**Commnets Rules For**

**Enterprise Architect - Detailed Design Generator**

OBJECT

This document describes a set of rules for writing comments in code so that they are parsed correctly by the **Enterprise Architect Detailed Design Generator** tool.

|  |  |  |
| --- | --- | --- |
| STATUS: | For comments | ✓ |
|  | For application |  |

|  |  |  |
| --- | --- | --- |
| Established by | Reviewed by | Approved by |
| Name: Puscasu David  Date: 24.05.2023  Visa: | Name:  Date:  Visa: | Name:  Date:  Visa: |

# EVOLUTION OF THE DOCUMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue** | **Date** | **Author** | **Rev.** | **Motive and nature of the modifications** |
|  | 24.05.2023 | Puscasu  David | 1.1 | First revision |
|  | 30.05.2023 | Puscasu  David | 1.2 | Add **FCR\_009** rule. |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**CONTENTS**

EVOLUTION OF THE DOCUMENT 2

1. RELEVANT DOCUMENTS 4

2. GLOSSARY OF TERMS / EXPRESSIONS 4

3. About Detailed Design Necessary Documents 4

4. COMMENTS RULES 5

4.1. File-level formatting rules (FLR) 6

4.2. Macro comment formatting rules (MCR) 8

4.3. Variable comment formatting rules (VCR) 8

4.4. Function comment formatting rules (FCR) 9

# RELEVANT DOCUMENTS

|  |  |  |
| --- | --- | --- |
| **N°** | **Title** | **Reference** |
|  | HowTo\_Generate\_Detailed\_Design\_in\_EnterpriseArchitect | <HowTo_Generate_Detailed_Design_in_EnterpriseArchitect.docx> |
|  | **Example\_Module.c\_template** |  |
|  | **Example\_Module.h\_template** |  |

# GLOSSARY OF TERMS / EXPRESSIONS

|  |  |
| --- | --- |
| ***Term*** | ***Definition*** |
| ***BAT*** | *Batch File (Windows batch file extension)* |
|  |  |
|  |  |

# About Detailed Design Necessary Documents

In order to generate the Detailed Design Document, we need to prepare several documents considered to be as input data for detailed design generation.

This documents will have the extension **c\_template** and **h\_template** and will contain a small description of the source code of the respective software component.

For example, if a software component has a total of 5 files, 3 source files with .c extension and 2 header files, **an each c\_template file is required for each .c source file and an each h\_template file is required for each .h header files.**

This c\_template and h\_template files shall be located in each software component Design folder in order for the tool to generate.

It is also recommended when creating a new software component, also to write first c\_template and h\_template files and after them to write source files ( .c files ) and header files ( .h files ).

# COMMENTS RULES

For the Enterprise Architect - Detailed Design Generator tool to work correctly, we need the **.c\_template** or **.h\_template** files that the tool parses to follow certain rules for writing both the file header and the comments for each entity in that file (macro, variable, or function).

To this scope, a series of rules structured in the following categories have been created:

* **File-level formatting rules** - what you should and should not have in the file header or anywhere in the file other than an entity such as a macro, variable, or function
* **Macro comment formatting rules** - how to format, what you should and should not have in the comment assigned to a macro
* **Variable comment formatting rules** - how to format, what you should and should not have in the comment assigned to a variable
* **Function comment formatting rules** - how to format, what to have and what not to have in the comment assigned to a function, or how to write the content of the function

**Formalism used**

Each rule is stated with this formalism:

**Identifier:** Represents the prefix section name (i.e. FLR) and the number of the rule by order of creation (i.e. 001).

**Category:** Defines **Mandatory** or **Required** or **Advisory**

*Rationale:* Gives more information on the rule. Note that information is part of the rule and shall be respected absolutely.

*Example:* Examples are given to validate the rule.

**Application field**.

This *Coding Comments Standard* defines three rules categories to apply in the c\_template and h\_template files:

* **Mandatory**: The c\_template and h\_template files shall comply with comments rules categorized as **Mandatory**. Violations must be fixed because the Enterprise Architect - Detailed Design Generator tool will generate the Detailed Design project incorrectly, or the execution of the tool will crash with an error.
* **Required**: The c\_template and h\_template files may follow the rules for comments classified as **Required**. If these rules are not followed the Enterprise Architect - Detailed Design Generator tool will add some default data to the Detailed Design project.
* **Advisory**: These rules can be considered more as recommendations

The three colours: magenta, green and yellow have been chosen to differentiate the Coding Comments Standard rules.

## File-level formatting rules (FLR)

**FLR\_001:** Thec\_template and h\_template files shall NOT contain any "@" in details section of the beginning of the file.

**Category**: **Mandatory**

**Example - wrong:**

|  |
| --- |
| /\*!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @details  <Describes details of this module 'Template.c' file within overall  context of component implementation>  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

**FLR\_002:** c\_template and h\_template files shall NOT contain any defines associated to memory sections and includes of memory map headers.

**Category**: **Mandatory**

**Example - wrong:**

|  |
| --- |
| LOCAL uint32 acc\_u32AlarmRegistersUsefulData = KU32\_ZERO;  #define ACC\_STOP\_SEC\_VAR\_INIT\_32\_ASIL\_A  #include "ACC\_MemMap.h"  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DEFINITION OF EXPORTED VARIABLES  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #define ACC\_START\_SEC\_VAR\_INIT\_BOOL\_ASIL\_A  #include "ACC\_MemMap.h"  /\*\*  \* \brief  \* A flag used to notify that the TDK accelerometer chip has some errors.  \* \initialization  \* False.  \* \range  \* 0..1.  \*/  EXPORTED boolean ACC\_bAlarmPresentFlag = KB\_FALSE; |

**FLR\_003:** The doxygen comment of a macro, variable, or function, MUST NOT contain any character, **"-"**, **"\*"**, **">"**, **"<"**, **"1"**, **"2"** etc., at the beginning of a line for a better document generation.

If needed to set a list, please use the character **"o"** at the beginning of every line in the list.

Also, the beginning of a list must be marked with the **":"** character, and the items in the list must be separated by the **";"** character.

**Category**: **Mandatory**

**Example - wrong:**

|  |
| --- |
| /\*\*  \* \brief  \* Maximum time for POST RUN state to monitor:  \* NVP\_u32TPostRun - return maximum time to stay in POST RUN State in cycles  \* - 10 ms = 1 cycle (GPT Interrupt)  \* - 1 seconds = 1000 ms = 100 cycles (GPT Interrupts)  \* - 10 seconds = 10000 ms = 1000 cycles (GPT Interrupts)  \* - 100 seconds = 100000 ms = 10000 cycles (GPT Interrupts)  \*/ |

**Example - correct:**

|  |
| --- |
| /\*\*  \* \brief  \* Maximum time for POST RUN state to monitor:  \* NVP\_u32TPostRun - return maximum time to stay in POST RUN State in cycles:  \* o 10 ms = 1 cycle (GPT Interrupt);  \* o 1 seconds = 1000 ms = 100 cycles (GPT Interrupts);  \* o 10 seconds = 10000 ms = 1000 cycles (GPT Interrupts);  \* o 100 seconds = 100000 ms = 10000 cycles (GPT Interrupts);  \*/ |

**FLR\_004:** All keyword description phrases must be indented at the same level unless a list is present in the description.

**Category**: **Required**

**Example - wrong:**

|  |
| --- |
| \* \brief  \* An example of INCORRECT description.  \* Because this phrase is here.  \* And this phrase is here.  \* And this phrase is here. |

**Example - correct:**

|  |
| --- |
| \* \brief  \* An example of CORRECT description.  \* Because this phrase is here.  \* And this phrase is here  \* And this phrase is here. |

**FLR\_005:** The Not Applicable value of the parameters shall be written as **“NA.”**

**Category**: **Required**

**Example - wrong:**

|  |
| --- |
| \inputparam  \*        Name: kpstCyclicList;  \*        Type: const ATM\_stCyclicListEltType \*const (struct[uint16, uint8, uint8]);  \*        Description: Pointer to the head of the cyclic list constant to play;  \*        Range: N/A; |

**Example - correct:**

|  |
| --- |
| \inputparam  \*        Name: kpstCyclicList;  \*        Type: const ATM\_stCyclicListEltType \*const (struct[uint16, uint8, uint8]);  \*        Description: Pointer to the head of the cyclic list constant to play;  \*        Range: NA.; |

**FLR\_006:** Range value of parameters shall be written with **“..”** as delimiter

**Category**: **Required**

**Example - wrong:**

|  |
| --- |
| \inputparam  \*     Name: u8TransitionStatus;  \*     Type: uint8;  \*     Description: State of transition;  \*     Range: 0x55 - 0xAA; |

**Example - correct:**

|  |
| --- |
| \inputparam  \*     Name: u8TransitionStatus;  \*     Type: uint8;  \*     Description: State of transition;  \*     Range: 0x55..0xAA; |

**FLR\_006:** If the range value of parameters is a list, each element should be on a new line, and the whole list should have a tab relative to the range tag

**Category**: **Advisory**

**Example - wrong:**

|  |
| --- |
| \outputparam  \*     Name: u8TestResult;  \*     Type: u8TestResultType \* (uint8);  \*     Description: The reported status of the Autotest;  \*     Range: KU8\_ATM\_TEST\_OK, KU8\_ATM\_TEST\_NOK, KU8\_ATM\_TEST\_NOT\_DECIDED, KU8\_ATM\_TEST\_NON\_EXISTENT, KU8\_ATM\_TEST\_BAD\_CONDITIONS; |

**Example - correct:**

|  |
| --- |
| \outputparam  \*     Name: u8TestResult;  \*     Type: u8TestResultType \* (uint8);  \*     Description: The reported status of the Autotest;  \*     Range:  \*        KU8\_ATM\_TEST\_OK  \*        KU8\_ATM\_TEST\_NOK  \*        KU8\_ATM\_TEST\_NOT\_DECIDED  \*        KU8\_ATM\_TEST\_NON\_EXISTENT  \*        KU8\_ATM\_TEST\_BAD\_CONDITIONS; |

**FLR\_006:** If the range value of parameters is the whole range of the type, it should be written as type\_MAX

**Category**: **Advisory**

**Example - wrong:**

|  |
| --- |
| \range   \*      0.. 18446744073709551615   \*/ |

**Example - correct:**

|  |
| --- |
| \* \range   \*      0..KU64\_MAX   \*/ |

**FLR\_007:** Functions that are declared in the .h but are not defined in the .c files must be deleted from h\_templates

**Category**: **Mandatory**

## Macro comment formatting rules (MCR)

**MCR\_001:** Comments of macros shall have the following Doxygen Tags:

- **\brief**

**Category**: **Mandatory**

**Example - correct:**

|  |
| --- |
| /\*\*  \* \brief  \* Defined macro from H file  \*/  #define DATATYPE\_MACRO\_DEFINED\_FROM\_H\_FILE ((uint8) 0x01) |

**MCR\_002: \brief** tag of a macro comment MUST respect the rule **FLR\_003** and **FLR\_004.**

**Category**: **Mandatory**

## Variable comment formatting rules (VCR)

**VCR\_001:** Comments of variables shall have the following Doxygen Tags:

**- \brief**

**- \initialization**

**- \range**

**Category**: **Mandatory**

**Example - correct:**

|  |
| --- |
| /\*\*  \* \brief  \* This variable is used to give an example description of a variable.  \* \initialization  \* Initialization description or value.  \* \range  \* 0..255 or anything.  \*/  EXPORTED uint8 u8VariableExample = ZERO; |

**VCR\_002: \brief**, **\initialization** and **\range** tag of a variable comment MUST respect the rule **FLR\_003** and **FLR\_004.**

**Category**: **Mandatory**

## Function comment formatting rules (FCR)

**FCR\_001:** Comments of functions shall have the following Doxygen Tags:

**- \brief**

**- \inputparam**

**- Name:**

**- Type:**

**- Description:**

**- Range:**

**- \outputparam**

**- Name:**

**- Type:**

**- Description:**

**- Range:**

**- \exception**

**- \pre**

**- \post**

**- \return**

**- \staticaspect**

**- \dynamicaspectcaller**

**- \dynamicaspectdescription**

**- \constrains**

**- \ddesignrequirement**

**- \archrequirement**

**Category**: **Mandatory**

**Example - correct:**

|  |
| --- |
| /\*\*  \* \brief  \* An example of description for local function. Belongs to First\_Example\_Module.  \* \inputparam  \* Name: variable1;  \* Type: uint8;  \* Description: This is a description for variable1;  \* Range: 0..255;  \* \inputparam  \* Name: variable2;  \* Type: uint16;  \* Description: This is a description for variable 2;  \* Range: 0, 1, 2;  \* \outputparam  \* Name: ouput\_param;  \* Type: uint8 \*;  \* Description: This is an ouput parameter.;  \* Range: OK, NOK, other\_values;  \* \exception  \* This function has the following exceptions  \* \pre  \* This function has the following preconditions  \* \post  \* This function has the following postconditions  \* \return  \* This function has no return.  \* \staticaspect  \* Write here the static aspect of the function.  \* \dynamicaspectcaller  \* Write here who calls this function.  \* \dynamicaspectdescription  \* Write here a description caller of this function.  \* \constrains  \* Write here if you have any constraints.  \* \ddesignrequirement  \* DSG\_Example\_Module\_service\_c\_template  \* \archrequirement  \* ARCH\_TEST\_REQ  \*\*/  EXPORTED void Example\_Module\_service\_c\_template (uint8 variable1, uint16 variable2, uint8 \* ouput\_param ) |

**FCR\_002:** For function descriptions, there MUST be an **/inputparam** tag in the comment for **each input parameter** and an **/outputparam** tag in the comment for **each output parameter** that the function has.

**Category**: **Mandatory**

**Example - correct:**

|  |
| --- |
| /\*\*  \* \brief  \* An example of description for local function. Belongs to First\_Example\_Module.  \* \inputparam  \* Name: variable1;  \* Type: uint8;  \* Description: This is a description for variable1;  \* Range: 0..255;  \* \inputparam  \* Name: variable2;  \* Type: uint16;  \* Description: This is a description for variable 2;  \* Range: 0, 1, 2;  \* \outputparam  \* Name: ouput\_param;  \* Type: uint8 \*;  \* Description: This is an ouput parameter.;  \* Range: OK, NOK, other\_values;  \* \exception  \* This function has the following exceptions  ...  \* \archrequirement  \* ARCH\_TEST\_REQ  \*\*/  EXPORTED void Example\_Module\_service\_c\_template (uint8 variable1, uint16 variable2, uint8 \* ouput\_param ) |

**FCR\_003:** For functions descriptions, each INPUT or OUTPUT parameter shall have the following tags:

**- Name:**

**- Type:**

**- Description:**

**- Range:**

Each tag shall end with character **":"** and the content of the tag shall end with character **";"**.

**Category**: **Mandatory**

**Example - wrong:**

|  |
| --- |
| /\*\*  ...  \* \inputparam  \* Name variable2  \* Type uint16  \* Description This is a description for variable 2  \* Range 0, 1, 2  \* \outputparam  \* Name ouput\_param  \* Type uint8 \*  \* Description This is an ouput parameter.  \* Range OK, NOK, other\_values  ...  \*\*/ |

**Example - correct:**

|  |
| --- |
| /\*\*  ...  \* \inputparam  \* Name: variable2;  \* Type: uint16;  \* Description: This is a description for variable 2;  \* Range: 0, 1, 2;  \* \outputparam  \* Name: ouput\_param;  \* Type: uint8 \*;  \* Description: This is an ouput parameter.;  \* Range: OK, NOK, other\_values;  ...  \*\*/ |

**FCR\_004:** c\_template and h\_template files shall NOT contain any **"TASK"**, **"FUNC"** in functions definitions. These keywords shall be integrated in Function Name.

In this case, the function definition is replaced according to the examples below:

**TASK(OsTask\_Bsw\_ReadWrite\_AllProcess) => void TASK\_OsTask\_Bsw\_ReadWrite\_AllProcess()**

**EXPORTED** **FUNC(void, eCS\_CODE) eCS\_runMainFunction() => EXPORTED void eCS\_runMainFunction()**

**Category**: **Mandatory**

**FCR\_005: \brief**, **\pre**, **\post**, **\return**, **\staticaspect**, **\constrains**, **\dynamicaspectcaller**, **\dynamicaspectdescription**, **\ddesignrequirement** and **\archrequirement** tag of a function comment MUST respect the rule **FLR\_003** and **FLR\_004.**

**Category**: **Mandatory**

**FCR\_006: In Function descriptions, local variables** defined inside in function **shall be removed.**

**Explanation**: In order to maintain generated flowchart in a reasonable dimensions, we need to cut every unusable information.

**Category**: **Required**

**Example - wrong:**

|  |
| --- |
| EXPORTED void Example\_Module\_service\_c\_template(uint8 variable1, uint16 variable2, uint8 \*ouput\_param)  {  uint8 u8VariableDefinition;  uint8 u8AnotherVariableDefinition;  /\* Function code here \*/  /\* Assign to parameter\_1 upper defined value \*/  Variable = value\_from\_defined\_macro;  /\* Check values of parameter\_1 \*/  if (variable\_condition)  {  /\* Increment it \*/  increment\_variable;  }  else  {  /\* Decrement it \*/  decrement\_variable;  }  } |

**FCR\_007:** Functions description shall simplify operations and conditions to ease flowchart understanding.

**Category**: **Advisory**

**Example - wrong:**

|  |
| --- |
| EXPORTED void Example\_Module\_service\_c\_template(uint8 variable1, uint16 variable2, uint8 \*ouput\_param)  {  /\* Function code here \*/  /\* Assign to parameter\_1 upper defined value \*/  Variable = value\_from\_defined\_macro;  /\* Check values of parameter\_1 \*/  if (((u16Center - pkau16CurState[KU8\_THREE]) <= u16AdcVal) && ((u16Center + pkau16CurState[KU8\_THREE]) >= u16AdcVal))  {  ...  }  else  {  ...  }  } |

**Example - correct:**

|  |
| --- |
| EXPORTED void Example\_Module\_service\_c\_template(uint8 variable1, uint16 variable2, uint8 \*ouput\_param)  {  /\* Function code here \*/  /\* Assign to parameter\_1 upper defined value \*/  Variable = value\_from\_defined\_macro;  /\* Check values of parameter\_1 \*/  if (check\_status\_of\_pkau16CurState)  {  ...  }  else  {  ...  }  } |

**FCR\_008: Functions implementation shall be constructed on the C programming language syntax** for tool to generate flowchart. It is necessary to pay attention to have:

1. **";"** character at the end of each source code line

2. **"{"** and **"}"** characters when defining a function or an "if", "for", "while", "case" logic.

3. **"/\*"** and **"\*/"** characters when adding a comment.

**Category**: **Mandatory**

**FCR\_009:** The function implementation **must contain only lines of code that do not exceed 256 characters** in length.

**Category**: **Mandatory**

**FCR\_010:** **In Function descriptions,** the **comments code must not contain** **apostrophe “ ‘ ”**, because that will generate an error in application.

**Category**: **Mandatory**