PRQA
Programming Research

COMPONENT MANUAL

M3CM 1.1.0

December, 2015

IMPORTANT NOTICE

DISCLAIMER OF WARRANTY

The staff of Programming Research Ltd. have taken due care in preparing this document which is believed to be accurate at the time of printing. However, no liability can be accepted for errors or omissions nor should this document be considered as an expressed or implied warranty that the products described perform as specified within.

COPYRIGHT NOTICE

TRADEMARKS

PRQA, the PRQA logo , QA·C, QA·C++ and High Integrity C++ (HIC++) are trademarks of *Programming Research* Ltd.
"MISRA", "MISRA C" and "MISRA C++" are registered trademarks of MIRA Limited, held on behalf of the MISRA Consortium.
Yices is a registered trademark of SRI International.
Windows is a registered trademark of Microsoft Corporation.

CONTACTING PROGRAMMING RESEARCH LTD

For technical support, contact your nearest Programming Research Ltd authorized distributor or you can contact Programming Research's head office:

by telephone on +44 (0) 1932 888 080
by fax on +44 (0) 1932 888 081
or by webpage: www.programmingresearch.com/services/contact-support/

Contents

1 Introduction **1**
1.1 MISRA C . . . 1
1.2 The MISRA C:2012 Compliance Module . . . 1
1.3 Installation . . . 2

2 M3CM Components **3**
2.1 Dataflow . . . 3

3 Using M3CM **4**
3.1 Directives and Rules . . . 4
3.2 Configuring M3CM for Conformance to C90 or C99 . . . 4
3.3 The QA·C Message System . . . 5
3.4 Mapping Messages to MISRA Directives and Rules . . . 5

4 Global Checks **7**

5 Cross Reference Listing **8**

6 Rule Enforcement **27**
6.1 Enforced Rules - Advisory . . . 27
6.2 Unenforced Rules - Advisory . . . 28
6.3 Not Statically Enforceable Rules - Advisory . . . 29
6.4 Enforced Rules - Mandatory . . . 29
6.5 Unenforced Rules - Mandatory . . . 30
6.6 Enforced Rules - Required . . . 30
6.7 Unenforced Rules - Required . . . 39
6.8 Not Statically Enforceable Rules - Required . . . 39
6.9 Rule Enforcement Summary . . . 40

List of Figures

List of Tables

6.1 M3CM Enforcement Rules - Advisory 27
6.2 M3CM Unenforced Rules - Advisory 28
6.3 M3CM Not Statically Enforceable Rules - Advisory 29
6.4 M3CM Enforced Rules - Mandatory 29
6.5 M3CM Unenforced Rules - Mandatory 30
6.6 M3CM Enforced Rules - Required 31
6.7 M3CM Unenforced Rules - Required 39
6.8 M3CM Not Statically Enforceable Rules - Required 39
6.9 M3CM Rule Enforcement Summary 40

1 Introduction

1.1 MISRA C

The original MISRA C Guidelines were published in April 1998 by the Motor Industry Research Association in a document entitled "Guidelines For The Use Of The C Language In Vehicle Based Software". In 2004, an updated version of this document was published under the title "MISRA C:2004 Guidelines for the use of the C language in critical systems". In July 2007 a Technical Corrigendum was issued to address a number of issues which required correction or clarification. This was incorporated into the base document when MISRA C2 was reprinted in June 2008. Most recently in 2013, an updated version of the MISRA C Guidelines was published under the title "MISRA C:2012 Guidelines for the use of the C language in critical systems". There are therefore three versions of the guidelines and these are now referred to as MISRA C:1998 (MISRA C1), MISRA C:2004 (MISRA C2) and MISRA C:2012 (MISRA C3).

Although derived from MISRA C2, MISRA C3 contains some significant changes. The Guidelines are divided into Directives and Rules, and each is classified as either Advisory, Required, or Mandatory. Code that is compliant with MISRA C2 will not necessarily be compliant with MISRA C3 and code that is compliant with MISRA C3 will not necessarily be compliant with MISRA C2.

1.2 The MISRA C:2012 Compliance Module

This document provides an introduction to the MISRA C:2012 Compliance Module (M3CM). The M3CM module is designed to enforce compliance with the MISRA C:2012 programming standard in conjunction with QA·C.

QA·C is a software tool that performs static analysis on C source code. Invariably a process of configuration is required in order to introduce QA·C effectively to a particular software development environment. It is important to have QA·C configured so as to restrict the output of warning messages to those that are of particular concern. M3CM is one such configuration, it configures QA·C to enforce compliance with the MISRA C:2012 coding guidelines.

QA·C is able to detect most of the statically detectable conditions identified in the MISRA C:2012 guidelines (as well as many others). M3CM configures QA·C to identify issues which are specific to those guidelines, and provides a cross-reference between the standard QA·C warning message(s) and the corresponding MISRA C:2012 guideline. Also available is an HTML description of each guideline.

For some MISRA C:2012 coding standard guidelines which are not enforceable directly

by QA·C, additional checks are provided through a secondary-analysis program. These checks are programmed using the analysis output files (.met files) generated by the QA·C "primary analysis" process.

Some MISRA C:2012 rules, as noted in Section 6.6 of the guidelines, impose constraints on the complete system rather than the code in a single translation unit. These requirements are enforced by the Cross-Module Analysis capability of QA·C.

1.3 Installation

For details of installation please refer to the PRQA Framework Installation Notes.

Please check the PRQA Framework Release Notes for the version of M3CM that is compatible with the installed version of QA·C.

2 M3CM Components

The M3CM compliance module introduces a number of additional components to your QA·C installation which provide the following features:

- A subset of the QA·C messages appropriate to enforcement of the MISRA C:2012 guidelines.
- A new message structure reflecting the MISRA C:2012 guidelines.
- Secondary-analysis checks to enforce a number of the MISRA C:2012 guidelines.

2.1 Dataflow

Dataflow analysis is required for the enforcement of a number of MISRA guidelines. An appropriate configuration for dataflow analysis will depend on factors such as the hardware environment and the complexity of the code. These issues are discussed in the dataflow reference manual. There are no configuration requirements specific to M3CM enforcement.

3 Using M3CM

3.1 Directives and Rules

MISRA C:2012 defines 159 guidelines consisting of 16 directives and 143 rules. The guidelines are grouped into a number of sections, for example Compilation and build, Code design, Unused code, Types, Pointer type conversions.

The MISRA C:2012 document describes the distinction between rules and directives as follows:

"*A directive is a guideline for which it is not possible to provide the full description necessary to perform a check for compliance. Additional information, such as might be provided in design documents or requirements specifications, is required in order to be able to perform the check. Static analysis tools may be able to assist in checking compliance with directives but tools may place widely different interpretations on what constitutes a non-compliance.*"

"*A rule is a guideline for which a complete description of the requirement has been provided. It should be possible to check that source code complies with a rule without needing any other information. In particular, static analysis tools should be capable of checking compliance with rules subject to the limitations described in Section 6.5.*"

M3CM describes directives as "Not Statically Enforceable (NSE)". Enforcement is provided for some directives but this may not be sufficient by itself to ensure compliance in every situation. It should be noted that the percentage of enforcement indicated by the Rule Enforcement report excludes all directives even though partial enforcement is available for some of them.

3.2 Configuring M3CM for Conformance to C90 or C99

MISRA C:2012 rules are classified as being related to the C90 Language Standard, the C99 Language Standard, or both (as described in Appendix F). By default, M3CM is configured to produce messages for systems conforming to the C90 standard and in this configuration, diagnostic messages will be triggered wherever a C99 feature is used.

If M3CM is to be used in a system conforming to C99 (or a subset of the additional features in C99), a number of messages may be suppressed.

- QA·C 8.1 has 19 messages which identify the use of C99 features: 0180, 0320, 0604, 0617, 0618, 0850, 0930, 0945, 1011, 1018, 1027, 1030, 1031, 1051, 1052, 1053, 1054, 1055, 1056. All these messages are prefixed with "[C99]":
- QA·C 8.1 has 19 messages which identify a breach of a C90 "environmental limit":

0410, 0609, 0611, 0612, 0614, 0639, 0647, 0715, 0739, 0776, 0778, 0810, 0815, 0816, 0828, 0857, 0858, 0859, 0875. These are messages which identify potential limits on portability which apply in C90 but not in a C99 environment. All these messages contain the text "ISO:C90".

3.3 The QA·C Message System

Each MISRA C Guideline is classified as either Mandatory, Required or Advisory and this classification is reflected in the structure of the message system. QA·C messages are arranged in 10 levels. In M3CM the following levels are used.

Level 0: **Information**
Level 4: **MISRA Advisory Directives**
Level 5: **MISRA Advisory Rules**
Level 6: **MISRA Required Directives**
Level 7: **MISRA Required Rules**
Level 8: **MISRA Mandatory Rules**

Level 0 contains informational messages, sub-messages and error recovery messages.

Levels 4 - 8 contain message groups corresponding to each MISRA C:2012 guideline, divided into advisory, required and mandatory rules and directives. This separation allows users to disable, for example, all advisory items in order to focus only on required and mandatory guidelines.

A rule or directive may be enforced by

- QA·C (primary analysis)
- Dataflow analysis
- Secondary Analysis
- CMA (Cross-Module Analysis)
- by some combination of these 4 elements.

The type of analysis used to enforce a given guideline is listed in the Cross Reference Listing.

3.4 Mapping Messages to MISRA Directives and Rules

Some analysis messages are associated with more than one MISRA Guideline. This occurs when:

- The requirements of 2 MISRA guidelines overlap. For example, any violation of Rule 11.7 ("A cast shall not be performed between pointer to object and a non-integer arithmetic type") will also violate Rule 1.3, or
- The granularity of the condition identified by an analysis message does not coincide uniquely with a single MISRA Guideline. For example, message 2771 ("Definite: These pointers address different objects.") is mapped to both Rule 18.2 ("Subtraction between pointers shall only be applied to pointers that address elements of the same array") and Rule 18.3 ("The relational operators and >= shall not be applied to objects of pointer type except where they point into the same object"), or
- Violation of one rule may also result in violation of another. For example Rule 5.8 ("Identifiers that define objects or functions with external linkage shall be unique") and Rule 5.9 ("Identifiers that define objects or functions with internal linkage should be unique").

The guideline or guidelines with which each message is associated will always be reported in the "reference text". However, in the (few) situations where a message is associated with more than one guideline, it will only be specifically *mapped* to one of the guidelines - the *primary* guideline. Where this situation occurs, any statistics relating to the number of violations of any particular guideline may be slightly misleading.

It is also important to note that if a message which is mapped to more than one guideline is suppressed, it will be suppressed for all the guidelines with which it is associated.

4 Global Checks

Cross-Module Analysis checks are required to fully implement a compliance modules rule set. This analysis is normally only of significant value when the code from a complete project is available for analysis.

For details of configuration and use of Cross-Module Analysis, please refer to the RCMA Manual.

5 Cross Reference Listing

MISRA C:2012 Dir-1.1 (Required)
Any implementation-defined behaviour on which the output of the program depends shall be documented and understood
QAC Messages: 0180, 0202, 0284, 0285, 0286, 0287, 0288, 0289, 0292, 0299, 0320, 0371, 0372, 0375, 0380, 0388, 0390, 0391, 0392, 0410, 0581, 0604, 0609, 0611, 0612, 0614, 0617, 0618, 0634, 0639, 0647, 0715, 0739, 0810, 0828, 0850, 0857, 0858, 0859, 0875, 0930, 1011, 1018, 1027, 1030, 1031, 1053, 1054, 1055, 1056, 2855, 2856, 2857, 2860, 2861, 2862, 2890, 2891, 2892, 2895, 2896, 2897, 3116
Enforcement: NSE - Documentation requirements cannot be statically enforced

MISRA C:2012 Dir-2.1 (Required)
All source files shall compile without any compilation errors
Enforcement: NSE - Compilation process requirements cannot be statically enforced

MISRA C:2012 Dir-3.1 (Required)
All code shall be traceable to documented requirements
Enforcement: NSE - Traceability to documentation requirements cannot be statically enforced

MISRA C:2012 Dir-4.1 (Required)
Run-time failures shall be minimized
QAC Messages: 2791, 2792, 2801, 2802, 2811, 2812, 2821, 2822, 2831, 2832, 2841, 2842, 2845, 2846, 2847, 2871, 2872, 2877
Enforcement: NSE - The requirements are not well-defined and enforcement can never be comprehensive

MISRA C:2012 Dir-4.2 (Advisory)
All usage of assembly language should be documented
QAC Messages: 1003, 1006
Enforcement: NSE - Documentation requirements cannot be statically enforced

MISRA C:2012 Dir-4.3 (Required)
Assembly language shall be encapsulated and isolated
QAC Messages: 3006
Enforcement: NSE - Encapsulation and isolation are not well-defined requirements

MISRA C:2012 Dir-4.4 (Advisory)
Sections of code should not be “commented out”
Enforcement: NSE - The requirement to distinguish between “code” and “comments” is not well-defined

MISRA C:2012 Dir-4.5 (Advisory)
Identifiers in the same name space with overlapping visibility should be typographically

unambiguous
Enforcement: NSE - The requirements are only defined for English language environments

MISRA C:2012 Dir-4.6 (Advisory)
typedefs that indicate size and signedness should be used in place of the basic numerical types
QAC Messages: 5209
Enforcement: NSE - The ways in which size and signedness may be specified are not defined

MISRA C:2012 Dir-4.7 (Required)
If a function returns error information, then that error information shall be tested
Enforcement: NSE - The ways in which a function may return error information cannot be defined

MISRA C:2012 Dir-4.8 (Advisory)
If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden
Enforcement: NSE - Choosing to use an opaque pointer declaration may not be appropriate in all circumstances

MISRA C:2012 Dir-4.9 (Advisory)
A function should be used in preference to a function-like macro where they are interchangeable
QAC Messages: 3453
Enforcement: NSE - Choosing to use a function rather than a function-like macro is not appropriate in all circumstances

MISRA C:2012 Dir-4.10 (Required)
Precautions shall be taken in order to prevent the contents of a header file being included more then once
QAC Messages: 0883
Enforcement: NSE - The requirement may be met in a variety of ways

MISRA C:2012 Dir-4.11 (Required)
The validity of values passed to library functions shall be checked
Enforcement: NSE - The ways in which function arguments may be checked cannot be defined

MISRA C:2012 Dir-4.12 (Required)
Dynamic memory allocation shall not be used
Enforcement: NSE - The ways in which memory allocation may occur cannot be defined

MISRA C:2012 Dir-4.13 (Advisory)
Functions which are designed to provide operations on a resource should be called in an

appropriate sequence
Enforcement: NSE - The requirements are not well-defined

MISRA C:2012 Rule-1.1 (Required)
The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits
QAC Messages: 0232, 0233, 0244, 0261, 0321, 0322, 0338, 0422, 0423, 0426, 0427, 0429, 0430, 0431, 0432, 0435, 0436, 0437, 0446, 0447, 0448, 0449, 0450, 0451, 0452, 0453, 0454, 0456, 0457, 0458, 0466, 0467, 0468, 0469, 0476, 0477, 0478, 0481, 0482, 0483, 0484, 0485, 0486, 0487, 0493, 0494, 0495, 0496, 0513, 0514, 0515, 0536, 0537, 0540, 0541, 0542, 0546, 0547, 0550, 0554, 0555, 0556, 0557, 0558, 0559, 0560, 0561, 0562, 0563, 0564, 0565, 0580, 0588, 0589, 0590, 0591, 0605, 0616, 0619, 0620, 0621, 0622, 0627, 0628, 0629, 0631, 0638, 0640, 0641, 0642, 0643, 0644, 0645, 0646, 0649, 0650, 0651, 0653, 0655, 0656, 0657, 0659, 0664, 0665, 0671, 0673, 0674, 0675, 0677, 0682, 0683, 0684, 0685, 0690, 0698, 0699, 0708, 0709, 0736, 0737, 0738, 0746, 0747, 0755, 0756, 0757, 0758, 0766, 0767, 0768, 0774, 0775, 0801, 0802, 0803, 0804, 0811, 0812, 0821, 0834, 0835, 0844, 0845, 0851, 0852, 0856, 0866, 0873, 0877, 0940, 0941, 0943, 0944, 1023, 1024, 1025, 1033, 1047, 1048, 3236, 3237, 3238, 3244
Enforcement: QAC

MISRA C:2012 Rule-1.2 (Advisory)
Language extensions should not be used
QAC Messages: 0240, 0241, 0246, 0551, 0601, 0633, 0635, 0660, 0662, 0830, 0831, 0899, 1001, 1002, 1003, 1006, 1008, 1012, 1014, 1015, 1019, 1020, 1021, 1022, 1026, 1028, 1029, 1034, 1035, 1036, 1037, 1038, 1041, 1042, 1043, 1044, 1045, 1046, 3664
Enforcement: QAC

MISRA C:2012 Rule-1.3 (Required)
There shall be no occurrence of undefined or critical unspecified behaviour
QAC Messages: 0160, 0161, 0162, 0163, 0164, 0165, 0166, 0167, 0168, 0169, 0170, 0171, 0172, 0173, 0174, 0175, 0176, 0177, 0178, 0179, 0184, 0185, 0186, 0190, 0191, 0192, 0193, 0194, 0195, 0196, 0197, 0198, 0199, 0200, 0201, 0203, 0204, 0206, 0207, 0208, 0235, 0275, 0301, 0302, 0304, 0307, 0309, 0337, 0400, 0401, 0402, 0403, 0475, 0543, 0544, 0545, 0602, 0623, 0625, 0626, 0630, 0632, 0636, 0654, 0658, 0661, 0667, 0668, 0672, 0676, 0678, 0680, 0706, 0745, 0777, 0779, 0809, 0813, 0814, 0836, 0837, 0848, 0853, 0854, 0864, 0865, 0867, 0872, 0874, 0885, 0887, 0888, 0914, 0915, 0942, 1331, 1332, 1333, 1509, 1510, 2800, 2810, 2820, 2830, 2840, 3113, 3114, 3239, 3311, 3312, 3319, 3320, 3437, 3438
Enforcement: QAC, Dataflow and CMA

MISRA C:2012 Rule-2.1 (Required)
A project shall not contain unreachable code
QAC Messages: 0594, 1460, 1503, 2742, 2744, 2880, 2882, 3219
Enforcement: QAC, Dataflow and CMA

MISRA C:2012 Rule-2.2 (Required)
There shall be no dead code
QAC Messages: 2980, 2981, 2982, 2983, 2984, 2985, 2986, 2995, 2996, 3110, 3112, 3425, 3426, 3427
Enforcement: QAC and Dataflow

MISRA C:2012 Rule-2.3 (Advisory)
A project should not contain unused type declarations
Enforcement: Not Enforced

MISRA C:2012 Rule-2.4 (Advisory)
A project should not contain unused tag declarations
Enforcement: Not Enforced

MISRA C:2012 Rule-2.5 (Advisory)
A project should not contain unused macro declarations
Enforcement: Not Enforced

MISRA C:2012 Rule-2.6 (Advisory)
A function should not contain unused label declarations
QAC Messages: 3202
Enforcement: QAC

MISRA C:2012 Rule-2.7 (Advisory)
There should be no unused parameters in functions
QAC Messages: 3206
Enforcement: QAC

MISRA C:2012 Rule-3.1 (Required)
The character sequences /* and // shall not be used within a comment.
QAC Messages: 3108, 5133
Enforcement: QAC and Secondary-analysis

MISRA C:2012 Rule-3.2 (Required)
Line-splicing shall not be used in // comments.
QAC Messages: 5134
Enforcement: Secondary-analysis

MISRA C:2012 Rule-4.1 (Required)
Octal and hexadecimal escape sequences shall be terminated
Enforcement: Not Enforced

MISRA C:2012 Rule-4.2 (Advisory)
Trigraphs should not be used
QAC Messages: 3601
Enforcement: QAC

MISRA C:2012 Rule-5.1 (Required)
External identifiers shall be distinct
QAC Messages: 0777
Enforcement: QAC

MISRA C:2012 Rule-5.2 (Required)
Identifiers declared in the same scope and name space shall be distinct
QAC Messages: 0779
Enforcement: QAC

MISRA C:2012 Rule-5.3 (Required)
An identifier declared in an inner scope shall not hide an identifier declared in an outer scope
QAC Messages: 2547, 3334
Enforcement: QAC

MISRA C:2012 Rule-5.4 (Required)
Macro identifiers shall be distinct
Enforcement: Not Enforced

MISRA C:2012 Rule-5.5 (Required)
Identifiers shall be distinct from macro names
Enforcement: Not Enforced

MISRA C:2012 Rule-5.6 (Required)
A typedef name shall be a unique identifier
QAC Messages: 1506, 1507, 1508, 3448
Enforcement: QAC and CMA

MISRA C:2012 Rule-5.7 (Required)
A tag name shall be a unique identifier
Enforcement: Not Enforced

MISRA C:2012 Rule-5.8 (Required)
Identifiers that define objects or functions with external linkage shall be unique
QAC Messages: 1525, 1526
Enforcement: CMA

MISRA C:2012 Rule-5.9 (Advisory)
Identifiers that define objects or functions with internal linkage should be unique
QAC Messages: 1525, 1527, 1528
Enforcement: CMA

MISRA C:2012 Rule-6.1 (Required)
Bit-fields shall only be declared with an appropriate type
QAC Messages: 0634, 0635
Enforcement: QAC

MISRA C:2012 Rule-6.2 (Required)
Single-bit named bit fields shall not be of a signed type
QAC Messages: 3660, 3665
Enforcement: QAC

MISRA C:2012 Rule-7.1 (Required)
Octal constants shall not be used
QAC Messages: 0336, 0339, 3628
Enforcement: QAC

MISRA C:2012 Rule-7.2 (Required)
A “u” or “U” suffix shall be applied to all integer constants that are represented in an unsigned type
QAC Messages: 1281
Enforcement: QAC

MISRA C:2012 Rule-7.3 (Required)
The lowercase character “l” shall not be used in a literal suffix
QAC Messages: 1280
Enforcement: QAC

MISRA C:2012 Rule-7.4 (Required)
A string literal shall not be assigned to an object unless the object’s type is “pointer to const-qualified char”
QAC Messages: 0752, 0753
Enforcement: QAC

MISRA C:2012 Rule-8.1 (Required)
Types shall be explicitly specified
QAC Messages: 2050, 2051
Enforcement: QAC

MISRA C:2012 Rule-8.2 (Required)
Function types shall be in prototype form with named parameters
QAC Messages: 1335, 1336, 3001, 3002, 3007
Enforcement: QAC

MISRA C:2012 Rule-8.3 (Required)
All declarations of an object or function shall use the same names and type qualifiers
QAC Messages: 0624, 1330, 3675
Enforcement: QAC

MISRA C:2012 Rule-8.4 (Required)
A compatible declaration shall be visible when an object or function with external linkage is defined
QAC Messages: 3408

Enforcement: QAC

MISRA C:2012 Rule-8.5 (Required)
An external object or function shall be declared once in one and only one file
QAC Messages: 1513, 3221, 3222, 3447, 3451
Enforcement: QAC and CMA

MISRA C:2012 Rule-8.6 (Required)
An identifier with external linkage shall have exactly one external definition
QAC Messages: 0630, 1509, 3406
Enforcement: QAC and CMA

MISRA C:2012 Rule-8.7 (Advisory)
Functions and objects should not be defined with external linkage if they are referenced in only one translation unit
QAC Messages: 1504, 1505
Enforcement: CMA

MISRA C:2012 Rule-8.8 (Required)
The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage
QAC Messages: 3224
Enforcement: QAC

MISRA C:2012 Rule-8.9 (Advisory)
An object should be defined at block scope if its identifier only appears in a single function
QAC Messages: 1514, 3218
Enforcement: QAC and CMA

MISRA C:2012 Rule-8.10 (Required)
An inline function shall be declared with the static storage class
QAC Messages: 3240, 3243
Enforcement: QAC

MISRA C:2012 Rule-8.11 (Advisory)
When an array with external linkage is declared, its size should be explicitly specified
QAC Messages: 3684
Enforcement: QAC

MISRA C:2012 Rule-8.12 (Required)
Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique
QAC Messages: 0724
Enforcement: QAC

MISRA C:2012 Rule-8.13 (Advisory)
A pointer should point to a const-qualified type whenever possible

QAC Messages: 3673
Enforcement: QAC

MISRA C:2012 Rule-8.14 (Required)
The restrict type qualifier shall not be used
QAC Messages: 5137
Enforcement: QAC

MISRA C:2012 Rule-9.1 (Mandatory)
The value of an object with automatic storage duration shall not be read before it has been set
QAC Messages: 2883, 2961, 2962, 2971, 2972
Enforcement: Dataflow

MISRA C:2012 Rule-9.2 (Required)
The initializer for an aggregate or union shall be enclosed in braces
QAC Messages: 0693, 0694
Enforcement: QAC

MISRA C:2012 Rule-9.3 (Required)
Arrays shall not be partially initialized
QAC Messages: 0686
Enforcement: QAC

MISRA C:2012 Rule-9.4 (Required)
An element of an object shall not be initialized more than once
Enforcement: Not Enforced

MISRA C:2012 Rule-9.5 (Required)
Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly
Enforcement: Not Enforced

MISRA C:2012 Rule-10.1 (Required)
Operands shall not be of an inappropriate essential type.
QAC Messages: 3101, 3102, 4500, 4501, 4502, 4503, 4504, 4505, 4507, 4510, 4511, 4512, 4513, 4514, 4518, 4519, 4521, 4522, 4523, 4524, 4527, 4528, 4529, 4532, 4533, 4534, 4538, 4539, 4542, 4543, 4544, 4548, 4549, 4558, 4559, 4568, 4569
Enforcement: QAC

MISRA C:2012 Rule-10.2 (Required)
Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations
QAC Messages: 1810, 1811, 1812, 1813
Enforcement: QAC

MISRA C:2012 Rule-10.3 (Required)

The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category.
QAC Messages: 0570, 0572, 1257, 1264, 1265, 1266, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 2850, 2851, 2852, 2900, 2901, 2902, 4401, 4402, 4403, 4404, 4405, 4410, 4412, 4413, 4414, 4415, 4420, 4421, 4422, 4423, 4424, 4425, 4430, 4431, 4432, 4434, 4435, 4436, 4437, 4440, 4441, 4442, 4443, 4445, 4446, 4447, 4450, 4451, 4452, 4453, 4454, 4460, 4461, 4462, 4463, 4464, 4465
Enforcement: QAC and Dataflow

MISRA C:2012 Rule-10.4 (Required)
Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category
QAC Messages: 1800, 1802, 1803, 1804, 1820, 1821, 1822, 1823, 1824, 1830, 1831, 1832, 1833, 1834, 1840, 1841, 1842, 1843, 1844, 1850, 1851, 1852, 1853, 1854, 1860, 1861, 1862, 1863, 1864, 1880, 1881, 1882
Enforcement: QAC

MISRA C:2012 Rule-10.5 (Advisory)
The value of an expression should not be cast to an inappropriate essential type
QAC Messages: 4301, 4302, 4303, 4304, 4305, 4310, 4312, 4315, 4320, 4322, 4330, 4332, 4340, 4342, 4350, 4351, 4352
Enforcement: QAC

MISRA C:2012 Rule-10.6 (Required)
The value of a composite expression shall not be assigned to an object with wider essential type
QAC Messages: 4490, 4491, 4492, 4499
Enforcement: QAC

MISRA C:2012 Rule-10.7 (Required)
If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type
QAC Messages: 1890, 1891, 1892, 1893, 1894, 1895
Enforcement: QAC

MISRA C:2012 Rule-10.8 (Required)
The value of a composite expression shall not be cast to a different essential type category or a wider essential type
QAC Messages: 4390, 4391, 4392, 4393, 4394, 4395, 4398, 4399
Enforcement: QAC

MISRA C:2012 Rule-11.1 (Required)
Conversions shall not be performed between a pointer to a function and any other type
QAC Messages: 0302, 0305, 0307, 0313

Enforcement: QAC

MISRA C:2012 Rule-11.2 (Required)
Conversions shall not be performed between a pointer to an incomplete type and any other type
QAC Messages: 0308
Enforcement: QAC

MISRA C:2012 Rule-11.3 (Required)
A cast shall not be performed between a pointer to object type and a pointer to a different object type
QAC Messages: 0310, 3305
Enforcement: QAC

MISRA C:2012 Rule-11.4 (Advisory)
A conversion should not be performed between a pointer to object and an integer type
QAC Messages: 0303, 0306, 0360, 0361, 0362
Enforcement: QAC

MISRA C:2012 Rule-11.5 (Advisory)
A conversion should not be performed from pointer to void into pointer to object
QAC Messages: 0316, 0317
Enforcement: QAC

MISRA C:2012 Rule-11.6 (Required)
A cast shall not be performed between pointer to void and an arithmetic type
QAC Messages: 0306
Enforcement: QAC

MISRA C:2012 Rule-11.7 (Required)
A cast shall not be performed between pointer to object and a non-integer arithmetic type
QAC Messages: 0301
Enforcement: QAC

MISRA C:2012 Rule-11.8 (Required)
A cast shall not remove any const or volatile qualification from the type pointed to by a pointer
QAC Messages: 0311, 0312
Enforcement: QAC

MISRA C:2012 Rule-11.9 (Required)
The macro NULL shall be the only permitted form of integer null pointer constant
QAC Messages: 3003, 3004
Enforcement: QAC

MISRA C:2012 Rule-12.1 (Advisory)
The precedence of operators within expressions should be made explicit

QAC Messages: 3389, 3391, 3392, 3394, 3395, 3396, 3397
Enforcement: QAC

MISRA C:2012 Rule-12.2 (Required)
The right hand operand of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand
QAC Messages: 0499, 2790
Enforcement: QAC and Dataflow

MISRA C:2012 Rule-12.3 (Advisory)
The comma operator should not be used
QAC Messages: 3417, 3418
Enforcement: QAC

MISRA C:2012 Rule-12.4 (Advisory)
Evaluation of constant expressions should not lead to unsigned integer wrap-around
QAC Messages: 2910
Enforcement: Dataflow

MISRA C:2012 Rule-13.1 (Required)
Initializer lists shall not contain persistent side-effects
Enforcement: Not Enforced

MISRA C:2012 Rule-13.2 (Required)
The value of an expression and its persistent side-effects shall be the same under all permitted evaluation orders
QAC Messages: 0400, 0401, 0402, 0403
Enforcement: QAC

MISRA C:2012 Rule-13.3 (Advisory)
A full expression containing an increment (++) or decrement (–) operator should have no other potential side effects other than that caused by the increment or decrement operator
QAC Messages: 3440
Enforcement: QAC

MISRA C:2012 Rule-13.4 (Advisory)
The result of an assignment operator should not be used
QAC Messages: 3226, 3326
Enforcement: QAC

MISRA C:2012 Rule-13.5 (Required)
The right hand operand of a logical && or || operator shall not contain persistent side effects
QAC Messages: 3415
Enforcement: QAC

MISRA C:2012 Rule-13.6 (Mandatory)

The operand of the sizeof operator shall not contain any expression which has potential side-effects
QAC Messages: 3307
Enforcement: QAC

MISRA C:2012 Rule-14.1 (Required)
A loop counter shall not have essentially floating type
QAC Messages: 3340, 3342
Enforcement: QAC

MISRA C:2012 Rule-14.2 (Required)
A for loop shall be well-formed
QAC Messages: 2461, 2462, 2463, 2464, 2467, 2469, 2471, 2472
Enforcement: QAC

MISRA C:2012 Rule-14.3 (Required)
Controlling expressions shall not be invariant
QAC Messages: 2990, 2991, 2992, 2993, 2994
Enforcement: Dataflow

MISRA C:2012 Rule-14.4 (Required)
The controlling expression of an if-statement and the controlling expression of an iteration-statement shall have essentially Boolean type
QAC Messages: 3344
Enforcement: QAC

MISRA C:2012 Rule-15.1 (Advisory)
The goto statement should not be used
QAC Messages: 2001
Enforcement: QAC

MISRA C:2012 Rule-15.2 (Required)
The goto statement shall jump to a label declared later in the same function
Enforcement: Not Enforced

MISRA C:2012 Rule-15.3 (Required)
Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement
QAC Messages: 3311
Enforcement: QAC

MISRA C:2012 Rule-15.4 (Advisory)
There should be no more than one break or goto statement used to terminate any iteration statement
QAC Messages: 0771
Enforcement: QAC

MISRA C:2012 Rule-15.5 (Advisory)
A function should have a single point of exit at the end
QAC Messages: 2889
Enforcement: Dataflow

MISRA C:2012 Rule-15.6 (Required)
The body of an iteration-statement or a selection-statement shall be a compound-statement
QAC Messages: 2212, 2214, 3402
Enforcement: QAC

MISRA C:2012 Rule-15.7 (Required)
All if ... else if constructs shall be terminated with an else statement
QAC Messages: 2004
Enforcement: QAC

MISRA C:2012 Rule-16.1 (Required)
All switch statements shall be well-formed
QAC Messages: 2008, 3234
Enforcement: QAC

MISRA C:2012 Rule-16.2 (Required)
A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement
QAC Messages: 2019
Enforcement: QAC

MISRA C:2012 Rule-16.3 (Required)
An unconditional break statement shall terminate every switch-clause
QAC Messages: 2003, 2020
Enforcement: QAC

MISRA C:2012 Rule-16.4 (Required)
Every switch statement shall have a default label
QAC Messages: 2002
Enforcement: QAC

MISRA C:2012 Rule-16.5 (Required)
A default label shall appear as either the first or the last switch label of a switch statement
QAC Messages: 2009
Enforcement: QAC

MISRA C:2012 Rule-16.6 (Required)
Every switch statement shall have at least two switch-clauses
QAC Messages: 3315
Enforcement: QAC

MISRA C:2012 Rule-16.7 (Required)

A switch-expression shall not have essentially Boolean type
QAC Messages: 0735
Enforcement: QAC

MISRA C:2012 Rule-17.1 (Required)
The features of <stdarg.h> shall not be used
QAC Messages: 1337, 5130
Enforcement: QAC and Secondary-analysis.

MISRA C:2012 Rule-17.2 (Required)
Functions shall not call themselves, either directly or indirectly
QAC Messages: 1520, 3670
Enforcement: QAC and CMA

MISRA C:2012 Rule-17.3 (Mandatory)
A function shall not be declared implicitly
QAC Messages: 3335
Enforcement: QAC

MISRA C:2012 Rule-17.4 (Mandatory)
All exit paths from a function with non-void return type shall have an explicit return statement with an expression
QAC Messages: 0745, 2887, 2888, 3113, 3114
Enforcement: QAC and Dataflow

MISRA C:2012 Rule-17.5 (Advisory)
The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements
Enforcement: Not Enforced

MISRA C:2012 Rule-17.6 (Mandatory)
The declaration of an array parameter shall not contain the static keyword between the [ ]
Enforcement: Not Enforced

MISRA C:2012 Rule-17.7 (Required)
The value returned by a function having non-void return type shall be used
QAC Messages: 3200
Enforcement: QAC

MISRA C:2012 Rule-17.8 (Advisory)
A function parameter should not be modified
Enforcement: Not Enforced

MISRA C:2012 Rule-18.1 (Required)
A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand
QAC Messages: 2930, 2931, 2932

Enforcement: Dataflow

MISRA C:2012 Rule-18.2 (Required)
Subtraction between pointers shall only be applied to pointers that address elements of the same array
QAC Messages: 2771, 2772
Enforcement: Dataflow

MISRA C:2012 Rule-18.3 (Required)
The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object
QAC Messages: 2771, 2772
Enforcement: Dataflow

MISRA C:2012 Rule-18.4 (Advisory)
The +, -, += and -= operators should not be applied to an expression of pointer type
QAC Messages: 0488
Enforcement: QAC

MISRA C:2012 Rule-18.5 (Advisory)
Declarations should contain no more than two levels of pointer nesting
QAC Messages: 3260, 3261, 3262, 3263
Enforcement: QAC

MISRA C:2012 Rule-18.6 (Required)
The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist
QAC Messages: 3217, 3225, 3230, 4140
Enforcement: QAC

MISRA C:2012 Rule-18.7 (Required)
Flexible array members shall not be declared
Enforcement: Not Enforced

MISRA C:2012 Rule-18.8 (Required)
Variable-length array types shall not be used
QAC Messages: 0945, 1051, 1052
Enforcement: QAC

MISRA C:2012 Rule-19.1 (Mandatory)
An object shall not be assigned or copied to an overlapping object
QAC Messages: 2776, 2777
Enforcement: Dataflow

MISRA C:2012 Rule-19.2 (Advisory)
The union keyword should not be used
QAC Messages: 0750, 0759

Enforcement: QAC

MISRA C:2012 Rule-20.1 (Advisory)
#include directives should only be preceded by preprocessor directives or comments
QAC Messages: 5087
Enforcement: Secondary-analysis

MISRA C:2012 Rule-20.2 (Required)
The ', " or \ characters and the /* or // character sequences shall not occur in a header file name
QAC Messages: 0813, 0814, 0831
Enforcement: QAC

MISRA C:2012 Rule-20.3 (Required)
The #include directive shall be followed by either a <filename> or "filename" sequence
QAC Messages: 0809
Enforcement: QAC

MISRA C:2012 Rule-20.4 (Required)
A macro shall not be defined with the same name as a keyword
QAC Messages: 3439
Enforcement: QAC

MISRA C:2012 Rule-20.5 (Advisory)
#undef should not be used
QAC Messages: 0841
Enforcement: QAC

MISRA C:2012 Rule-20.6 (Required)
Tokens that look like a preprocessing directive shall not occur within a macro argument
QAC Messages: 0853
Enforcement: QAC

MISRA C:2012 Rule-20.7 (Required)
Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses
QAC Messages: 3410
Enforcement: QAC

MISRA C:2012 Rule-20.8 (Required)
The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1
Enforcement: Not Enforced

MISRA C:2012 Rule-20.9 (Required)
All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation
QAC Messages: 3332

Enforcement: QAC

MISRA C:2012 Rule-20.10 (Advisory)
The # and ## preprocessor operators should not be used
QAC Messages: 0341, 0342
Enforcement: QAC

MISRA C:2012 Rule-20.11 (Required)
A macro parameter immediately following a # operator shall not immediately be followed by a ## operator
Enforcement: Not Enforced

MISRA C:2012 Rule-20.12 (Required)
A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators
Enforcement: Not Enforced

MISRA C:2012 Rule-20.13 (Required)
A line whose first token is # shall be a valid preprocessing directive
QAC Messages: 3115
Enforcement: QAC

MISRA C:2012 Rule-20.14 (Required)
All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related
QAC Messages: 3317, 3318
Enforcement: QAC

MISRA C:2012 Rule-21.1 (Required)
#define and #undef shall not be used on a reserved identifier or reserved macro name
QAC Messages: 0836, 0848, 0854, 4600, 4601
Enforcement: QAC

MISRA C:2012 Rule-21.2 (Required)
A reserved identifier or macro name shall not be declared
QAC Messages: 0602, 4602, 4603, 4604, 4605, 4606, 4607, 4608
Enforcement: QAC

MISRA C:2012 Rule-21.3 (Required)
The memory allocation and deallocation functions of <stdlib.h> shall not be used
QAC Messages: 5118
Enforcement: QAC

MISRA C:2012 Rule-21.4 (Required)
The standard header file <setjmp.h> shall not be used
QAC Messages: 5132
Enforcement: Secondary-analysis.

MISRA C:2012 Rule-21.5 (Required)
The standard header file <signal.h> shall not be used
QAC Messages: 5123
Enforcement: Secondary-analysis.

MISRA C:2012 Rule-21.6 (Required)
The Standard Library input/output functions shall not be used
QAC Messages: 5124
Enforcement: Secondary-analysis.

MISRA C:2012 Rule-21.7 (Required)
The atof, atoi, atol and atoll functions of <stdlib.h> shall not be used
QAC Messages: 5125
Enforcement: QAC

MISRA C:2012 Rule-21.8 (Required)
The library functions abort, exit, getenv and system of <stdlib.h> shall not be used
QAC Messages: 5126
Enforcement: QAC

MISRA C:2012 Rule-21.9 (Required)
The library functions bsearch and qsort of <stdlib.h> shall not be used
QAC Messages: 5135
Enforcement: QAC

MISRA C:2012 Rule-21.10 (Required)
The Standard Library time and date functions shall not be used
QAC Messages: 5127
Enforcement: Secondary-analysis

MISRA C:2012 Rule-21.11 (Required)
The standard header file <tgmath.h> shall not be used
QAC Messages: 5131
Enforcement: Secondary-analysis

MISRA C:2012 Rule-21.12 (Advisory)
The exception handling features of <fenv.h> should not be used
QAC Messages: 5136
Enforcement: QAC

MISRA C:2012 Rule-22.1 (Required)
All resources obtained dynamically by means of Standard Library functions shall be explicitly released
Enforcement: Not Enforced

MISRA C:2012 Rule-22.2 (Mandatory)
A block of memory shall only be freed if it was allocated by means of a Standard Library

function
Enforcement: Not Enforced

MISRA C:2012 Rule-22.3 (Required)
The same file shall not be open for read and write access at the same time on different streams
Enforcement: Not Enforced

MISRA C:2012 Rule-22.4 (Mandatory)
There shall be no attempt to write to a stream which has been opened as read-only
Enforcement: Not Enforced

MISRA C:2012 Rule-22.5 (Mandatory)
A pointer to a FILE object shall not be dereferenced
Enforcement: Not Enforced

MISRA C:2012 Rule-22.6 (Mandatory)
The value of a pointer to a FILE shall not be used after the associated stream has been closed
Enforcement: Not Enforced

6 Rule Enforcement

This report lists all rules, grouping by "Advisory" or "Mandatory" or "Required" and whether enforced.

6.1 Enforced Rules - Advisory

Table 6.1: M3CM Enforcement Rules - Advisory

Rule Id	Rule	Enforcement
Rule-1.2	Language extensions should not be used	0240, 0241, 0246, 0551, 0601, 0633, 0635, 0660, 0662, 0830, 0831, 0899, 1001, 1002, 1003, 1006, 1008, 1012, 1014, 1015, 1019, 1020, 1021, 1022, 1026, 1028, 1029, 1034, 1035, 1036, 1037, 1038, 1041, 1042, 1043, 1044, 1045, 1046, 3664
Rule-2.6	A function should not contain unused label declarations	3202
Rule-2.7	There should be no unused parameters in functions	3206
Rule-4.2	Trigraphs should not be used	3601
Rule-5.9	Identifiers that define objects or functions with internal linkage should be unique	1525, 1527, 1528
Rule-8.7	Functions and objects should not be defined with external linkage if they are referenced in only one translation unit	1504, 1505
Rule-8.9	An object should be defined at block scope if its identifier only appears in a single function	1514, 3218
Rule-8.11	When an array with external linkage is declared, its size should be explicitly specified	3684
Rule-8.13	A pointer should point to a const-qualified type whenever possible	3673
Rule-10.5	The value of an expression should not be cast to an inappropriate essential type	4301, 4302, 4303, 4304, 4305, 4310, 4312, 4315, 4320, 4322, 4330, 4332, 4340, 4342, 4350, 4351, 4352

continued on next page

Table 6.1 – continued from previous page

Rule Id	Rule	Enforcement
Rule-11.4	A conversion should not be performed between a pointer to object and an integer type	0303, 0306, 0360, 0361, 0362
Rule-11.5	A conversion should not be performed from pointer to void into pointer to object	0316, 0317
Rule-12.1	The precedence of operators within expressions should be made explicit	3389, 3391, 3392, 3394, 3395, 3396, 3397
Rule-12.3	The comma operator should not be used	3417, 3418
Rule-12.4	Evaluation of constant expressions should not lead to unsigned integer wrap-around	2910
Rule-13.3	A full expression containing an increment (++) or decrement (–) operator should have no other potential side effects other than that caused by the increment or decrement operator	3440
Rule-13.4	The result of an assignment operator should not be used	3226, 3326
Rule-15.1	The goto statement should not be used	2001
Rule-15.4	There should be no more than one break or goto statement used to terminate any iteration statement	0771
Rule-15.5	A function should have a single point of exit at the end	2889
Rule-18.4	The +, -, += and -= operators should not be applied to an expression of pointer type	0488
Rule-18.5	Declarations should contain no more than two levels of pointer nesting	3260, 3261, 3262, 3263
Rule-19.2	The union keyword should not be used	0750, 0759
Rule-20.1	#include directives should only be preceded by preprocessor directives or comments	5087
Rule-20.5	#undef should not be used	0841
Rule-20.10	The # and ## preprocessor operators should not be used	0341, 0342
Rule-21.12	The exception handling features of <fenv.h> should not be used	5136

Count of Enforced Advisory Rules = 27

6.2 Unenforced Rules - Advisory

Table 6.2: M3CM Unenforced Rules - Advisory

Rule Id	Rule
Rule-2.3	A project should not contain unused type declarations

continued on next page

Table 6.2 – continued from previous page

Rule Id	Rule
Rule-2.4	A project should not contain unused tag declarations
Rule-2.5	A project should not contain unused macro declarations
Rule-17.5	The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements
Rule-17.8	A function parameter should not be modified

Count of Unenforced Advisory Rules = 5

6.3 Not Statically Enforceable Rules - Advisory

Table 6.3: M3CM Not Statically Enforceable Rules - Advisory

Rule Id	Rule
Dir-4.2	All usage of assembly language should be documented
Dir-4.4	Sections of code should not be ”commented out”
Dir-4.5	Identifiers in the same name space with overlapping visibility should be typographically unambiguous
Dir-4.6	typedefs that indicate size and signedness should be used in place of the basic numerical types
Dir-4.8	If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden
Dir-4.9	A function should be used in preference to a function-like macro where they are interchangeable
Dir-4.13	Functions which are designed to provide operations on a resource should be called in an appropriate sequence

Count of Not Statically Enforceable Advisory Rules = 7

6.4 Enforced Rules - Mandatory

Table 6.4: M3CM Enforced Rules - Mandatory

Rule Id	Rule	Enforcement
Rule-9.1	The value of an object with automatic storage duration shall not be read before it has been set	2883, 2961, 2962, 2971, 2972
Rule-13.6	The operand of the sizeof operator shall not contain any expression which has potential side-effects	3307

continued on next page

Table 6.4 – continued from previous page

Rule Id	Rule	Enforcement
Rule-17.3	A function shall not be declared implicitly	3335
Rule-17.4	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	0745, 2887, 2888, 3113, 3114
Rule-19.1	An object shall not be assigned or copied to an overlapping object	2776, 2777

Count of Enforced Mandatory Rules = 5

6.5 Unenforced Rules - Mandatory

Table 6.5: M3CM Unenforced Rules - Mandatory

Rule Id	Rule
Rule-17.6	The declaration of an array parameter shall not contain the static keyword between the [ ]
Rule-22.2	A block of memory shall only be freed if it was allocated by means of a Standard Library function
Rule-22.4	There shall be no attempt to write to a stream which has been opened as read-only
Rule-22.5	A pointer to a FILE object shall not be dereferenced
Rule-22.6	The value of a pointer to a FILE shall not be used after the associated stream has been closed

Count of Unenforced Mandatory Rules = 5

6.6 Enforced Rules - Required

Table 6.6: M3CM Enforced Rules - Required

Rule Id	Rule	Enforcement
Rule-1.1	The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits	0232, 0233, 0244, 0261, 0321, 0322, 0338, 0422, 0423, 0426, 0427, 0429, 0430, 0431, 0432, 0435, 0436, 0437, 0446, 0447, 0448, 0449, 0450, 0451, 0452, 0453, 0454, 0456, 0457, 0458, 0466, 0467, 0468, 0469, 0476, 0477, 0478, 0481, 0482, 0483, 0484, 0485, 0486, 0487, 0493, 0494, 0495, 0496
Rule-1.1	Continued...	0513, 0514, 0515, 0536, 0537, 0540, 0541, 0542, 0546, 0547, 0550, 0554, 0555, 0556, 0557, 0558, 0559, 0560, 0561, 0562, 0563, 0564, 0565, 0580, 0588, 0589, 0590, 0591, 0605, 0616, 0619, 0620, 0621, 0622, 0627, 0628, 0629, 0631, 0638, 0640, 0641, 0642, 0643, 0644, 0645, 0646, 0649, 0650

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-1.1	Continued...	0651, 0653, 0655, 0656, 0657, 0659, 0664, 0665, 0671, 0673, 0674, 0675, 0677, 0682, 0683, 0684, 0685, 0690, 0698, 0699, 0708, 0709, 0736, 0737, 0738, 0746, 0747, 0755, 0756, 0757, 0758, 0766, 0767, 0768, 0774, 0775, 0801, 0802, 0803, 0804, 0811, 0812, 0821, 0834, 0835, 0844, 0845, 0851
Rule-1.1	Continued...	0852, 0856, 0866, 0873, 0877, 0940, 0941, 0943, 0944, 1023, 1024, 1025, 1033, 1047, 1048, 3236, 3237, 3238, 3244
Rule-1.3	There shall be no occurrence of undefined or critical unspecified behaviour	0160, 0161, 0162, 0163, 0164, 0165, 0166, 0167, 0168, 0169, 0170, 0171, 0172, 0173, 0174, 0175, 0176, 0177, 0178, 0179, 0184, 0185, 0186, 0190, 0191, 0192, 0193, 0194, 0195, 0196, 0197, 0198, 0199, 0200, 0201, 0203, 0204, 0206, 0207, 0208, 0235, 0275, 0301, 0302, 0304, 0307, 0309, 0337

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-1.3	Continued...	0400, 0401, 0402, 0403, 0475, 0543, 0544, 0545, 0602, 0623, 0625, 0626, 0630, 0632, 0636, 0654, 0658, 0661, 0667, 0668, 0672, 0676, 0678, 0680, 0706, 0745, 0777, 0779, 0809, 0813, 0814, 0836, 0837, 0848, 0853, 0854, 0864, 0865, 0867, 0872, 0874, 0885, 0887, 0888, 0914, 0915, 0942, 1331
Rule-1.3	Continued...	1332, 1333, 1509, 1510, 2800, 2810, 2820, 2830, 2840, 3113, 3114, 3239, 3311, 3312, 3319, 3320, 3437, 3438
Rule-2.1	A project shall not contain unreachable code	0594, 1460, 1503, 2742, 2744, 2880, 2882, 3219
Rule-2.2	There shall be no dead code	2980, 2981, 2982, 2983, 2984, 2985, 2986, 2995, 2996, 3110, 3112, 3425, 3426, 3427
Rule-3.1	The character sequences /* and // shall not be used within a comment.	3108, 5133
Rule-3.2	Line-splicing shall not be used in // comments.	5134
Rule-5.1	External identifiers shall be distinct	0777
Rule-5.2	Identifiers declared in the same scope and name space shall be distinct	0779
Rule-5.3	An identifier declared in an inner scope shall not hide an identifier declared in an outer scope	2547, 3334
Rule-5.6	A typedef name shall be a unique identifier	1506, 1507, 1508, 3448
Rule-5.8	Identifiers that define objects or functions with external linkage shall be unique	1525, 1526

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-6.1	Bit-fields shall only be declared with an appropriate type	0634, 0635
Rule-6.2	Single-bit named bit fields shall not be of a signed type	3660, 3665
Rule-7.1	Octal constants shall not be used	0336, 0339, 3628
Rule-7.2	A “u” or “U” suffix shall be applied to all integer constants that are represented in an unsigned type	1281
Rule-7.3	The lowercase character “l” shall not be used in a literal suffix	1280
Rule-7.4	A string literal shall not be assigned to an object unless the object’s type is “pointer to const-qualified char”	0752, 0753
Rule-8.1	Types shall be explicitly specified	2050, 2051
Rule-8.2	Function types shall be in prototype form with named parameters	1335, 1336, 3001, 3002, 3007
Rule-8.3	All declarations of an object or function shall use the same names and type qualifiers	0624, 1330, 3675
Rule-8.4	A compatible declaration shall be visible when an object or function with external linkage is defined	3408
Rule-8.5	An external object or function shall be declared once in one and only one file	1513, 3221, 3222, 3447, 3451
Rule-8.6	An identifier with external linkage shall have exactly one external definition	0630, 1509, 3406
Rule-8.8	The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage	3224
Rule-8.10	An inline function shall be declared with the static storage class	3240, 3243
Rule-8.12	Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	0724
Rule-8.14	The restrict type qualifier shall not be used	5137
Rule-9.2	The initializer for an aggregate or union shall be enclosed in braces	0693, 0694
Rule-9.3	Arrays shall not be partially initialized	0686

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-10.1	Operands shall not be of an inappropriate essential type.	3101, 3102, 4500, 4501, 4502, 4503, 4504, 4505, 4507, 4510, 4511, 4512, 4513, 4514, 4518, 4519, 4521, 4522, 4523, 4524, 4527, 4528, 4529, 4532, 4533, 4534, 4538, 4539, 4542, 4543, 4544, 4548, 4549, 4558, 4559, 4568, 4569
Rule-10.2	Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations	1810, 1811, 1812, 1813
Rule-10.3	The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category.	0570, 0572, 1257, 1264, 1265, 1266, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 2850, 2851, 2852, 2900, 2901, 2902, 4401, 4402, 4403, 4404, 4405, 4410, 4412, 4413, 4414, 4415, 4420, 4421, 4422, 4423, 4424, 4425, 4430, 4431, 4432, 4434, 4435, 4436, 4437, 4440, 4441, 4442, 4443
Rule-10.3	Continued...	4445, 4446, 4447, 4450, 4451, 4452, 4453, 4454, 4460, 4461, 4462, 4463, 4464, 4465

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-10.4	Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category	1800, 1802, 1803, 1804, 1820, 1821, 1822, 1823, 1824, 1830, 1831, 1832, 1833, 1834, 1840, 1841, 1842, 1843, 1844, 1850, 1851, 1852, 1853, 1854, 1860, 1861, 1862, 1863, 1864, 1880, 1881, 1882
Rule-10.6	The value of a composite expression shall not be assigned to an object with wider essential type	4490, 4491, 4492, 4499
Rule-10.7	If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type	1890, 1891, 1892, 1893, 1894, 1895
Rule-10.8	The value of a composite expression shall not be cast to a different essential type category or a wider essential type	4390, 4391, 4392, 4393, 4394, 4395, 4398, 4399
Rule-11.1	Conversions shall not be performed between a pointer to a function and any other type	0302, 0305, 0307, 0313
Rule-11.2	Conversions shall not be performed between a pointer to an incomplete type and any other type	0308
Rule-11.3	A cast shall not be performed between a pointer to object type and a pointer to a different object type	0310, 3305
Rule-11.6	A cast shall not be performed between pointer to void and an arithmetic type	0306
Rule-11.7	A cast shall not be performed between pointer to object and a non-integer arithmetic type	0301
Rule-11.8	A cast shall not remove any const or volatile qualification from the type pointed to by a pointer	0311, 0312
Rule-11.9	The macro NULL shall be the only permitted form of integer null pointer constant	3003, 3004
Rule-12.2	The right hand operand of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand	0499, 2790
Rule-13.2	The value of an expression and its persistent side-effects shall be the same under all permitted evaluation orders	0400, 0401, 0402, 0403
Rule-13.5	The right hand operand of a logical && or || operator shall not contain persistent side effects	3415
Rule-14.1	A loop counter shall not have essentially floating type	3340, 3342

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-14.2	A for loop shall be well-formed	2461, 2462, 2463, 2464, 2467, 2469, 2471, 2472
Rule-14.3	Controlling expressions shall not be invariant	2990, 2991, 2992, 2993, 2994
Rule-14.4	The controlling expression of an if-statement and the controlling expression of an iteration-statement shall have essentially Boolean type	3344
Rule-15.3	Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement	3311
Rule-15.6	The body of an iteration-statement or a selection-statement shall be a compound-statement	2212, 2214, 3402
Rule-15.7	All if ... else if constructs shall be terminated with an else statement	2004
Rule-16.1	All switch statements shall be well-formed	2008, 3234
Rule-16.2	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	2019
Rule-16.3	An unconditional break statement shall terminate every switch-clause	2003, 2020
Rule-16.4	Every switch statement shall have a default label	2002
Rule-16.5	A default label shall appear as either the first or the last switch label of a switch statement	2009
Rule-16.6	Every switch statement shall have at least two switch-clauses	3315
Rule-16.7	A switch-expression shall not have essentially Boolean type	0735
Rule-17.1	The features of <stdarg.h> shall not be used	1337, 5130
Rule-17.2	Functions shall not call themselves, either directly or indirectly	1520, 3670
Rule-17.7	The value returned by a function having non-void return type shall be used	3200
Rule-18.1	A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	2930, 2931, 2932
Rule-18.2	Subtraction between pointers shall only be applied to pointers that address elements of the same array	2771, 2772
Rule-18.3	The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object	2771, 2772

continued on next page

Table 6.6 – continued from previous page

Rule Id	Rule	Enforcement
Rule-18.6	The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist	3217, 3225, 3230, 4140
Rule-18.8	Variable-length array types shall not be used	0945, 1051, 1052
Rule-20.2	The ', " or \ characters and the /* or // character sequences shall not occur in a header file name	0813, 0814, 0831
Rule-20.3	The #include directive shall be followed by either a <filename> or "filename" sequence	0809
Rule-20.4	A macro shall not be defined with the same name as a keyword	3439
Rule-20.6	Tokens that look like a preprocessing directive shall not occur within a macro argument	0853
Rule-20.7	Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses	3410
Rule-20.9	All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation	3332
Rule-20.13	A line whose first token is # shall be a valid preprocessing directive	3115
Rule-20.14	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related	3317, 3318
Rule-21.1	#define and #undef shall not be used on a reserved identifier or reserved macro name	0836, 0848, 0854, 4600, 4601
Rule-21.2	A reserved identifier or macro name shall not be declared	0602, 4602, 4603, 4604, 4605, 4606, 4607, 4608
Rule-21.3	The memory allocation and deallocation functions of <stdlib.h> shall not be used	5118
Rule-21.4	The standard header file <setjmp.h> shall not be used	5132
Rule-21.5	The standard header file <signal.h> shall not be used	5123
Rule-21.6	The Standard Library input/output functions shall not be used	5124
Rule-21.7	The atof, atoi, atol and atoll functions of <stdlib.h> shall not be used	5125
Rule-21.8	The library functions abort, exit, getenv and system of <stdlib.h> shall not be used	5126
Rule-21.9	The library functions bsearch and qsort of <stdlib.h> shall not be used	5135
Rule-21.10	The Standard Library time and date functions shall not be used	5127
Rule-21.11	The standard header file <tgmath.h> shall not be used	5131

Count of Enforced Required Rules = 87

6.7 Unenforced Rules - Required

Table 6.7: M3CM Unenforced Rules - Required

Rule Id	Rule
Rule-4.1	Octal and hexadecimal escape sequences shall be terminated
Rule-5.4	Macro identifiers shall be distinct
Rule-5.5	Identifiers shall be distinct from macro names
Rule-5.7	A tag name shall be a unique identifier
Rule-9.4	An element of an object shall not be initialized more than once
Rule-9.5	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly
Rule-13.1	Initializer lists shall not contain persistent side-effects
Rule-15.2	The goto statement shall jump to a label declared later in the same function
Rule-18.7	Flexible array members shall not be declared
Rule-20.8	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1
Rule-20.11	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator
Rule-20.12	A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators
Rule-22.1	All resources obtained dynamically by means of Standard Library functions shall be explicitly released
Rule-22.3	The same file shall not be open for read and write access at the same time on different streams

Count of Unenforced Required Rules = 14

6.8 Not Statically Enforceable Rules - Required

Table 6.8: M3CM Not Statically Enforceable Rules - Required

Rule Id	Rule
Dir-1.1	Any implementation-defined behaviour on which the output of the program depends shall be documented and understood
Dir-2.1	All source files shall compile without any compilation errors

continued on next page

Table 6.8 – continued from previous page

Rule Id	Rule
Dir-3.1	All code shall be traceable to documented requirements
Dir-4.1	Run-time failures shall be minimized
Dir-4.3	Assembly language shall be encapsulated and isolated
Dir-4.7	If a function returns error information, then that error information shall be tested
Dir-4.10	Precautions shall be taken in order to prevent the contents of a header file being included more then once
Dir-4.11	The validity of values passed to library functions shall be checked
Dir-4.12	Dynamic memory allocation shall not be used

Count of Not Statically Enforceable Required Rules = 9

6.9 Rule Enforcement Summary

Table 6.9: M3CM Rule Enforcement Summary

Rule	Number
(a) Total Number of Rules	159
(b) Total Number of 'Not Statically Enforceable' Rules	16
(c) Total Number of Enforceable Rules (a-b)	143
(d) Total Number of Enforced Rules	119
(e) Total Number of Unenforced Rules (c-d)	24
(f) Enforced Rules Percentage (d/c)	83%
(g) Unenforced Rules Percentage (e/c)	17%