Manual Review of MISRA Rules

Customer

Project

**Revision History**

| **Version** | **Date** | **Change Description / Reason** | **Author** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 4

1.1 Definitions, Glossary 4

2. Checking MISRA Rules manually 4

3. Results of the review 5

4. References to Other Documents 6

# Introduction

This plan applies to project <Project name> and is binding for all phases of the project. The document forms the basis for checking MISRA rules which can’t be covered automatically with the static code analysis tool QA-C.

## Definitions, Glossary

If applicable, own project-specific terms and abbreviations.

# Checking MISRA Rules manually

The following table lists all MISRA rules, whose compliance has to be checked in a manual review. All MISRA rules listed here are part of the subset which is required by the HIS . Each software module has to be checked by the responsible SW Developer himself. The check of the following MISRA rules is not required for autogenerated code by TargetLink.

MISRA-C:1998

|  |  |
| --- | --- |
| **Number** | **MISRA Rule** |
| **2** | Code written in languages other than C should only be used if there is a defined interface standard for object code to which the compilers/assemblers for both languages conform. |
| **10** | Sections of code should not be ‘commented out’. |
| **41** | The implementation of integer division in the chosen compiler should be determined, documented and taken into account. |
| **109** | Overlapping variable storage shall not be used. |
| **116** | All libraries used in production code shall be written to comply with the provisions of this document, and shall have been subject to appropriate validation. |
| **117** | The validity of values passed to library functions shall be checked. |

MISRA-C:2004

|  |  |
| --- | --- |
| **Number** | **MISRA Rule** |
| **1.3** | Multiple compilers and/or languages shall only be used if there is a common defined interface standard for object code to which the languages/compilers/assemblers conform. |
| **1.4** | The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers. |
| **1.5** | Floating-point implementations should comply with a defined floating-point standard. |
| **2.4** | Sections of code should not be 'commented out'. |
| **3.2** | The character set and the corresponding encoding shall be documented. |
| **3.3** | The implementation of integer division in the chosen compiler should be determined, documented and taken into account. |
| **3.5** | If it is being relied upon, the implementation-defined behaviour and packing of bitfields shall be documented. |
| **3.6** | All libraries used in production code shall be written to comply with the provisions of this document, and shall have been subject to appropriate validation. |
| **5.7** | No identifier name should be reused. |
| **10.5** | If the bitwise operators ~ and << are applied to an operand of underlying type unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand. |
| **17.1** | Pointer arithmetic shall only be applied to pointers that address an array or array element. |
| **17.2** | Pointer subtraction shall only be applied to pointers that address elements of the same array. |
| **17.3** | >, >=, <, <= shall not be applied to pointer types except where they point to the same array. |
| **18.2** | An object shall not be assigned to an overlapping object. |
| **18.3** | An area of memory shall not be reused for unrelated purposes. |
| **20.3** | The validity of values passed to library functions shall be checked. |

The following table shows which C files have been analyzed in a manual review and in which table the dedicated results can be found:

|  |  |
| --- | --- |
| **Name of the C File** | **Dedicated Table** |
| Here the name of the C file has to be inserted | Here a link to the results has to be inserted. The link can be created with “Insert 🡪 Reference 🡪 Create cross-reference” |

# Results of the review

In the following tables all deviations against the MISRA rules listed above have to be entered. Furthermore a reason has to be given why the deviation can’t be eliminated. The caption of a table can be created by choosing “Insert 🡪 Reference 🡪 Caption”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Elimination possible?** | |  |
| **Number of MISRA Rule** | **Row** | **Code** | **Yes** | **No** | **Comment** |
| 117 | 1 | strcat(tempStr, asciiBuffer); |  | X | The variable asciiBuffer contains status data from the eeprom. An additional checking of the data is not required in this case, because the eeprom contains only valid data. |
|  | 2 |  |  |  |  |
|  |  |  |  |  |  |

Table 1 - Name of the C File

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Elimination possible?** | |  |
| **Number of MISRA Rule** | **Row** | **Code** | **Yes** | **No** | **Comment** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table 2 - Name of the C File

# References to Other Documents

*State important references if applicable*