DO Qualification Kit

Software Code Standards (SCS)

R2017a

How to Contact MathWorks

Latest news: [www.mathworks.com](http://www.mathworks.com/)

Sales and services: [www.mathworks.com/sales\_and\_services](http://www.mathworks.com/sales_and_services)

User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral/)

Technical support: [www.mathworks.com/support/contact\_us](http://www.mathworks.com/support/contact_us/)

Phone: 508-647-7000

The MathWorks, Inc.   
3 Apple Hill Drive  
Natick, MA 01760-2098

DO Qualification Kit: Software Code Standards (SCS)

© COPYRIGHT 2014 – 2015 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government)and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government’s needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

Revision History

February 2016 New for DO Qualification Kit Version 3.0 (Applies to Release 2015b)

Document Title: **Software Code Standards for <*Project*>.**

**Document Number: *<DocNo>***

Revision: ***<Revision>***

Project: **<*Project*>**

|  |  |  |
| --- | --- | --- |
| Approvals: |  |  |
| *<Name1>*, Author |  | Date |
|  |  |  |
| *<Name 2>*, Project Management |  | Date |
|  |  |  |
| *<Name 3>*, Engineering |  | Date |
|  |  |  |
| *<Name 4>*, Quality Engineering |  | Date |

| Change History | | | | |
| --- | --- | --- | --- | --- |
| Rev. | Modification / Description | Date | Author | Checked |
| 1.0 | *First release of Software Code Standards (SCS)* |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Contents

[1 Introduction 1-1](#_Toc493052343)

[1.1 Purpose and Scope 1-2](#_Toc493052344)

[1.2 Applicable Documents 1-3](#_Toc493052345)

[Table 1 – Regulations and Standards 1-3](#_Toc493052346)

[Table 2 – Company and Project Plans, Standards, and Documents 1-3](#_Toc493052347)

[1.3 Referenced Documents 1-5](#_Toc493052348)

[2 Programming Languages 2-1](#_Toc493052349)

[2.1 C Language 2-2](#_Toc493052350)

[3 Source Code Presentation Standars 3-1](#_Toc493052351)

[3.1 Coding Guidelines for Safety Critical Systems 3-2](#_Toc493052352)

[3.1.1 MISRA C:2012 Directives 3-2](#_Toc493052353)

[Compilation and Build 3-2](#_Toc493052354)

[Code Design 3-2](#_Toc493052355)

[3.1.2 MISRA C:2012 Rules 3-2](#_Toc493052356)

[A standard C Environment 3-2](#_Toc493052357)

[Unused Code 3-2](#_Toc493052358)

[Comments 3-2](#_Toc493052359)

[Character Sets and Lexical Conventions 3-3](#_Toc493052360)

[Identifiers 3-3](#_Toc493052361)

[Types 3-3](#_Toc493052362)

[Literals and Constants 3-3](#_Toc493052363)

[Declarations and Definitions 3-3](#_Toc493052364)

[Initialization 3-4](#_Toc493052365)

[Pointer Type Conversions 3-4](#_Toc493052366)

[Expressions 3-4](#_Toc493052367)

[Side Effects 3-4](#_Toc493052368)

[Control Statement Expressions 3-5](#_Toc493052369)

[Control Flow 3-5](#_Toc493052370)

[Functions 3-5](#_Toc493052371)

[Pointers and Arrays 3-5](#_Toc493052372)

[Overlapping Storage 3-5](#_Toc493052373)

[Preprocessing Directives 3-6](#_Toc493052374)

[Standard Libraries 3-6](#_Toc493052375)

[Resources 3-7](#_Toc493052376)

[4 Naming Conventions 4-1](#_Toc493052377)

[4.1 Automatic Generated Identifiers 4-2](#_Toc493052378)

[4.2 User-Defined Identifiers 4-3](#_Toc493052379)

[5 Coding Conditions and Constraints 5-1](#_Toc493052380)

[5.1 Coding Conditions 5-2](#_Toc493052381)

[5.2 Coding Constraints 5-3](#_Toc493052382)

[6 Constraints on Coding Tools 6-1](#_Toc493052383)

[6.1 Coder Constraints 6-2](#_Toc493052384)

[6.2 Compiler and Linker Constraints 6-3](#_Toc493052385)

# Introduction

## Purpose and Scope

This document comprises the Software Code Standards as referenced by the Software Development Plan (SDP), according to DO-178C 11.2 and DO-331 MB.11.2 for the project <*Project*>. As specified in DO-178C 11.8 and DO-331 MB.11.8, the Software Code Standards define the programming languages, methods, rules, and tools to be used to code the software.

Therefore, this document defines all of the following:

* Programming languages.
* Source code presentation standards.
* Naming conventions.
* Conditions and constraints on coding conventions.
* Constraints on coding tools.

This document provides the planning data defined in [DO-178C] Section 11.8 and [DO-331] Section MB.11.8, respectively.

You can use this SCS template as a resource when creating a SCS. If you are updating an existing SCS to support Model-Based Design (MBD), you can use this template as a reference document. Although representative of SCSs used in the industry, this SCS template has not been reviewed, approved, or accepted by any certification authority. It is the user’s responsibility to gain approval and acceptance of their SCS by the appropriate certification authority.

## Applicable Documents

Table 1 – Regulations and Standards

| ID | Document Title |
| --- | --- |
| DO-178C | *Software Considerations in Airborne Systems and Equipment Certification*.  RTCA, Inc., 2011 |
| DO-330 | *Software Tool Qualification Considerations*.  RTCA, Inc., 2011 |
| DO-331 | *Model-Based Development and Verification Supplement to DO-178C and DO-278A*.  RTCA, Inc., 2011 |
|  | *<List additional documents here, e.g. Advisory Circulars, EASA Certification Memos, etc.>* |

Table 2 – Company and Project Plans, Standards, and Documents

| Document | Document Title |
| --- | --- |
| PSAC | ***Plan for Software Aspects of Certification for*** <*Project*> |
| SDP | ***Software Development Plan for*** <*Project*> |
| SVP | ***Software Verification Plan for*** <*Project*> |
| SCMP | ***Software Configuration Management Plan for*** <*Project*> |
| SQAP | ***Software Quality Assurance Plan for*** <*Project*> |
| SRS | ***Software Requirements Standards for*** <*Project*> |
| SDS | ***Software Design Standards for*** <*Project*> |
| SCS | ***Software Code Standards for*** <*Project*> This document. |
| SMS | ***Software Model Standards for*** <*Project*> |
| SCI | ***Software Configuration Index for*** <*Project*> \* |
| SECI | ***Software Life Cycle Environment Configuration Index for*** <*Project*> \*\* |
| SAS | *Software Accomplishment Summary* ***for*** <*Project*> |
|  | *<List additional documents>* |

\* The information that defines the software configuration for the project <*Project*> is recorded in the *Software Configuration Index* (SCI).

\*\* The tool environment used for the project <*Project*> is defined in the *Software Life Cycle Environment Configuration Index* (SECI).

This initial release will identify the versions of completed documents, versions of the tools used, and the initial software configuration.

At the completion of the project, the SCIs and SECI will be updated with the final configuration information and final document version information.

If any of the plans are revised during the project, the reasons for the changes are captured and documented in the *Software Accomplishment Summary* (SAS).

## Referenced Documents

| ID | Document Title |
| --- | --- |
| MISRA | *Guidelines for the Use of the C Language in Critical Systems (MISRA C:2012)* |
| ECUG | *Embedded Coder User's Guide* |
| SLCIREF | *Simulink Code Inspector Reference* |
|  |  |
|  | *<List additional documents here.>* |

# Programming Languages

This section specifies the programming languages for the development of source code, according to (DO-178C 11.8.a).

## C Language

C is a high-level computer programming language use for imperative, structured programming. C must be used in conjuction with a compiler to translate the source code into the executable object code.

ISO C or ANSI C

# Source Code Presentation Standars

This section specifies the rules and style guidelines for the use of C as a programming language for source code development, according to (DO-178C 11.8.b).

## Coding Guidelines for Safety Critical Systems

### MISRA C:2012 Directives

#### The Implementation

1.1: Any implementation-defined behaviour on which the output of the program depends shall be documented and understood. Required.

#### Compilation and Build

2.1: All source files shall compile without any compilation errors. Required.

#### Code Design

4.1: Run-time failures shall be minimized. Required.

4.3: Assembly language shall be encapsulated and isolated. Required.

4.7: If a function returns error information, then that error information shall be tested. Required.

4.10: Precautions shall be taken in order to prevent the contents of a header file being included more than once. Required.

4.11: The validity of values passed to library functions shall be checked. Required.

4.14: The validity of values received from external sources shall be checked. Required.

### MISRA C:2012 Rules

#### A standard C Environment

1.1: The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits. Required.

1.3: There shall be no occurrence of undefined or critical unspecified behaviour. Required.

#### Unused Code

2.1: A project shall not contain unreachable code. Required.

2.2: There shall be no dead code. Required.

2.7: There should be no unused parameters in functions. Readability.

#### Comments

3.1: The character sequences /\* and // shall not be used within a comment. Required.

3.2: Line-splicing shall not be used in // comments. Required.

#### Character Sets and Lexical Conventions

4.1: Octal and hexadecimal escape sequences shall be terminated. Required.

#### Identifiers

5.1: External identifiers shall be distinct. Required.

5.2: Identifiers declared in the same scope and name space shall be distinct. Required.

5.4: Macro identifiers shall be distinct. Required.

5.5: Identifiers shall be distinct from macro names. Required.

5.6: A typedef name shall be a unique identifier. Required.

5.7: A tag name shall be a unique identifier. Required.

5.8: Identifiers that define objects or functions with external linkage shall be unique. Required.

#### Types

6.1: Bit-fields shall only be declared with an appropriate type. Required.

6.2: Single-bit named bit fields shall not be of a signed type. Required.

#### Literals and Constants

7.4: A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char". Required.

#### Declarations and Definitions

8.1: Types shall be explicitly specified. Required.

8.2: Function types shall be in prototype form with named parameters. Required.

8.3: All declarations of an object or function shall use the same names and type qualifiers. Required.

8.6: An identifier with external linkage shall have exactly one external definition. Required.

8.8: The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage. Required.

8.10: An inline function shall be declared with the static storage class. Required.

8.12: Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique. Required.

#### Initialization

9.1: The value of an object with automatic storage duration shall not be read before it has been set. Mandatory.

9.4: An element of an object shall not be initialized more than once. Required.

#### Pointer Type Conversions

11.1: Conversions shall not be performed between a pointer to a function and any other type. Required.

11.2: Conversions shall not be performed between a pointer to an incomplete type and any other type. Required.

11.3: A cast shall not be performed between a pointer to object type and a pointer to a different object type. Required.

11.6: A cast shall not be performed between pointer to void and an arithmetic type. Required.

11.7: A cast shall not be performed between pointer to object and a non-integer arithmetic type. Required.

11.8: A cast shall not remove any const or volatile qualification from the type pointed to by a pointer. Required.

#### Expressions

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. Required.

12.5: The sizeof operatorshall not have an operand which is a function parameter declared as “array of type”. Mandatory.

#### Side Effects

13.1: Initializer lists shall not contain persistent side effects. Required.

13.2: The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders. Required.

13.5: The right hand operand of a logical && or || operator shall not contain persistent side effects. Required.

13.6: The operand of the sizeof operator shall not contain any expression which has potential side effects. Mandatory.

#### Control Statement Expressions

14.1: A loop counter shall not have essentially floating type. Advisory.

14.2: A for loop shall be well-formed. Readability.

14.3: Controlling expressions shall not be invariant. Required.

#### Control Flow

15.6: The body of an iteration-statement or a selection-statement shall be a compound-statement. Required.

#### Functions

17.1: The features of <stdarg.h> shall not be used. Required.

17.2: Functions shall not call themselves, either directly or indirectly. Required.

17.3: A function shall not be declared implicitly. Mandatory.

17.4: All exit paths from a function with non-void return type shall have an explicit return statement with an expression. Mandatory.

17.6: The declaration of an array parameter shall not contain the static keyword between the [ ]. Mandatory.

#### Pointers and Arrays

18.1: A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand. Required.

18.2: Subtraction between pointers shall only be applied to pointers that address elements of the same array. Required.

18.3: The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object. Required.

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. Required.

18.7: Flexible array members shall not be declared. Required.

18.8: Variable-length array types shall not be used. Required.

#### Overlapping Storage

19.1: An object shall not be assigned or copied to an overlapping object. Mandatory.

#### Preprocessing Directives

20.2: The ', " or \ characters and the /\* or // character sequences shall not occur in a header file name. Required.

20.3: The #include directive shall be followed by either a <filename> or "filename"sequence. Required.

20.4: A macro shall not be defined with the same name as a keyword. Required.

20.6: Tokens that look like a preprocessing directive shall not occur within a macro argument. Required.

20.7: Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses. Required.

20.9: All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation. Required.

20.11: A macro parameter immediately following a # operator shall not immediately be followed by a ## operator. Required.

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. Required.

20.13: A line whose first token is # shall be a valid preprocessing directive. Required.

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. Required.

#### Standard Libraries

21.1: #define and #undef shall not be used on a reserved identifier or reserved macro name. Required.

21.2: A reserved identifier or macro name shall not be declared. Required.

21.3: The memory allocation and deallocation functions of <stdlib.h> shall not be used. Required.

21.4: The standard header file <setjmp.h> shall not be used. Required.

21.5: The standard header file <signal.h> shall not be used. Required.

21.6: The Standard Library input/output functions shall not be used. Required.

21.7: The atof, atoi, atol, and atoll functions of <stdlib.h> shall not be used. Required.

21.8: The library functions abort, exit, getenv and system of <stdlib.h> shall not be used. Required.

21.9: The library functions bsearch and qsort of <stdlib.h> shall not be used. Required.

21.10: The Standard Library time and date functions shall not be used. Required.

21.11: The standard header file <tgmath.h> shall not be used. Required.

21.13: Any value passed to a function in <ctype.h> shall be representable as an unsigned char or be the value EOF. Mandatory.

21.14: The Standard Library function memcmp shall not be used to compare null terminated strings. Required.

21.15: The pointer arguments to the Standard Library function memcpy, memmove and memcmp shall be pointers to qualified or unqualified versions of compatible types. Required.

21.16: The pointer arguments to the Standard Library function memcpy shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type. Required.

21.17: Use of the string handling functions from <string.h> shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters. Mandatory.

21.18: The size\_t argument passed to any function in <string.h> shall have an appropriate value. Mandatory.

21.19: The pointers returned by the Standard Library functions localeconv, getenv, setlocale or, strerror shall only be used as if they have pointer to const-qualified type. Mandatory.

21.20: The pointers returned by the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv,setlocale or strerror shall not be used following a subsequent call to the same function. Mandatory.

#### Resources

22.1: All resources obtained dynamically by means of Standard Library functions shall be explicitly released. Required.

22.2: A block of memory shall only be freed if it was allocated by means of a Standard Library function. Mandatory.

22.3: The same file shall not be open for read and write access at the same time on different streams. Required.

22.4: There shall be no attempt to write to a stream which has been opened as read-only. Mandatory.

22.5: A pointer to a FILE object shall not be dereferenced. Mandatory.

22.6: The value of a pointer to a FILE shall not be used after the associated stream has been closed. Mandatory.

22.7: The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF. Required.

22.8: The value of errno shall be set to zero prior to a call to an errno-setting-function. Required.

22.9: The value of errno shall be tested against zero after calling an errno-setting-function. Requried.

22.10:The value of errno shall only be tested when the last function to be called was an errno-setting-function. Required.

# 

# Naming Conventions

This section specifies the naming conventions for functions, variables, and constants in the source code, according to (DO-178C 11.8.c).

## Automatic Generated Identifiers

## User-Defined Identifiers

# Coding Conditions and Constraints

This section specifies the conditions and constraints on the allowable coding conventions, according to (DO-178C 11.8.d).

## Coding Conditions

Code replacement libraries

Overflow handling

Exception handling

## Coding Constraints

Identifier length

Word sizes

# Constraints on Coding Tools

This section specifies the constraints on the use of the applicable coding tools, according to (DO-178C 11.8.e).

## Coder Constraints

Configuration for Emebdded Coder

## Compiler and Linker Constraints

Configuration for compiler and linker