

ECLIPSE_DOC_ID = org.doxygen.Project

# The DISABLE_INDEX tag can be used to turn on/off the condensed index (tabs)
# at top of each HTML page. The value NO (the default) enables the index and
# the value YES disables it. Since the tabs have the same information as the
# navigation tree you can set this option to NO if you already set
# GENERATE_TREEVIEW to YES.

DISABLE_INDEX = NO

# The GENERATE_TREEVIEW tag is used to specify whether a tree-like index
# structure should be generated to display hierarchical information.
# If the tag value is set to YES, a side panel will be generated
# containing a tree-like index structure (just like the one that
# is generated for HTML Help). For this to work a browser that supports
```

```
# JavaScript, DHTML, CSS and frames is required (i.e. any modern browser).
# Windows users are probably better off using the HTML help feature.
# Since the tree basically has the same information as the tab index you
# could consider to set DISABLE_INDEX to NO when enabling this option.

GENERATE_TREEVIEW = NO

# The ENUM_VALUES_PER_LINE tag can be used to set the number of enum values
# (range [0,1..20]) that doxygen will group on one line in the generated HTML
# documentation. Note that a value of 0 will completely suppress the enum
# values from appearing in the overview section.

ENUM_VALUES_PER_LINE = 4

# If the treeview is enabled (see GENERATE_TREEVIEW) then this tag can be
# used to set the initial width (in pixels) of the frame in which the tree
# is shown.

TREEVIEW_WIDTH = 250

# When the EXT_LINKS_IN_WINDOW option is set to YES doxygen will open
# links to external symbols imported via tag files in a separate window.

EXT_LINKS_IN_WINDOW = NO

# Use this tag to change the font size of Latex formulas included
# as images in the HTML documentation. The default is 10. Note that
# when you change the font size after a successful doxygen run you need
# to manually remove any form_*.png images from the HTML output directory
# to force them to be regenerated.

FORMULA_FONTSIZE = 10

# Use the FORMULA_TRANPARENT tag to determine whether or not the images
# generated for formulas are transparent PNGs. Transparent PNGs are
# not supported properly for IE 6.0, but are supported on all modern browsers.
# Note that when changing this option you need to delete any form_*.png files
# in the HTML output before the changes have effect.

FORMULA_TRANSPARENT = YES

# Enable the USE_MATHJAX option to render LaTeX formulas using MathJax
# (see http://www.mathjax.org) which uses client side Javascript for the
```

```
# rendering instead of using prerendered bitmaps. Use this if you do not
# have LaTeX installed or if you want to formulas look prettier in the HTML
# output. When enabled you may also need to install MathJax separately and
# configure the path to it using the MATHJAX_RELPATH option.

USE_MATHJAX = YES

# When MathJax is enabled you can set the default output format to be used for
# thA MathJax output. Supported types are HTML-CSS, NativeMML (i.e. MathML) and
# SVG. The default value is HTML-CSS, which is slower, but has the best
# compatibility.

MATHJAX_FORMAT = HTML-CSS

# When MathJax is enabled you need to specify the location relative to the
# HTML output directory using the MATHJAX_RELPATH option. The destination
# directory should contain the MathJax.js script. For instance, if the mathjax
# directory is located at the same level as the HTML output directory, then
# MATHJAX_RELPATH should be ../mathjax. The default value points to
# the MathJax Content Delivery Network so you can quickly see the result without
# installing MathJax.
# However, it is strongly recommended to install a local
# copy of MathJax from http://www.mathjax.org before deployment.

MATHJAX_RELPATH = https://cdn.mathjax.org/mathjax/latest

# The MATHJAX_EXTENSIONS tag can be used to specify one or MathJax extension
# names that should be enabled during MathJax rendering.

MATHJAX_EXTENSIONS =

# When the SEARCHENGINE tag is enabled doxygen will generate a search box
# for the HTML output. The underlying search engine uses javascript
# and DHTML and should work on any modern browser. Note that when using
# HTML help (GENERATE_HTMLHELP), Qt help (GENERATE_QHP), or docsets
# (GENERATE_DOCSET) there is already a search function so this one should
# typically be disabled. For large projects the javascript based search engine
# can be slow, then enabling SERVER_BASED_SEARCH may provide a better solution.

SEARCHENGINE = YES

# When the SERVER_BASED_SEARCH tag is enabled the search engine will be
# implemented using a web server instead of a web client using Javascript.
```

```
# There are two flavours of web server based search depending on the
# EXTERNAL_SEARCH setting. When disabled, doxygen will generate a PHP script for
# searching and an index file used by the script. When EXTERNAL_SEARCH is
# enabled the indexing and searching needs to be provided by external tools.
# See the manual for details.

SERVER_BASED_SEARCH = NO

# When EXTERNAL_SEARCH is enabled doxygen will no longer generate the PHP
# script for searching. Instead the search results are written to an XML file
# which needs to be processed by an external indexer. Doxygen will invoke an
# external search engine pointed to by the SEARCHENGINE_URL option to obtain
# the search results. Doxygen ships with an example indexer (doxyindexer) and
# search engine (doxysearch.cgi) which are based on the open source search engine
# library Xapian. See the manual for configuration details.

EXTERNAL_SEARCH = NO

# The SEARCHENGINE_URL should point to a search engine hosted by a web server
# which will returned the search results when EXTERNAL_SEARCH is enabled.
# Doxygen ships with an example search engine (doxysearch) which is based on
# the open source search engine library Xapian. See the manual for configuration
# details.

SEARCHENGINE_URL =

# When SERVER_BASED_SEARCH and EXTERNAL_SEARCH are both enabled the unindexed
# search data is written to a file for indexing by an external tool. With the
# SEARCHDATA_FILE tag the name of this file can be specified.

SEARCHDATA_FILE = searchdata.xml

# When SERVER_BASED_SEARCH AND EXTERNAL_SEARCH are both enabled the
# EXTERNAL_SEARCH_ID tag can be used as an identifier for the project. This is
# useful in combination with EXTRA_SEARCH_MAPPINGS to search through multiple
# projects and redirect the results back to the right project.

EXTERNAL_SEARCH_ID =

# The EXTRA_SEARCH_MAPPINGS tag can be used to enable searching through doxygen
# projects other than the one defined by this configuration file, but that are
# all added to the same external search index. Each project needs to have a
# unique id set via EXTERNAL_SEARCH_ID. The search mapping then maps the id
```

```
# of to a relative location where the documentation can be found.
# The format is: EXTRA_SEARCH_MAPPINGS = id1=loc1 id2=loc2 ...

EXTRA_SEARCH_MAPPINGS =

#---------------------------------------------------------------------------
# configuration options related to the LaTeX output
#---------------------------------------------------------------------------

# If the GENERATE_LATEX tag is set to YES (the default) Doxygen will
# generate Latex output.

GENERATE_LATEX = NO

# The LATEX_OUTPUT tag is used to specify where the LaTeX docs will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `latex' will be used as the default path.

LATEX_OUTPUT = latex

# The LATEX_CMD_NAME tag can be used to specify the LaTeX command name to be
# invoked. If left blank `latex' will be used as the default command name.
# Note that when enabling USE_PDFLATEX this option is only used for
# generating bitmaps for formulas in the HTML output, but not in the
# Makefile that is written to the output directory.

LATEX_CMD_NAME = latex

# The MAKEINDEX_CMD_NAME tag can be used to specify the command name to
# generate index for LaTeX. If left blank `makeindex' will be used as the
# default command name.

MAKEINDEX_CMD_NAME = makeindex

# If the COMPACT_LATEX tag is set to YES Doxygen generates more compact
# LaTeX documents. This may be useful for small projects and may help to
# save some trees in general.

COMPACT_LATEX = NO

# The PAPER_TYPE tag can be used to set the paper type that is used
# by the printer. Possible values are: a4, letter, legal and
# executive. If left blank a4wide will be used.
```

```
PAPER_TYPE = a4

# The EXTRA_PACKAGES tag can be to specify one or more names of LaTeX
# packages that should be included in the LaTeX output.

EXTRA_PACKAGES =

# The LATEX_HEADER tag can be used to specify a personal LaTeX header for
# the generated latex document. The header should contain everything until
# the first chapter. If it is left blank doxygen will generate a
# standard header. Notice: only use this tag if you know what you are doing!

LATEX_HEADER =

# The LATEX_FOOTER tag can be used to specify a personal LaTeX footer for
# the generated latex document. The footer should contain everything after
# the last chapter. If it is left blank doxygen will generate a
# standard footer. Notice: only use this tag if you know what you are doing!

LATEX_FOOTER =

# If the PDF_HYPERLINKS tag is set to YES, the LaTeX that is generated
# is prepared for conversion to pdf (using ps2pdf). The pdf file will
# contain links (just like the HTML output) instead of page references
# This makes the output suitable for online browsing using a pdf viewer.

PDF_HYPERLINKS = YES

# If the USE_PDFLATEX tag is set to YES, pdflatex will be used instead of
# plain latex in the generated Makefile. Set this option to YES to get a
# higher quality PDF documentation.

USE_PDFLATEX = YES

# If the LATEX_BATCHMODE tag is set to YES, doxygen will add the \\batchmode.
# command to the generated LaTeX files. This will instruct LaTeX to keep
# running if errors occur, instead of asking the user for help.
# This option is also used when generating formulas in HTML.

LATEX_BATCHMODE = NO

# If LATEX_HIDE_INDICES is set to YES then doxygen will not
```

```
# include the index chapters (such as File Index, Compound Index, etc.)
# in the output.

LATEX_HIDE_INDICES = NO

# If LATEX_SOURCE_CODE is set to YES then doxygen will include
# source code with syntax highlighting in the LaTeX output.
# Note that which sources are shown also depends on other settings
# such as SOURCE_BROWSER.

LATEX_SOURCE_CODE = NO

# The LATEX_BIB_STYLE tag can be used to specify the style to use for the
# bibliography, e.g. plainnat, or ieeetr. The default style is "plain". See
# http://en.wikipedia.org/wiki/BibTeX for more info.

LATEX_BIB_STYLE = plain

#---------------------------------------------------------------------------
# configuration options related to the RTF output
#---------------------------------------------------------------------------

# If the GENERATE_RTF tag is set to YES Doxygen will generate RTF output
# The RTF output is optimized for Word 97 and may not look very pretty with
# other RTF readers or editors.

GENERATE_RTF = NO

# The RTF_OUTPUT tag is used to specify where the RTF docs will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `rtf' will be used as the default path.

RTF_OUTPUT = rtf

# If the COMPACT_RTF tag is set to YES Doxygen generates more compact
# RTF documents. This may be useful for small projects and may help to
# save some trees in general.

COMPACT_RTF = NO

# If the RTF_HYPERLINKS tag is set to YES, the RTF that is generated
# will contain hyperlink fields. The RTF file will
# contain links (just like the HTML output) instead of page references.
```

```
# This makes the output suitable for online browsing using WORD or other
# programs which support those fields.
# Note: wordpad (write) and others do not support links.

RTF_HYPERLINKS = NO

# Load style sheet definitions from file. Syntax is similar to doxygen's
# config file, i.e. a series of assignments. You only have to provide
# replacements, missing definitions are set to their default value.

RTF_STYLESHEET_FILE =

# Set optional variables used in the generation of an rtf document.
# Syntax is similar to doxygen's config file.

RTF_EXTENSIONS_FILE =

#---------------------------------------------------------------------------
# configuration options related to the man page output
#---------------------------------------------------------------------------

# If the GENERATE_MAN tag is set to YES (the default) Doxygen will
# generate man pages

GENERATE_MAN = NO

# The MAN_OUTPUT tag is used to specify where the man pages will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `man' will be used as the default path.

MAN_OUTPUT = man

# The MAN_EXTENSION tag determines the extension that is added to
# the generated man pages (default is the subroutine's section .3)

MAN_EXTENSION = .3

# If the MAN_LINKS tag is set to YES and Doxygen generates man output,
# then it will generate one additional man file for each entity
# documented in the real man page(s). These additional files
# only source the real man page, but without them the man command
# would be unable to find the correct page. The default is NO.
```

```
MAN_LINKS = NO

#---------------------------------------------------------------------------
# configuration options related to the XML output
#---------------------------------------------------------------------------

# If the GENERATE_XML tag is set to YES Doxygen will
# generate an XML file that captures the structure of
# the code including all documentation.

GENERATE_XML = NO

# The XML_OUTPUT tag is used to specify where the XML pages will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `xml' will be used as the default path.

XML_OUTPUT = xml

# The XML_SCHEMA tag can be used to specify an XML schema,
# which can be used by a validating XML parser to check the
# syntax of the XML files.

#XML_SCHEMA =

# The XML_DTD tag can be used to specify an XML DTD,
# which can be used by a validating XML parser to check the
# syntax of the XML files.

#XML_DTD =

# If the XML_PROGRAMLISTING tag is set to YES Doxygen will
# dump the program listings (including syntax highlighting
# and cross-referencing information) to the XML output. Note that
# enabling this will significantly increase the size of the XML output.

XML_PROGRAMLISTING = YES

#---------------------------------------------------------------------------
# configuration options for the AutoGen Definitions output
#---------------------------------------------------------------------------

# If the GENERATE_AUTOGEN_DEF tag is set to YES Doxygen will
# generate an AutoGen Definitions (see autogen.sf.net) file
```

```
# that captures the structure of the code including all
# documentation. Note that this feature is still experimental
# and incomplete at the moment.

GENERATE_AUTOGEN_DEF = NO

#---------------------------------------------------------------------------
# configuration options related to the Perl module output
#---------------------------------------------------------------------------

# If the GENERATE_PERLMOD tag is set to YES Doxygen will
# generate a Perl module file that captures the structure of
# the code including all documentation. Note that this
# feature is still experimental and incomplete at the
# moment.

GENERATE_PERLMOD = NO

# If the PERLMOD_LATEX tag is set to YES Doxygen will generate
# the necessary Makefile rules, Perl scripts and LaTeX code to be able
# to generate PDF and DVI output from the Perl module output.

PERLMOD_LATEX = NO

# If the PERLMOD_PRETTY tag is set to YES the Perl module output will be
# nicely formatted so it can be parsed by a human reader.
# This is useful
# if you want to understand what is going on.
# On the other hand, if this
# tag is set to NO the size of the Perl module output will be much smaller
# and Perl will parse it just the same.

PERLMOD_PRETTY = YES

# The names of the make variables in the generated doxyrules.make file
# are prefixed with the string contained in PERLMOD_MAKEVAR_PREFIX.
# This is useful so different doxyrules.make files included by the same
# Makefile don't overwrite each other's variables.

PERLMOD_MAKEVAR_PREFIX =

#---------------------------------------------------------------------------
# Configuration options related to the preprocessor
```

```
#---------------------------------------------------------------------------

# If the ENABLE_PREPROCESSING tag is set to YES (the default) Doxygen will
# evaluate all C-preprocessor directives found in the sources and include
# files.

ENABLE_PREPROCESSING = YES

# If the MACRO_EXPANSION tag is set to YES Doxygen will expand all macro
# names in the source code. If set to NO (the default) only conditional
# compilation will be performed. Macro expansion can be done in a controlled
# way by setting EXPAND_ONLY_PREDEF to YES.

MACRO_EXPANSION = NO

# If the EXPAND_ONLY_PREDEF and MACRO_EXPANSION tags are both set to YES
# then the macro expansion is limited to the macros specified with the
# PREDEFINED and EXPAND_AS_DEFINED tags.

EXPAND_ONLY_PREDEF = NO

# If the SEARCH_INCLUDES tag is set to YES (the default) the includes files
# pointed to by INCLUDE_PATH will be searched when a #include is found.

SEARCH_INCLUDES = YES

# The INCLUDE_PATH tag can be used to specify one or more directories that
# contain include files that are not input files but should be processed by
# the preprocessor.

INCLUDE_PATH =

# You can use the INCLUDE_FILE_PATTERNS tag to specify one or more wildcard
# patterns (like *.h and *.hpp) to filter out the header-files in the
# directories. If left blank, the patterns specified with FILE_PATTERNS will
# be used.

INCLUDE_FILE_PATTERNS =

# The PREDEFINED tag can be used to specify one or more macro names that
# are defined before the preprocessor is started (similar to the -D option of
# gcc). The argument of the tag is a list of macros of the form: name
# or name=definition (no spaces). If the definition and the = are
```

```
# omitted =1 is assumed. To prevent a macro definition from being
# undefined via #undef or recursively expanded use the := operator
# instead of the = operator.

PREDEFINED =

# If the MACRO_EXPANSION and EXPAND_ONLY_PREDEF tags are set to YES then
# this tag can be used to specify a list of macro names that should be expanded.
# The macro definition that is found in the sources will be used.
# Use the PREDEFINED tag if you want to use a different macro definition that
# overrules the definition found in the source code.

EXPAND_AS_DEFINED =

# If the SKIP_FUNCTION_MACROS tag is set to YES (the default) then
# doxygen's preprocessor will remove all references to function-like macros
# that are alone on a line, have an all uppercase name, and do not end with a
# semicolon, because these will confuse the parser if not removed.

SKIP_FUNCTION_MACROS = YES

#---------------------------------------------------------------------------
# Configuration::additions related to external references
#---------------------------------------------------------------------------

# The TAGFILES option can be used to specify one or more tagfiles. For each
# tag file the location of the external documentation should be added. The
# format of a tag file without this location is as follows:
#
# TAGFILES = file1 file2 ...
# Adding location for the tag files is done as follows:
#
# TAGFILES = file1=loc1 "file2 = loc2" ...
# where "loc1" and "loc2" can be relative or absolute paths
# or URLs. Note that each tag file must have a unique name (where the name does
# NOT include the path). If a tag file is not located in the directory in which
# doxygen is run, you must also specify the path to the tagfile here.

TAGFILES =

# When a file name is specified after GENERATE_TAGFILE, doxygen will create
# a tag file that is based on the input files it reads.
```

```
GENERATE_TAGFILE =

# If the ALLEXTERNALS tag is set to YES all external classes will be listed
# in the class index. If set to NO only the inherited external classes
# will be listed.

ALLEXTERNALS = NO

# If the EXTERNAL_GROUPS tag is set to YES all external groups will be listed
# in the modules index. If set to NO, only the current project's groups will
# be listed.

EXTERNAL_GROUPS = YES

# The PERL_PATH should be the absolute path and name of the perl script
# interpreter (i.e. the result of `which perl').

PERL_PATH = /usr/bin/perl

#---------------------------------------------------------------------------
# Configuration options related to the dot tool
#---------------------------------------------------------------------------

# If the CLASS_DIAGRAMS tag is set to YES (the default) Doxygen will
# generate a inheritance diagram (in HTML, RTF and LaTeX) for classes with base
# or super classes. Setting the tag to NO turns the diagrams off. Note that
# this option also works with HAVE_DOT disabled, but it is recommended to
# install and use dot, since it yields more powerful graphs.

CLASS_DIAGRAMS = YES

# You can define message sequence charts within doxygen comments using the \msc
# command. Doxygen will then run the mscgen tool (see
# http://www.mcternan.me.uk/mscgen/) to produce the chart and insert it in the
# documentation. The MSCGEN_PATH tag allows you to specify the directory where
# the mscgen tool resides. If left empty the tool is assumed to be found in the
# default search path.

MSCGEN_PATH =

# If set to YES, the inheritance and collaboration graphs will hide
# inheritance and usage relations if the target is undocumented
# or is not a class.
```

```
HIDE_UNDOC_RELATIONS = YES

# If you set the HAVE_DOT tag to YES then doxygen will assume the dot tool is
# available from the path. This tool is part of Graphviz, a graph visualization
# toolkit from AT&T and Lucent Bell Labs. The other options in this section
# have no effect if this option is set to NO (the default)

HAVE_DOT = NO

# The DOT_NUM_THREADS specifies the number of dot invocations doxygen is
# allowed to run in parallel. When set to 0 (the default) doxygen will
# base this on the number of processors available in the system. You can set it
# explicitly to a value larger than 0 to get control over the balance
# between CPU load and processing speed.

DOT_NUM_THREADS = 0

# By default doxygen will use the Helvetica font for all dot files that
# doxygen generates. When you want a differently looking font you can specify
# the font name using DOT_FONTNAME. You need to make sure dot is able to find
# the font, which can be done by putting it in a standard location or by setting
# the DOTFONTPATH environment variable or by setting DOT_FONTPATH to the
# directory containing the font.

DOT_FONTNAME = Helvetica

# The DOT_FONTSIZE tag can be used to set the size of the font of dot graphs.
# The default size is 10pt.

DOT_FONTSIZE = 10

# By default doxygen will tell dot to use the Helvetica font.
# If you specify a different font using DOT_FONTNAME you can use DOT_FONTPATH to
# set the path where dot can find it.

DOT_FONTPATH =

# If the CLASS_GRAPH and HAVE_DOT tags are set to YES then doxygen
# will generate a graph for each documented class showing the direct and
# indirect inheritance relations. Setting this tag to YES will force the
# CLASS_DIAGRAMS tag to NO.
```

```
CLASS_GRAPH = YES

# If the COLLABORATION_GRAPH and HAVE_DOT tags are set to YES then doxygen
# will generate a graph for each documented class showing the direct and
# indirect implementation dependencies (inheritance, containment, and
# class references variables) of the class with other documented classes.

COLLABORATION_GRAPH = YES

# If the GROUP_GRAPHS and HAVE_DOT tags are set to YES then doxygen
# will generate a graph for groups, showing the direct groups dependencies

GROUP_GRAPHS = YES

# If the UML_LOOK tag is set to YES doxygen will generate inheritance and
# collaboration diagrams in a style similar to the OMG's Unified Modeling
# Language.

UML_LOOK = NO

# If the UML_LOOK tag is enabled, the fields and methods are shown inside
# the class node. If there are many fields or methods and many nodes the
# graph may become too big to be useful. The UML_LIMIT_NUM_FIELDS
# threshold limits the number of items for each type to make the size more
# managable. Set this to 0 for no limit. Note that the threshold may be
# exceeded by 50% before the limit is enforced.

UML_LIMIT_NUM_FIELDS = 10

# If set to YES, the inheritance and collaboration graphs will show the
# relations between templates and their instances.

TEMPLATE_RELATIONS = NO

# If the ENABLE_PREPROCESSING, SEARCH_INCLUDES, INCLUDE_GRAPH, and HAVE_DOT
# tags are set to YES then doxygen will generate a graph for each documented
# file showing the direct and indirect include dependencies of the file with
# other documented files.

INCLUDE_GRAPH = YES

# If the ENABLE_PREPROCESSING, SEARCH_INCLUDES, INCLUDED_BY_GRAPH, and
# HAVE_DOT tags are set to YES then doxygen will generate a graph for each
```

```
# documented header file showing the documented files that directly or
# indirectly include this file.

INCLUDED_BY_GRAPH = YES

# If the CALL_GRAPH and HAVE_DOT options are set to YES then
# doxygen will generate a call dependency graph for every global function
# or class method. Note that enabling this option will significantly increase
# the time of a run. So in most cases it will be better to enable call graphs
# for selected functions only using the \callgraph command.

CALL_GRAPH = NO

# If the CALLER_GRAPH and HAVE_DOT tags are set to YES then
# doxygen will generate a caller dependency graph for every global function
# or class method. Note that enabling this option will significantly increase
# the time of a run. So in most cases it will be better to enable caller
# graphs for selected functions only using the \callergraph command.

CALLER_GRAPH = NO

# If the GRAPHICAL_HIERARCHY and HAVE_DOT tags are set to YES then doxygen
# will generate a graphical hierarchy of all classes instead of a textual one.

GRAPHICAL_HIERARCHY = YES

# If the DIRECTORY_GRAPH and HAVE_DOT tags are set to YES
# then doxygen will show the dependencies a directory has on other directories
# in a graphical way. The dependency relations are determined by the #include
# relations between the files in the directories.

DIRECTORY_GRAPH = YES

# The DOT_IMAGE_FORMAT tag can be used to set the image format of the images
# generated by dot. Possible values are svg, png, jpg, or gif.
# If left blank png will be used. If you choose svg you need to set
# HTML_FILE_EXTENSION to xhtml in order to make the SVG files
# visible in IE 9+ (other browsers do not have this requirement).

DOT_IMAGE_FORMAT = png

# If DOT_IMAGE_FORMAT is set to svg, then this option can be set to YES to
# enable generation of interactive SVG images that allow zooming and panning.
```

```
# Note that this requires a modern browser other than Internet Explorer.
# Tested and working are Firefox, Chrome, Safari, and Opera. For IE 9+ you
# need to set HTML_FILE_EXTENSION to xhtml in order to make the SVG files
# visible. Older versions of IE do not have SVG support.

INTERACTIVE_SVG = NO

# The tag DOT_PATH can be used to specify the path where the dot tool can be
# found. If left blank, it is assumed the dot tool can be found in the path.

DOT_PATH =

# The DOTFILE_DIRS tag can be used to specify one or more directories that
# contain dot files that are included in the documentation (see the
# \dotfile command).

DOTFILE_DIRS =

# The MSCFILE_DIRS tag can be used to specify one or more directories that
# contain msc files that are included in the documentation (see the
# \mscfile command).

MSCFILE_DIRS =

# The DOT_GRAPH_MAX_NODES tag can be used to set the maximum number of
# nodes that will be shown in the graph. If the number of nodes in a graph
# becomes larger than this value, doxygen will truncate the graph, which is
# visualized by representing a node as a red box. Note that doxygen if the
# number of direct children of the root node in a graph is already larger than
# DOT_GRAPH_MAX_NODES then the graph will not be shown at all. Also note
# that the size of a graph can be further restricted by MAX_DOT_GRAPH_DEPTH.

DOT_GRAPH_MAX_NODES = 50

# The MAX_DOT_GRAPH_DEPTH tag can be used to set the maximum depth of the
# graphs generated by dot. A depth value of 3 means that only nodes reachable
# from the root by following a path via at most 3 edges will be shown. Nodes
# that lay further from the root node will be omitted. Note that setting this
# option to 1 or 2 may greatly reduce the computation time needed for large
# code bases. Also note that the size of a graph can be further restricted by
# DOT_GRAPH_MAX_NODES. Using a depth of 0 means no depth restriction.

MAX_DOT_GRAPH_DEPTH = 0
```

```
# Set the DOT_TRANSPARENT tag to YES to generate images with a transparent
# background. This is disabled by default, because dot on Windows does not
# seem to support this out of the box. Warning: Depending on the platform used,
# enabling this option may lead to badly anti-aliased labels on the edges of
# a graph (i.e. they become hard to read).

DOT_TRANSPARENT = NO

# Set the DOT_MULTI_TARGETS tag to YES allow dot to generate multiple output
# files in one run (i.e. multiple -o and -T options on the command line). This
# makes dot run faster, but since only newer versions of dot (>1.8.10)
# support this, this feature is disabled by default.

DOT_MULTI_TARGETS = NO

# If the GENERATE_LEGEND tag is set to YES (the default) Doxygen will
# generate a legend page explaining the meaning of the various boxes and
# arrows in the dot generated graphs.

GENERATE_LEGEND = YES

# If the DOT_CLEANUP tag is set to YES (the default) Doxygen will
# remove the intermediate dot files that are used to generate
# the various graphs.
```

DOT_CLEANUP = YES

13ANNEX. NAMING RULES FOR ECLIPSE

13.1Codename style settings

Name category	Capitalization	Word delimiter	Prefix	Suffix	Preview
Constant	Upper case	_			MY_CONSTANT
Variable	Lower camel case				myVariable
Class Field	Lower camel case				myField
Class Method	Camel case				MyMethod
Getter Method	Camel case		Get / Is		GetMyField
Setter Method	Camel case		Set		SetMyField
Include Guard (1)	Upper case			_H_	FILENAME_H_
Namespace (2)	Lower case				mynamespace

Table 9 Eclipse: codename styles

(1) “Include Guard” uses the name of the file as the guard.

(2) Namespace cannot be configured, but must be observed.

13.2Filename style settings

Name category	Capitalization	Word delimiter	Prefix	Suffix	Preview
C++ Header File	Original			.h	MyClass.h
C++ Source File	Original			.cpp	MyClass.cpp
C++ Test File	Original			.cpp	MyTestClass.cpp

Table 10 Eclipse: filename styles

14ANNEX. FORMATTING RULES FOR ECLIPSE

This annex lists the contents of the file *marte_cpp_formatting_rules.xml*, which defines the C++ formatting rules for C++ source files that will be used by Eclipse CDT. These rules are based on Eclipse CDT's K&R profile. [Last update on 05/06/2015]

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<profiles version="1">

<profile kind="CodeFormatterProfile" name="MARTe" version="1">

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_exception_specification"
value="do not insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_parenthesized_expression"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_for" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_brace_in_array_initializer"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_for" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_statements_compare_to_body" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_colon_in_base_clause" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_brace_in_array_initializer"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_switch" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_conditional_expression" value="34"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_colon_in_labeled_statement"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_empty_lines" value="false"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_cast" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_prefix_operator" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_after_opening_brace_in_array_initializer"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.comment.preserve_white_space_between_code_and_line_comments"
value="true"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_closing_paren_in_cast" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_between_empty_parens_in_method_declaration"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_method_invocation"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_throws_clause_in_method_declaration"
value="16"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_enum_declarations" value="do
not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_binary_operator" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_semicolon" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_binary_expression" value="16"/>
```

```
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_method_declaration" value="end_of_line"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_semicolon_in_for" value="do not
insert"/>
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_array_initializer" value="end_of_line"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_brace_in_method_declaration"
value="insert"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_method_declaration"
value="do not insert"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_between_empty_braces_in_array_initializer"
value="do not insert"/>
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_switch" value="end_of_line"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_expression_list" value="do not
insert"/>
<setting id="org.eclipse.cdt.core.formatter.keep_else_statement_on_same_line" value="false"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_colon_in_base_clause" value="do not
insert"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_base_types" value="do not
insert"/>
<setting id="org.eclipse.cdt.core.formatter.use_tabs_only_for_leading_indentations" value="false"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_between_empty_parens_in_method_invocation"
value="do not insert"/>
<setting id="org.eclipse.cdt.core.formatter.keep_then_statement_on_same_line" value="false"/>
<setting id="org.eclipse.cdt.core.formatter.insert_new_line_at_end_of_file_if_missing" value="do not
insert"/>
<setting id="org.eclipse.cdt.core.formatter.indent_body_declarations_compare_to_namespace_header"
value="false"/>
<setting id="org.eclipse.cdt.core.formatter.keep_empty_array_initializer_on_one_line" value="false"/>
<setting id="org.eclipse.cdt.core.formatter.indent_access_specifier_extra_spaces" value="0"/>
<setting id="org.eclipse.cdt.core.formatter.comment.never_indent_line_comments_on_first_column"
value="true"/>
<setting id="org.eclipse.cdt.core.formatter.insert_new_line_before_else_in_if_statement"
value="insert"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_expression_list"
value="insert"/>
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_block" value="end_of_line"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_bracket" value="do not insert"/>
<setting id="org.eclipse.cdt.core.formatter.alignment_for_base_clause_in_type_declaration" value="16"/>
<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_closing_angle_bracket_in_template_parameters"
value="do not insert"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_method_declaration_throws"
value="do not insert"/>
<setting id="org.eclipse.cdt.core.formatter.alignment_for_member_access" value="0"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_if" value="insert"/>
<setting id="org.eclipse.cdt.core.formatter.brace_position_for_namespace_declaration"
value="end_of_line"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_brace_in_block" value="insert"/>
<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_for" value="do not
insert"/>
```

```
<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_parenthesized_expression"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_parameters_in_method_declaration" value="83"/>

<setting id="org.eclipse.cdt.core.formatter.lineSplit" value="160"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_colon_in_default" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.number_of_empty_lines_to_preserve" value="1"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_method_declaration_parameters"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_between_empty_brackets" value="do not insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_closing_angle_bracket_in_template_arguments"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_method_declaration_throws"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_bracket" value="do not insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_opening_angle_bracket_in_template_arguments"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_template_arguments" value="do
not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_compact_if" value="16"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_prefix_operator" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_declarator_list" value="16"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_expressions_in_array_initializer" value="16"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_method_declaration"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_while" value="do not
insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_between_empty_parens_in_exception_specification"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_cast" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.continuation_indentation_for_array_initializer" value="2"/>

<setting id="org.eclipse.cdt.core.formatter.comment.min_distance_between_code_and_line_comment"
value="1"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_bracket" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_constructor_initializer_list" value="83"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_colon_in_case" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_brace_in_switch"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_catch" value="do not
insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_exception_specification"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_brace_in_array_initializer"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_declarator_list" value="do not
insert"/>
```

```
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_catch" value="insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_opening_angle_bracket_in_template_parameters"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_brace_in_type_declaration"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_semicolon_in_for" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_if" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_method_invocation_arguments"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.brace_position_for_block_in_case" value="end_of_line"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_postfix_operator" value="do not insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_after_closing_angle_bracket_in_template_parameters"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.compact_else_if" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.indent_switchstatements_compare_to_switch" value="false"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_overloaded_left_shift_chain" value="16"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_method_invocation_arguments"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_before_identifier_in_function_declaration"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_in_empty_block" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_method_declaration"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_declarator_list"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_colon_in_labeled_statement" value="do
not insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_statements_compare_to_block" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_brace_in_namespace_declaration"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_access_specifier_compare_to_type_header"
value="false"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_template_parameters"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_if" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_conditional_expression_chain" value="18"/>

<setting id="org.eclipse.cdt.core.formatter.tabulation.char" value="space"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_after_template_declaration" value="do not
insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_new_line_before_colon_in_constructor_initializer_list"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_enum_declarations"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_before_catch_in_try_statement"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.brace_position_for_type_declaration" value="end_of_line"/>
```

```
<setting id="org.eclipse.cdt.core.formatter.insert_space_after_assignment_operator" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_binary_operator" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.keep_imple_if_on_one_line" value="false"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_before_closing_brace_in_array_initializer"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.continuation_indentation" value="2"/>

<setting id="org.eclipse.cdt.core.formatter.insert_new_line_before_while_in_do_statement"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.put_empty_statement_on_new_line" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_closing_brace_in_block" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.format_guardian_clause_on_one_line" value="false"/>

<setting id="org.eclipse.cdt.core.formatter.indentation.size" value="0"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_after_opening_angle_bracket_in_template_parameters"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_switch" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_expression_list" value="0"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_method_invocation"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_colon_in_conditional" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_unary_operator" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_array_initializer"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_colon_in_case" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_base_types" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_assignment_operator" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_breaks_compare_to_cases" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_array_initializer" value="do
not insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_switchstatements_compare_to_cases" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.indent_declaration_compare_to_template_header"
value="false"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_method_declaration_parameters"
value="do not insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_after_opening_angle_bracket_in_template_arguments"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_opening_paren_in_catch" value="do not
insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_parenthesized_expression"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_arguments_in_method_invocation" value="18"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_while" value="insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_exception_specification"
value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_question_in_conditional" value="insert"/>
```

```
<setting id="org.eclipse.cdt.core.formatter.insert_space_before_opening_paren_in_switch"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_unary_operator" value="do not insert"/>

<setting id="org.eclipse.cdt.core.formatter.join_wrapped_lines" value="true"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_colon_in_conditional" value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_postfix_operator" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_assignment" value="16"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_while" value="do not
insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_after_comma_in_template_arguments"
value="insert"/>

<setting
id="org.eclipse.cdt.core.formatter.insert_space_after_closing_angle_bracket_in_template_arguments"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_comma_in_template_parameters" value="do
not insert"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_question_in_conditional"
value="insert"/>

<setting id="org.eclipse.cdt.core.formatter.indent_body_declarations_compare_to_access_specifier"
value="true"/>

<setting id="org.eclipse.cdt.core.formatter.alignment_for_enumerator_list" value="48"/>

<setting id="org.eclipse.cdt.core.formatter.tabulation.size" value="4"/>

<setting id="org.eclipse.cdt.core.formatter.insert_space_before_closing_paren_in_method_invocation"
value="do not insert"/>

</profile>

</profiles>
```

15ANNEX. CODE TEMPLATES FOR ECLIPSE

This annex lists the contents of the file *marte_cpp_code_templates.xml*, which defines the C++ templates for files and code fragments that will be used by Eclipse CDT. These code templates are based on those by default in Eclipse CDT. [Last update on 30/11/2015]

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><templates><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.constructorcomment_context" deleted="false"
description="Comment for created constructors" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.constructorcomment" name="constructorcomment">/*

 * @brief   ${todo} Write a brief description for the ${enclosing_type} constructor

 * @details ${todo} Write a detailed description for the ${enclosing_type} constructor

 */

       </template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.destructorcomment_context" deleted="false"
description="Comment for created destructors" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.destructorcomment" name="destructorcomment">/*

 * @brief   ${todo} Write a brief description for the ${enclosing_type} destructor

 * @details ${todo} Write a detailed description for the ${enclosing_type} destructor

 */</template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.filecomment_context" deleted="false" description="Comment
for created C/C++ files" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.filecomment"
name="filecomment"> * @date ${date} ${todo} Verify the value and format of the date

 * @author ${user} ${todo} Verify the name and format of the author

 *

 * @copyright Copyright 2015 F4E | European Joint Undertaking for ITER and

 * the Development of Fusion Energy ('Fusion for Energy').

 * Licensed under the EUPL, Version 1.1 or - as soon they will be approved

 * by the European Commission - subsequent versions of the EUPL (the "Licence")

 * You may not use this work except in compliance with the Licence.

 * You may obtain a copy of the Licence at: http://ec.europa.eu/idabc/eupl

 *

 * @warning Unless required by applicable law or agreed to in writing,

 * software distributed under the Licence is distributed on an "AS IS"

 * basis, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express

 * or implied. See the Licence permissions and limitations under the Licence.</template><template
autoinsert="false" context="org.eclipse.cdt.ui.text.codetemplates.typecomment_context" deleted="false"
description="Comment for created classes" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.typecomment" name="typecomment">/**

 * @brief TODO Write into the brief the main concept that the class represents

 * explained into a single line paragraph.

 * @details TODO Write into the details a detailed description of the class

 * that can span multiple lines on one paragraph. If more paragraphs are

 * needed, then put subsequent "@details" tags. This description shall explain

 * the  concept which the class represents alongside its responsibilities or

 * services it provides. Example: "This class represents a stack of doubles,

 * that is, a data structure which ..."
```

```
 */</template><template autoinsert="true"
context="org.eclipse.cdt.ui.text.codetemplates.fieldcomment_context" deleted="false"
description="Comment for fields" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.fieldcomment"
name="fieldcomment">/*
 *
 */
       </template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.methodcomment_context" deleted="false"
description="Comment for methods" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.methodcomment" name="methodcomment">/*
 * @brief   ${todo} Write a brief description for the ${enclosing_type} method
 * @details ${todo} Write a detailed description for the ${enclosing_type} method
 */
       </template><template autoinsert="false"
context="org.eclipse.cdt.core.cxxSource.contenttype_context" deleted="false" description="Default
template for newly created C++ source files" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.cppsourcefile" name="Default C++ source template">/**
 * @file ${file_name}
 * @brief Source file for class ${file_base}
${filecomment}

 * @details This source file contains the definition of all the methods for
 * the class ${file_base} (public, protected, and private). Be aware that some
 * methods, such as those inline could be defined on the header file, instead.
 */

#define DLL_API

/*---------------------------------------------------------------------------*/
/*                        Standard header includes                           */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                        Project header includes                            */
/*---------------------------------------------------------------------------*/

${includes}

/*---------------------------------------------------------------------------*/
/*                          Static definitions                               */
/*---------------------------------------------------------------------------*/

namespace {

}
```

```
/*---------------------------------------------------------------------------*/
/*                           Method definitions                              */
/*---------------------------------------------------------------------------*/


${namespace_begin}


${declarations}


${namespace_end}
	</template><template autoinsert="true"
context="org.eclipse.cdt.core.cxxSource.contenttype_context" deleted="false" description="Default
template for newly created C++ test files" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.cpptestfile" name="Default C++ test template">${filecomment}
${includes}


${namespace_begin}


${declarations}


${namespace_end}
	</template><template autoinsert="false"
context="org.eclipse.cdt.core.cxxHeader.contenttype_context" deleted="false" description="Default
template for newly created C++ header files" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.cppheaderfile" name="Default C++ header template">/**
 * @file ${file_name}
 * @brief Header file for class ${file_base}
${filecomment}


 * @details This header file contains the declaration of the class ${file_base}
 * with all of its public, protected and private members. It may also include
 * definitions for inline methods which need to be visible to the compiler.
 */


#ifndef ${include_guard_symbol}
#define ${include_guard_symbol}


/*---------------------------------------------------------------------------*/
/*                        Standard header includes                           */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/*                        Project header includes                            */
/*---------------------------------------------------------------------------*/


${includes}
```

```
/*---------------------------------------------------------------------------*/
/*                           Class declaration                               */
/*---------------------------------------------------------------------------*/

${namespace_begin}

${declarations}

${namespace_end}

/*---------------------------------------------------------------------------*/
/*                        Inline method definitions                          */
/*---------------------------------------------------------------------------*/

#endif /* ${include_guard_symbol} */
       </template><template autoinsert="false"
context="org.eclipse.cdt.core.cSource.contenttype_context" deleted="false" description="Default template
for newly created C source files" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.csourcefile"
name="Default C source template">/**
 * @file ${file_name}
 * @brief Source file for module ${file_base}
${filecomment}

 * @details This source file contains the definition of all the functions for
 * the module ${file_base}, both public and private. Be aware that some
 * functions, such as those inline could be defined on the header file,
 * instead.
 */

/*---------------------------------------------------------------------------*/
/*                          Standard header includes                         */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                          Project header includes                          */
/*---------------------------------------------------------------------------*/

${includes}

/*---------------------------------------------------------------------------*/
/*                            Static definitions                             */
/*---------------------------------------------------------------------------*/
```

```
/*---------------------------------------------------------------------------*/
/*                           Function definitions                            */
/*---------------------------------------------------------------------------*/


${declarations}

</template><template autoinsert="false" context="org.eclipse.cdt.core.cHeader.contenttype_context"
deleted="false" description="Default template for newly created C header files" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.cheaderfile" name="Default C header template">/**
 * @file ${file_name}
 * @brief Header file for module ${file_base}
${filecomment}

 * @details This header file contains the declaration of all the public
 * functions for the module ${file_base}. It may also include definitions
 * for inline methods which need to be visible to the compiler.
 *
 * @details TODO Explain the general purpose of this module
 */

#ifndef ${include_guard_symbol}
#define ${include_guard_symbol}

/*---------------------------------------------------------------------------*/
/*                        Standard header includes                           */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/*                        Project header includes                            */
/*---------------------------------------------------------------------------*/

${includes}

/*---------------------------------------------------------------------------*/
/*                           Function declarations                           */
/*---------------------------------------------------------------------------*/

${declarations}

/*---------------------------------------------------------------------------*/
/*                        Inline method definitions                          */
/*---------------------------------------------------------------------------*/

#endif /* ${include_guard_symbol} */
```

```
        </template><template autoinsert="true"
context="org.eclipse.cdt.ui.text.codetemplates.namespace_context" deleted="false" description="Beginning
of namespace declaration" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.namespace_begin"
name="namespace_begin">namespace ${namespace_name} {</template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.namespace_context" deleted="false" description="End of
namespace declaration" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.namespace_end"
name="namespace_end">}

        </template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.class_context" deleted="false" description="Code in
created class definitions" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.class_body"
name="class_body">//${todo} Add the macro DLL_API to the class declaration (i.e. class DLL_API
${enclosing_type})

${declarations}</template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.methodbody_context" deleted="false" description="Code in
created method stubs" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.methodbody"
name="methodbody">//${todo} Auto-generated method stub

${body_statement}

        </template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.constructorbody_context" deleted="false"
description="Code in created constructor stubs" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.constructorbody" name="constructorbody">//Auto-generated
constructor stub for ${enclosing_type}

${body_statement}


//${todo} Verify if manual additions are needed here</template><template autoinsert="false"
context="org.eclipse.cdt.ui.text.codetemplates.destructorbody_context" deleted="false" description="Code
in created destructor stubs" enabled="true" id="org.eclipse.cdt.ui.text.codetemplates.destructorbody"
name="destructorbody">//Auto-generated destructor stub for ${enclosing_type}&#13;

${body_statement}&#13;

&#13;

//${todo} Verify if manual additions are needed here</template><template autoinsert="true"
context="org.eclipse.cdt.core.asmSource.contenttype_context" deleted="false" description="Default
template for newly created assembly files" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.asmsourcefile" name="Default assembly template">${filecomment}

        </template><template autoinsert="true"
context="org.eclipse.core.runtime.text.contenttype_context" deleted="false" description="Default
template for newly created text files" enabled="true"
id="org.eclipse.cdt.ui.text.codetemplates.textfile" name="Default text file template">${file_name}

Created on: ${date}

Author: ${user}
```

</template></templates>

16ANNEX. EDITOR TEMPLATES FOR ECLIPSE

This annex lists the contents of the file *marte_cpp_editor_templates.xml*, which defines the C++ templates for files and code fragments that will be used by Eclipse CDT. These code templates are based on those by default in Eclipse CDT. [Last update on 27/11/2015]

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><templates><template autoinsert="true"
context="org.eclipse.cdt.ui.text.templates.comment" deleted="false" description="author name"
enabled="true" id="org.eclipse.cdt.ui.text.templates.comment.author" name="author">author
${user}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c"
deleted="false" description="catch block" enabled="true"
id="org.eclipse.cdt.ui.text.templates.cpp.catch" name="catch">catch (${Exception} e) {

        ${cursor}

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="class declaration" enabled="true" id="org.eclipse.cdt.ui.text.templates.cpp.class"
name="class">class DLL_API ${name} {

public:

        ${cursor}


private:

};</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="default multiline comment" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.comment"
name="comment">

/*

 * author ${user}

 *

 * To change this generated comment edit the template variable "comment":

 * Window &gt; Preferences &gt; C/C++ &gt; Editor &gt; Templates.

 */

</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="do while statement" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.do" name="do">do
{

        ${line_selection}${cursor}

} while (${condition});</template><template autoinsert="true"
context="org.eclipse.cdt.ui.text.templates.c" deleted="false" description="else block" enabled="true"
id="org.eclipse.cdt.ui.text.templates.c.else" name="else">else {

        ${cursor}

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="else if block" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.elseif"
name="elseif">else if (${condition}) {

        ${cursor}

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="for loop" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.for" name="for">for
(${var} = 0; ${var} &lt; ${max}; ++${var}) {

        ${line_selection}${cursor}

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="for loop with temporary variable" enabled="true"
id="org.eclipse.cdt.ui.text.templates.cpp.for" name="for">for (int ${var} = 0; ${var} &lt; ${max};
++${var}) {

        ${line_selection}${cursor}
```

```
}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="if statement" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.if" name="if">if
(${condition}) {

	${line_selection}${cursor}

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="if else statement" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.ifelse"
name="ifelse">if (${condition}) {

	${cursor}

} else {


}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="main method" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.main" name="main">int
main(int argc, char **argv) {

	${cursor}

}

</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="namespace declaration" enabled="true" id="org.eclipse.cdt.ui.text.templates.cpp.namespace"
name="namespace">namespace ${name} {


${cursor}


}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="create new object" enabled="true" id="org.eclipse.cdt.ui.text.templates.cpp.new"
name="new">${type} ${name} = new ${type}(${arguments});</template><template autoinsert="true"
context="org.eclipse.cdt.ui.text.templates.c" deleted="false" description="print to standard error"
enabled="true" id="org.eclipse.cdt.ui.text.templates.c.fprintf" name="stderr">fprintf(stderr,
${cursor});</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c"
deleted="false" description="print to standard output" enabled="true"
id="org.eclipse.cdt.ui.text.templates.c.printf" name="stdout">printf(${cursor});</template><template
autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false" description="switch case
statement" enabled="true" id="org.eclipse.cdt.ui.text.templates.c.switch" name="switch">switch (${key})
{

	case ${value}:

		${cursor}

		break;

	default:

		break;

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="try catch block" enabled="true" id="org.eclipse.cdt.ui.text.templates.cpp.try"
name="try">try {

	${line_selection}${cursor}

} catch (${Exception} e) {

}</template><template autoinsert="true" context="org.eclipse.cdt.ui.text.templates.c" deleted="false"
description="using a namespace" enabled="true" id="org.eclipse.cdt.ui.text.templates.cpp.using"
name="using">using namespace ${name};

</template></templates>
```