Rule Enforcement Report

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

Enforced 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
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

Unenforced Rules - Advisory

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

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

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
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

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

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, 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
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, 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
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
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-8.14	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
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, 4445, 4446, 4447, 4450, 4451, 4452, 4453, 4454, 4460, 4461, 4462, 4463, 4464, 4465
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, 4397
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
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
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 = 86

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-8.12	Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique
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 = 15

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
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

Rule Enforcement Summary

(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	**118**
(e) Total Number of Unenforced Rules (c-d)	**25**
(f) Enforced Rules Percentage (d/c)	**83 %**
(g) Unenforced Rules Percentage (e/c)	**17 %**

End of report