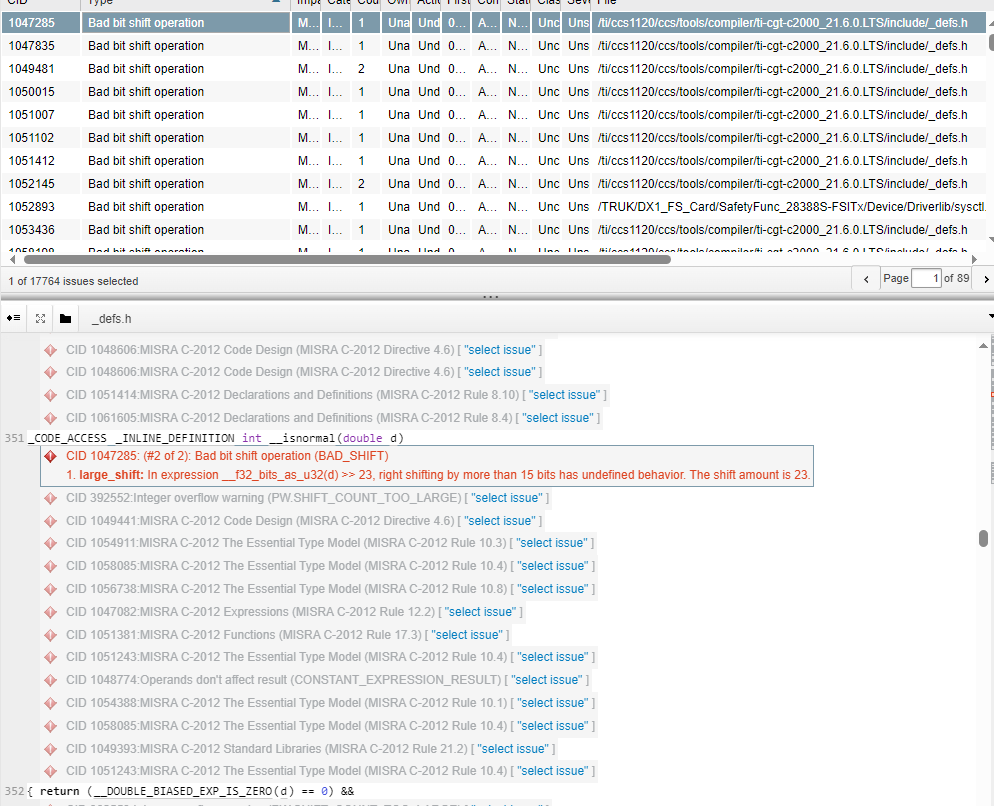
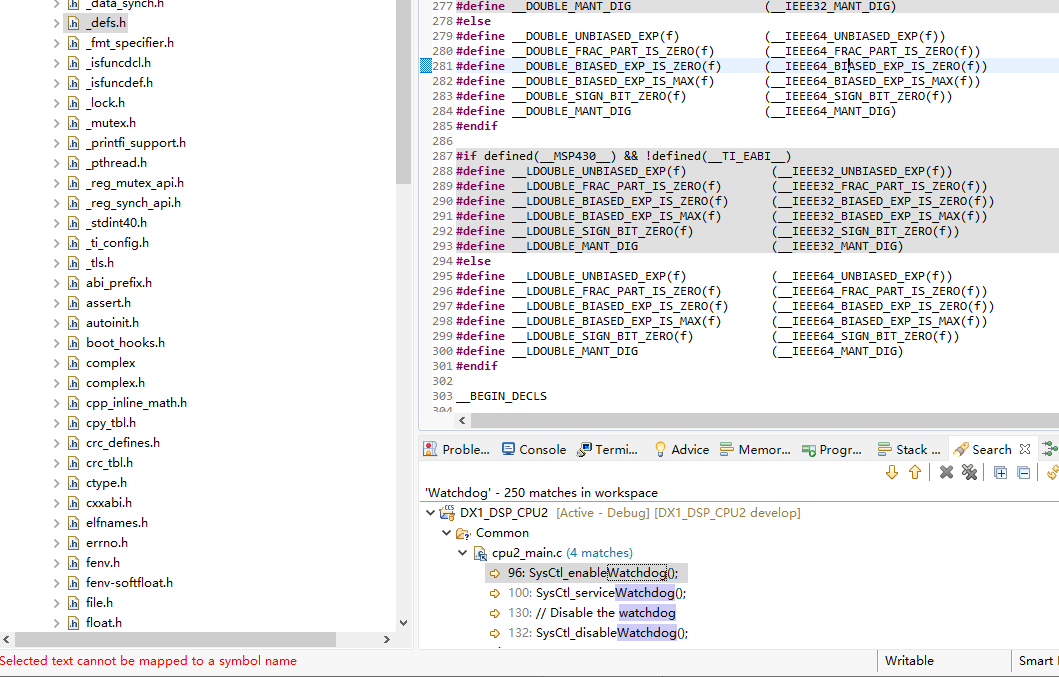
# Bad bit shift operation：移位操作错误

**Category Required**

CID 1047285: (#2 of 2): Bad bit shift operation (BAD\_SHIFT)

1. **large\_shift:** In expression \_\_f32\_bits\_as\_u32(d) >> 23, right shifting by more than 15 bits has undefined behavior. The shift amount is 23.

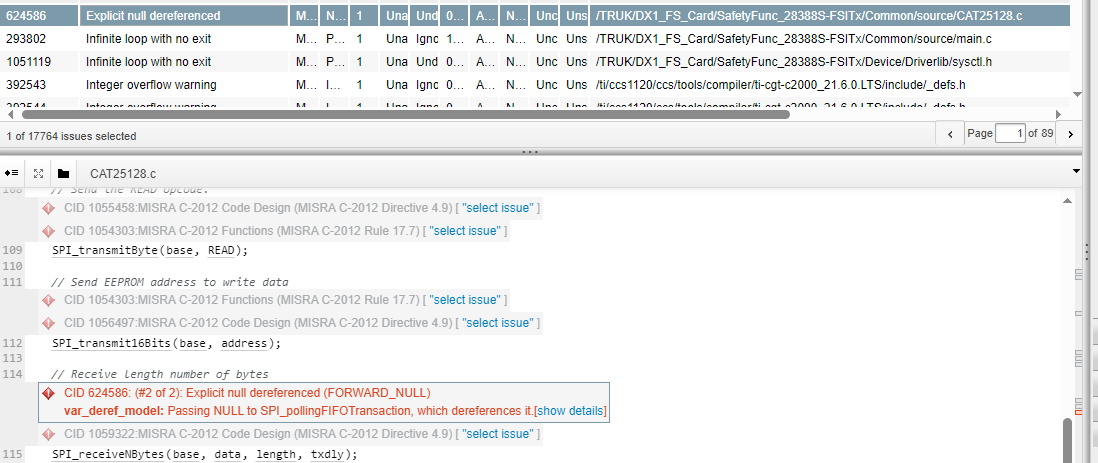
 

# Explicit null dereferenced：显式解引用空值

**Category Advisory**

CID 624586: (#2 of 2): Explicit null dereferenced (FORWARD\_NULL)

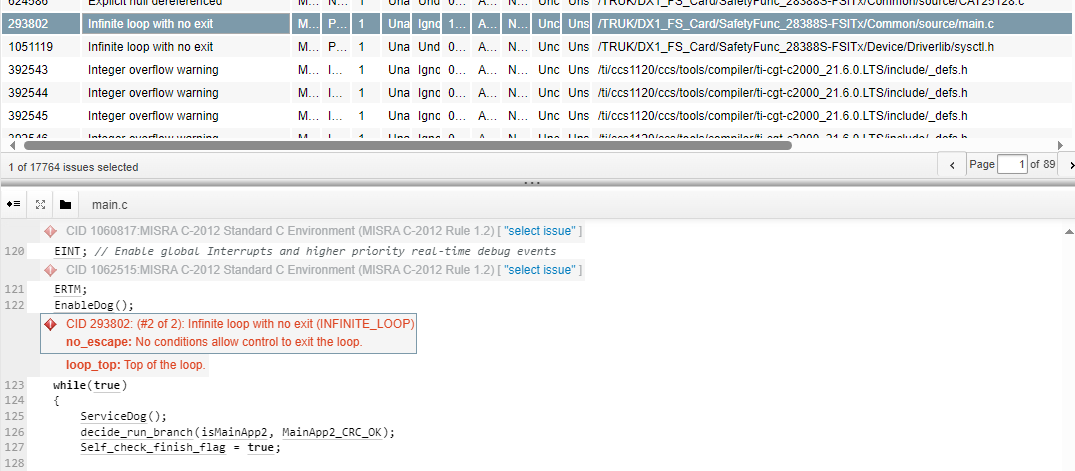
**var\_deref\_model:** Passing NULL to SPI\_pollingFIFOTransaction, which dereferences it



# Infinite loop with no exit

CID 293802: (#2 of 2): Infinite loop with no exit (INFINITE\_LOOP)

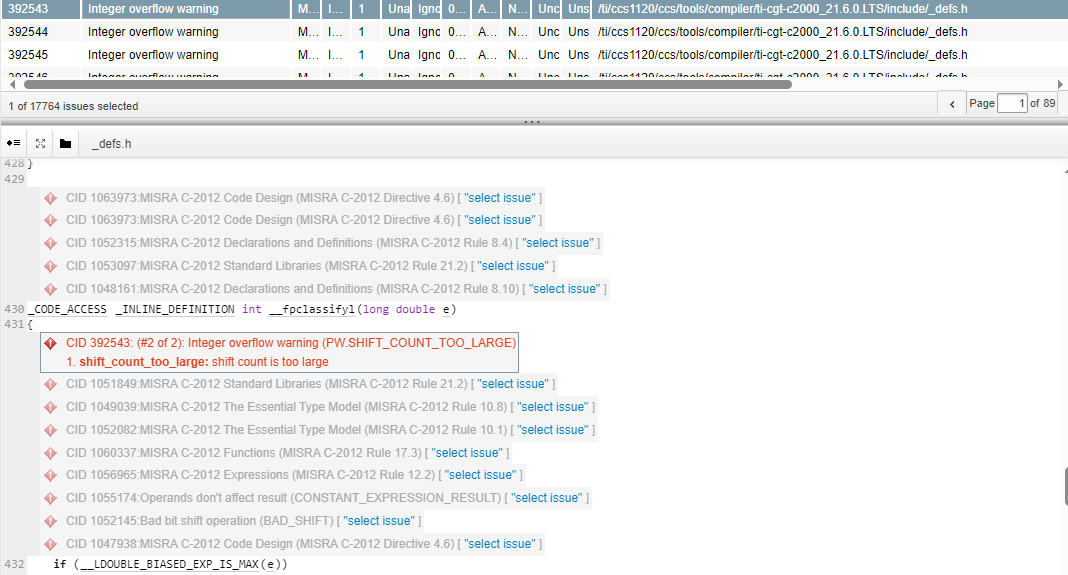
**no\_escape:** No conditions allow control to exit the loop.

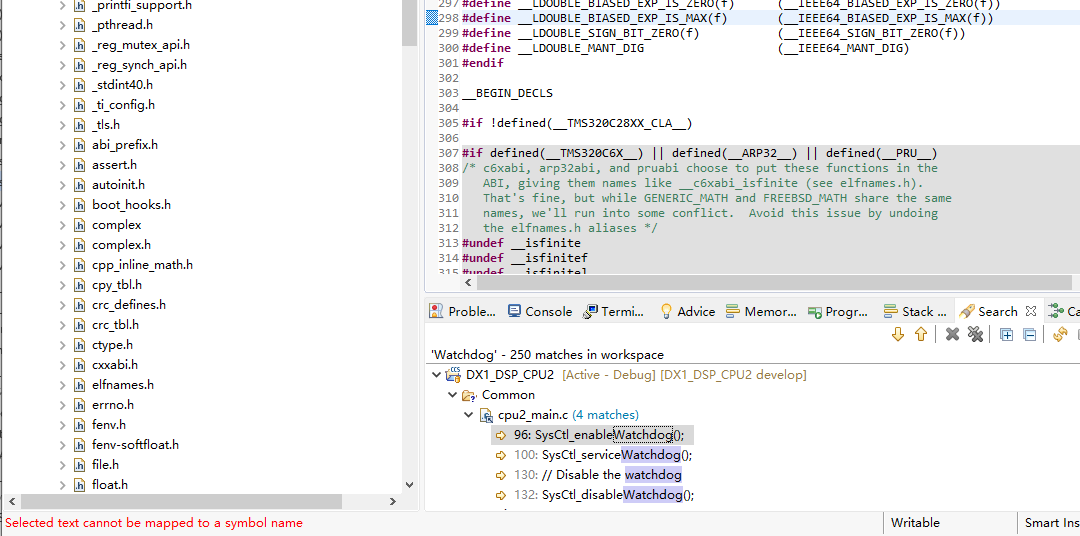


# Integer overflow warning

CID 392543: (#2 of 2): Integer overflow warning (PW.SHIFT\_COUNT\_TOO\_LARGE)

**shift\_count\_too\_large:** shift count is too large



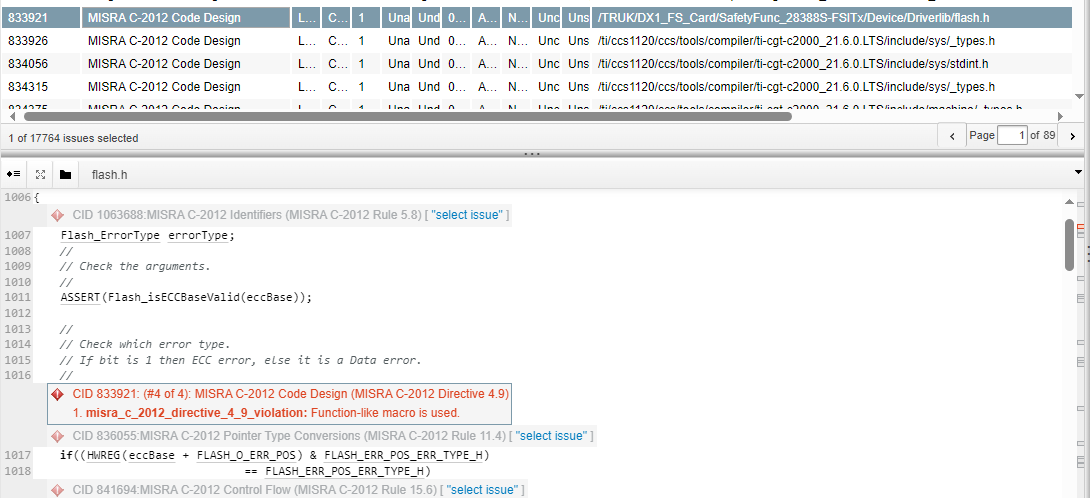


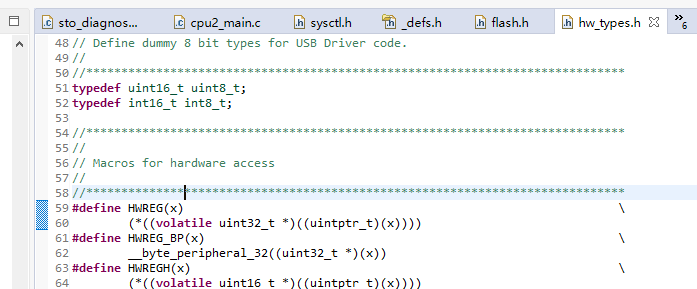
# MISRA C-2012 Code Design

**Category Advisory**

CID 833921: (#4 of 4): MISRA C-2012 Code Design (MISRA C-2012 Directive 4.9)

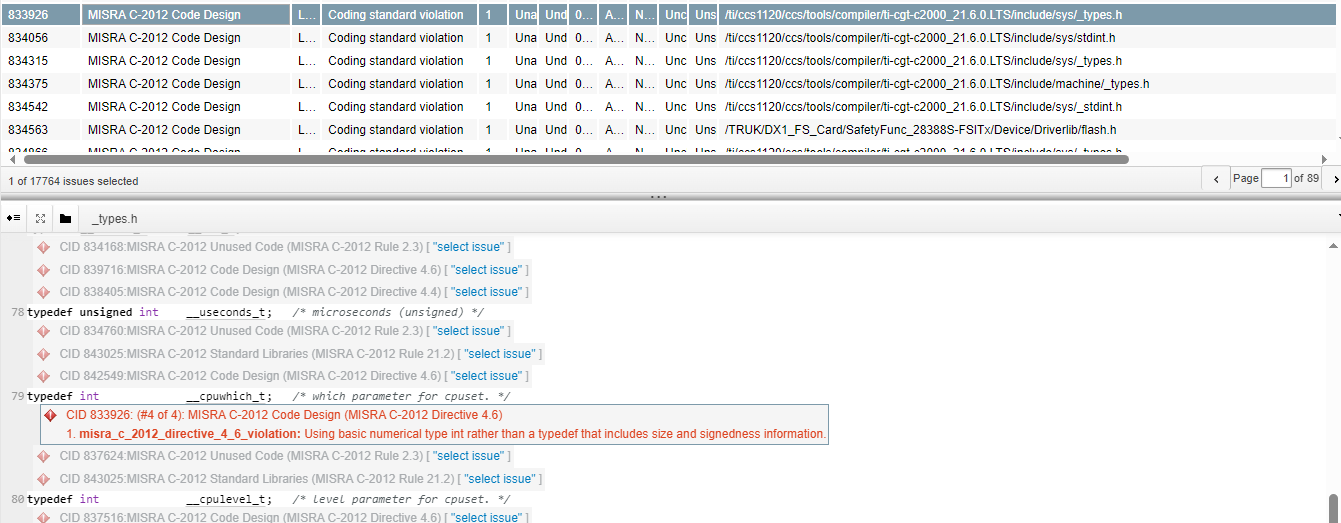
1. **misra\_c\_2012\_directive\_4\_9\_violation:** Function-like macro is used.





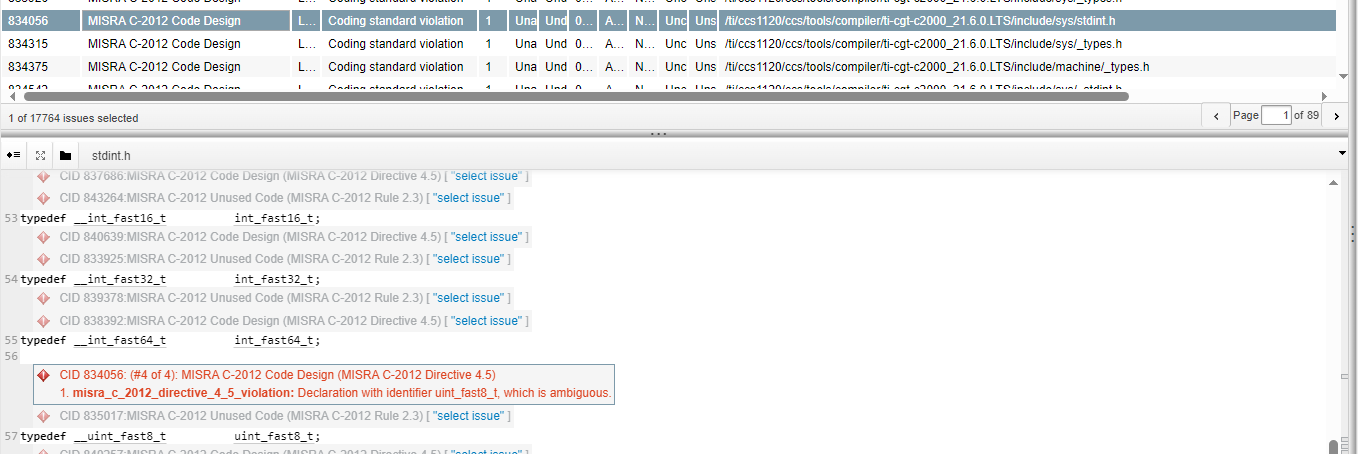
CID 833926: (#4 of 4): MISRA C-2012 Code Design (MISRA C-2012 Directive 4.6)

**misra\_c\_2012\_directive\_4\_6\_violation:** Using basic numerical type int rather than a typedef that includes size and signedness information



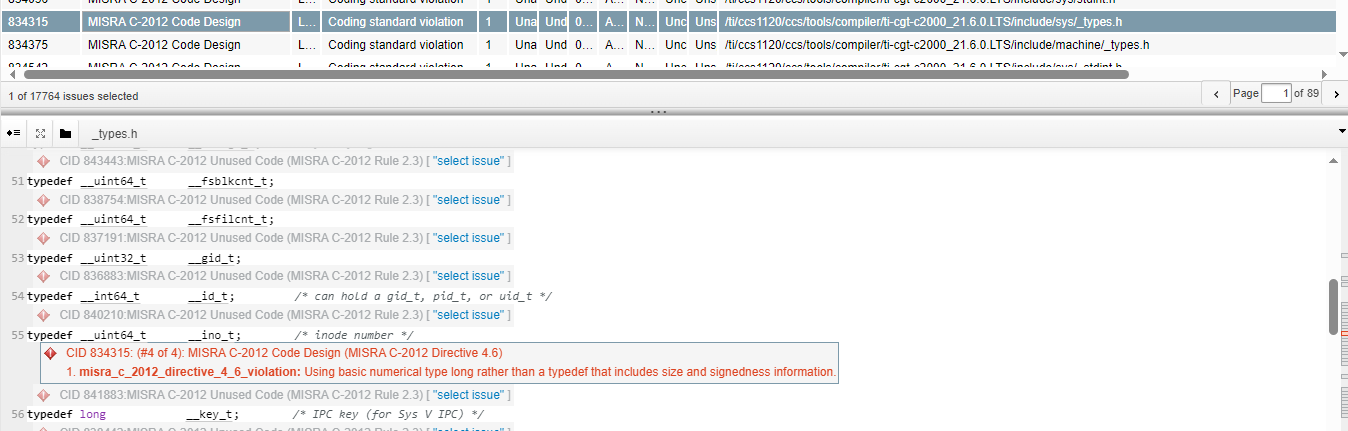
 Declaration with identifier uint\_fast8\_t, which is ambiguous.

CID 834056: (#4 of 4): MISRA C-2012 Code Design (MISRA C-2012 Directive 4.5)



CID 834315: (#4 of 4): MISRA C-2012 Code Design (MISRA C-2012 Directive 4.6)

**misra\_c\_2012\_directive\_4\_6\_violation:** Using basic numerical type long rather than a typedef that includes size and signedness information.

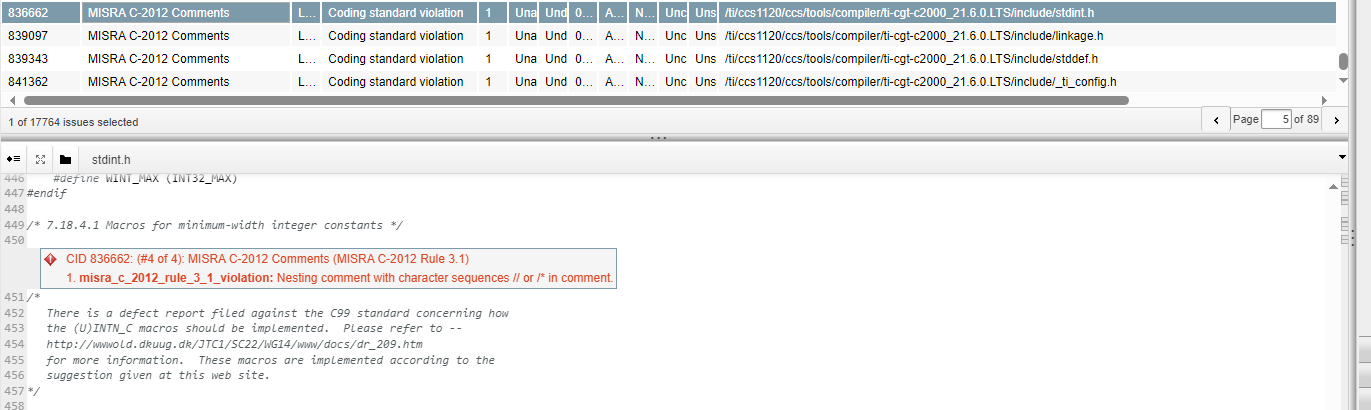


# MISRA C-2012 Comments

**Category Required**

CID 836662: (#4 of 4): MISRA C-2012 Comments (MISRA C-2012 Rule 3.1)

**misra\_c\_2012\_rule\_3\_1\_violation:** Nesting comment with character sequences // or /\* in comment.

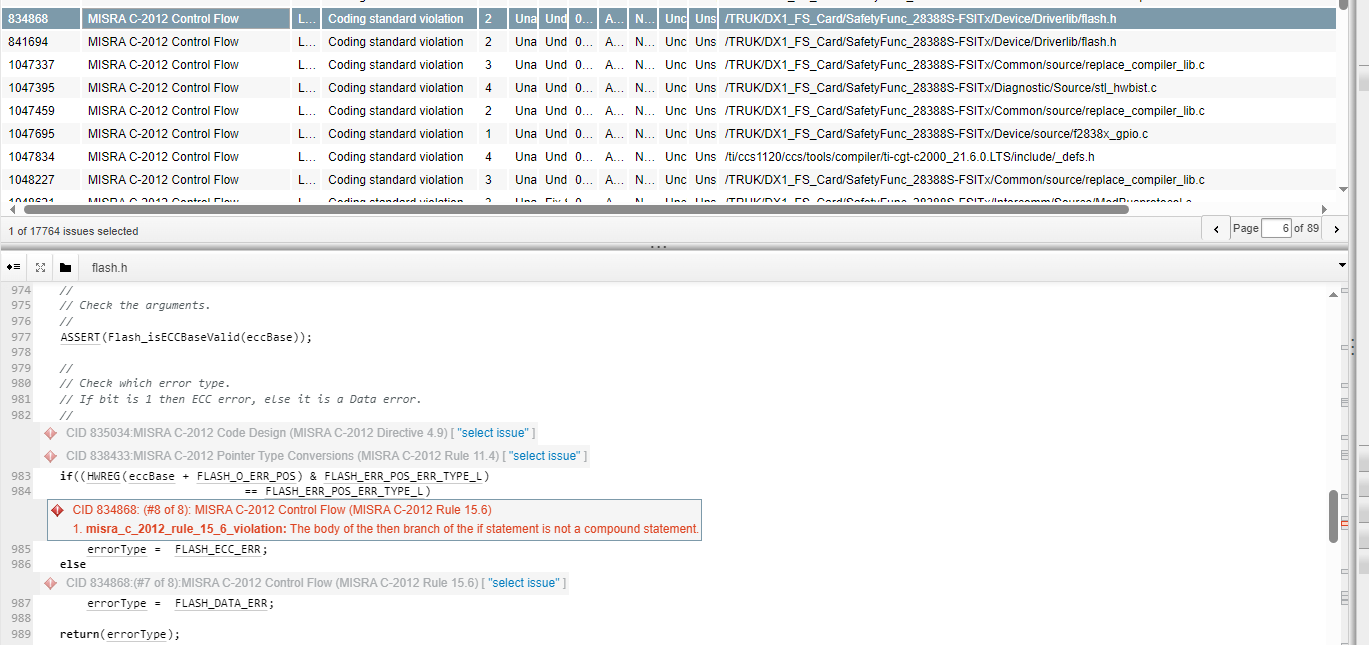


# MISRA C-2012 Control Flow

**Category Required**

CID 834868: (#8 of 8): MISRA C-2012 Control Flow (MISRA C-2012 Rule 15.6)

1. **misra\_c\_2012\_rule\_15\_6\_violation:** The body of the then branch of the if statement is not a compound statement.

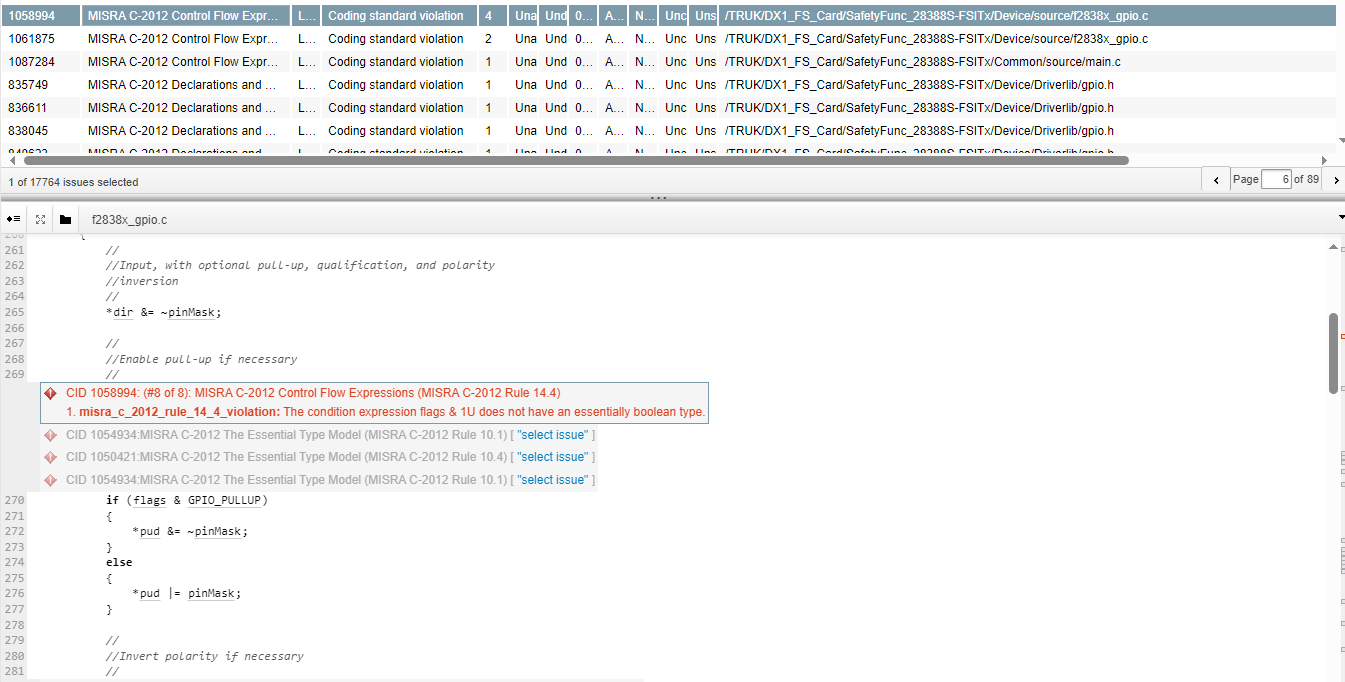


# MISRA C-2012 Control Flow Expressions

**Category Required**

CID 1058994: (#8 of 8): MISRA C-2012 Control Flow Expressions (MISRA C-2012 Rule 14.4)

1. **misra\_c\_2012\_rule\_14\_4\_violation:** The condition expression flags & 1U does not have an essentially boolean type.

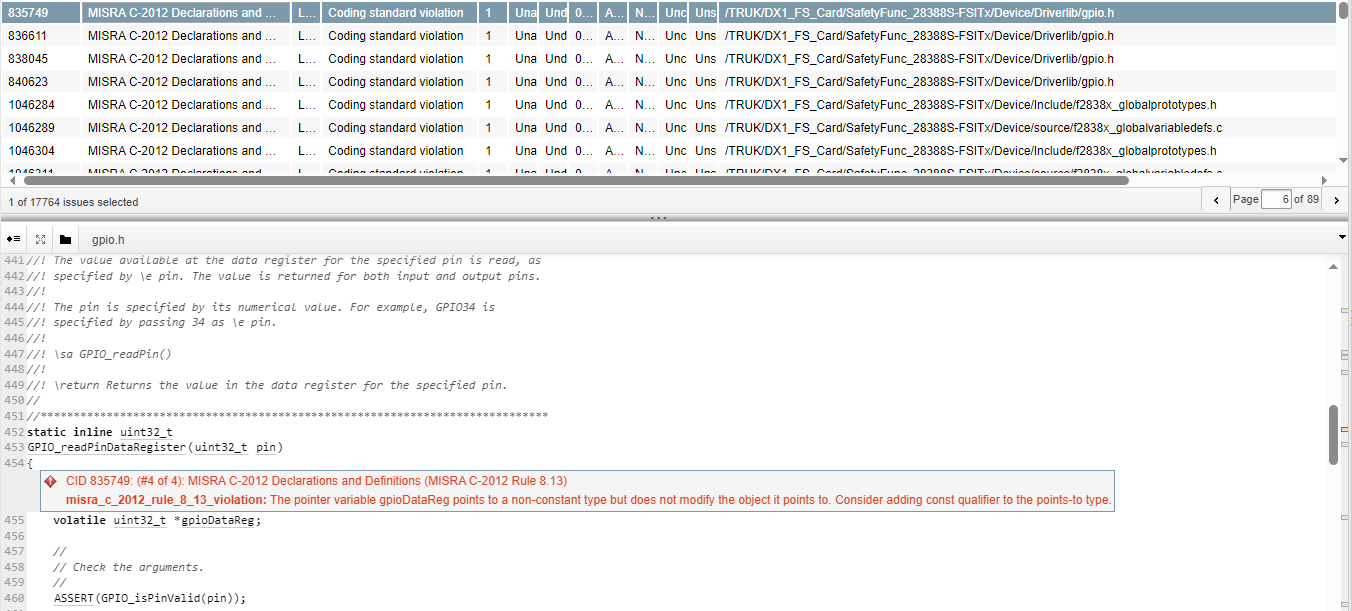


# MISRA C-2012 Declarations and Definitions

**Category Advisory**

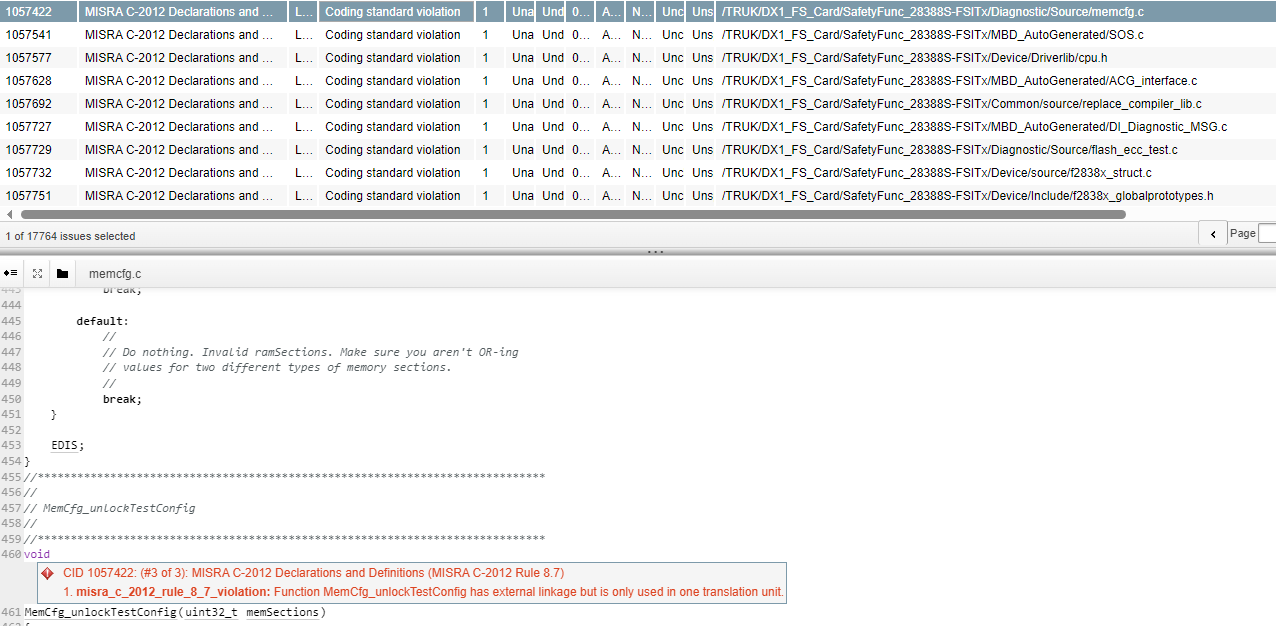
CID 835749: (#4 of 4): MISRA C-2012 Declarations and Definitions (MISRA C-2012 Rule 8.13)

**misra\_c\_2012\_rule\_8\_13\_violation:** The pointer variable gpioDataReg points to a non-constant type but does not modify the object it points to. Consider adding const qualifier to the points-to type.



CID 1057422: (#3 of 3): MISRA C-2012 Declarations and Definitions (MISRA C-2012 Rule 8.7)

1. **misra\_c\_2012\_rule\_8\_7\_violation:** Function MemCfg\_unlockTestConfig has external linkage but is only used in one translation unit

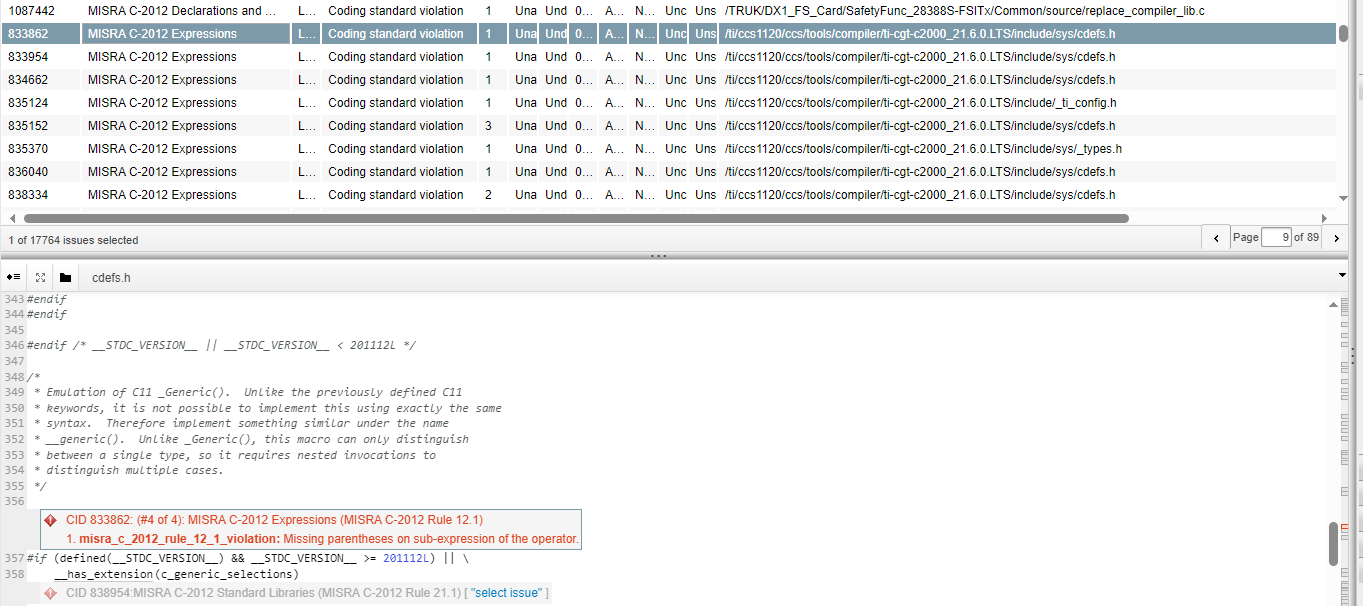


# MISRA C-2012 Expressions

**Category Advisory**

CID 833862: (#4 of 4): MISRA C-2012 Expressions (MISRA C-2012 Rule 12.1)

1. **misra\_c\_2012\_rule\_12\_1\_violation:** Missing parentheses on sub-expression of the operator.

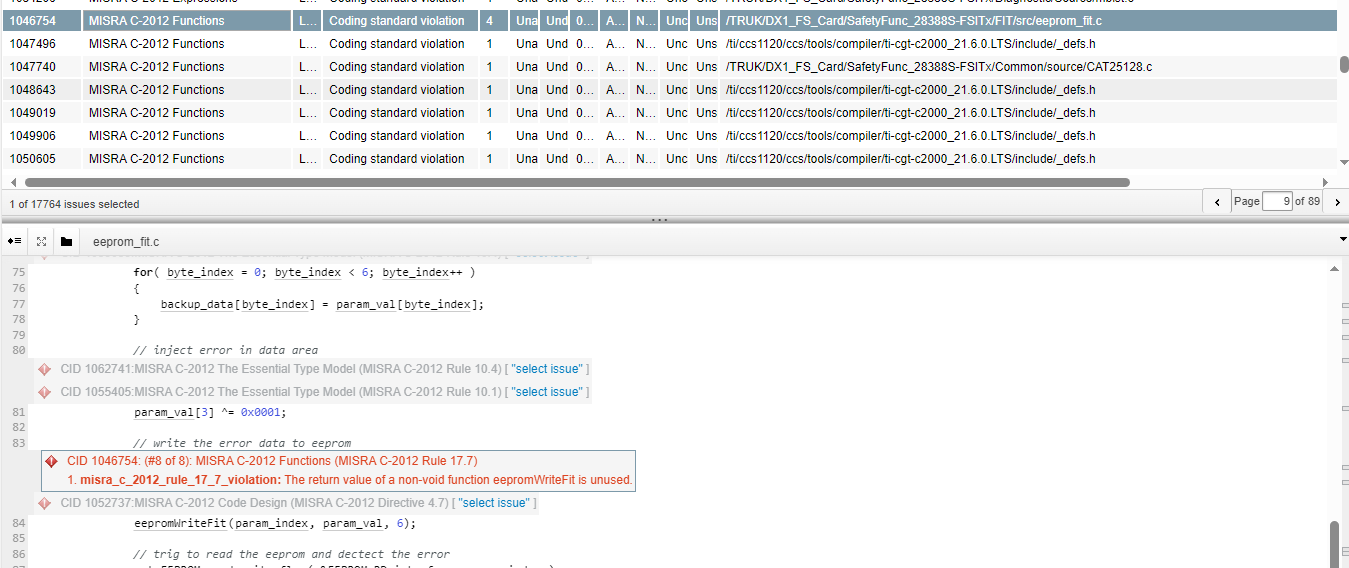


# MISRA C-2012 Functions

**Category Required**

CID 1046754: (#8 of 8): MISRA C-2012 Functions (MISRA C-2012 Rule 17.7)

1. **misra\_c\_2012\_rule\_17\_7\_violation:** The return value of a non-void function eepromWriteFit is unused.



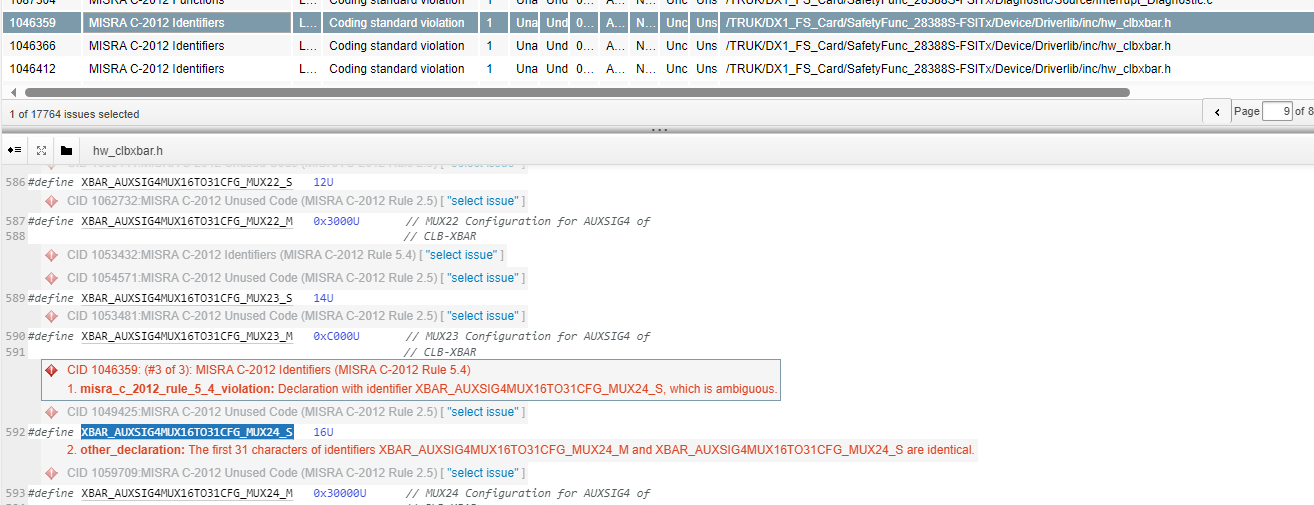
# MISRA C-2012 Identifiers

**Category Required**

CID 1046359: (#3 of 3): MISRA C-2012 Identifiers (MISRA C-2012 Rule 5.4)

1.misra\_c\_2012\_rule\_5\_4\_violation: Declaration with identifier XBAR\_AUXSIG4MUX16TO31CFG\_MUX24\_S, which is ambiguous.

2.other\_declaration: The first 31 characters of identifiers XBAR\_AUXSIG4MUX16TO31CFG\_MUX24\_M and XBAR\_AUXSIG4MUX16TO31CFG\_MUX24\_S are identical.

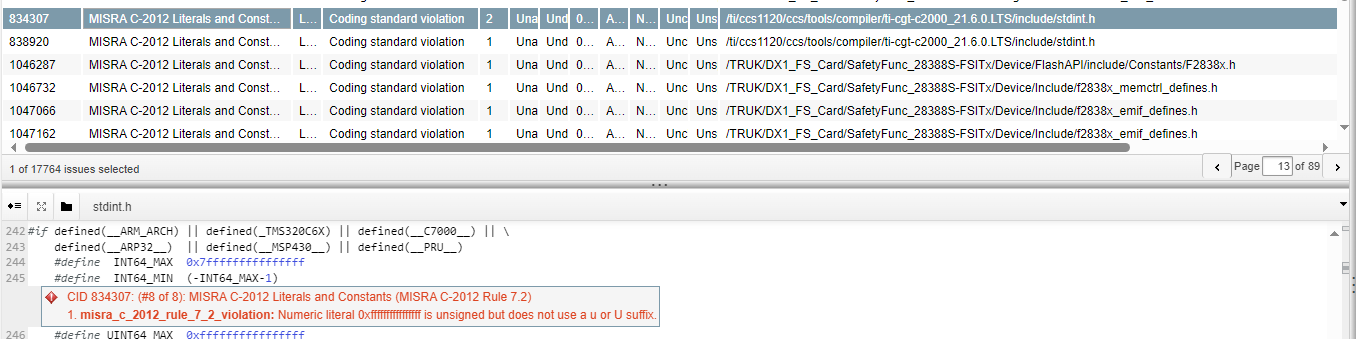


# MISRA C-2012 Literals and Constants

**Category Required**

CID 834307: (#8 of 8): MISRA C-2012 Literals and Constants (MISRA C-2012 Rule 7.2)

1. **misra\_c\_2012\_rule\_7\_2\_violation:** Numeric literal 0xffffffffffffffff is unsigned but does not use a u or U suffix.

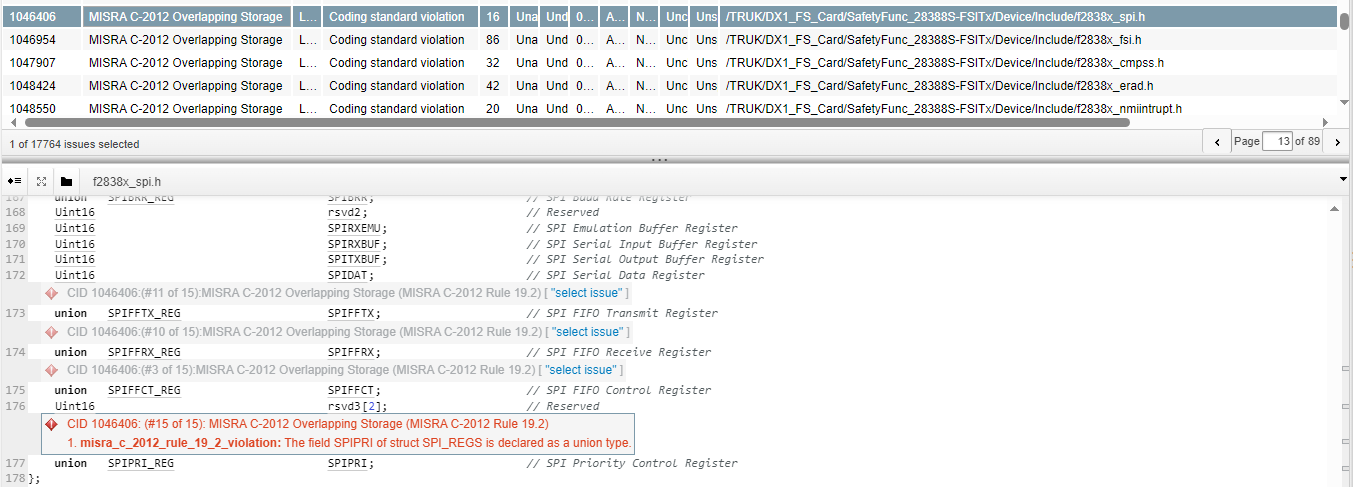


# MISRA C-2012 Overlapping Storage

**Category Advisory**

CID 1046406: (#15 of 15): MISRA C-2012 Overlapping Storage (MISRA C-2012 Rule 19.2)

**misra\_c\_2012\_rule\_19\_2\_violation:** The field SPIPRI of struct SPI\_REGS is declared as a union type.

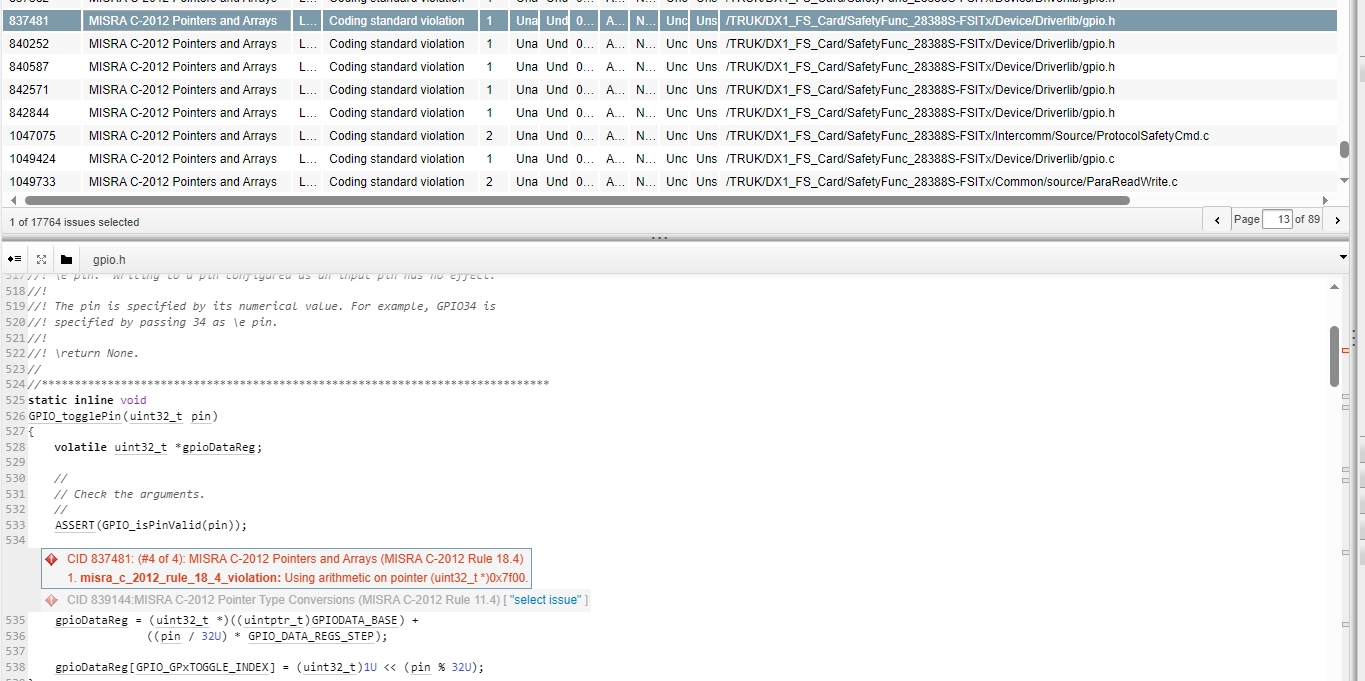
****

# MISRA C-2012 Pointers and Arrays

**Category Advisory**

CID 837481: (#4 of 4): MISRA C-2012 Pointers and Arrays (MISRA C-2012 Rule 18.4)

1. **misra\_c\_2012\_rule\_18\_4\_violation:** Using arithmetic on pointer (uint32\_t \*)0x7f00.

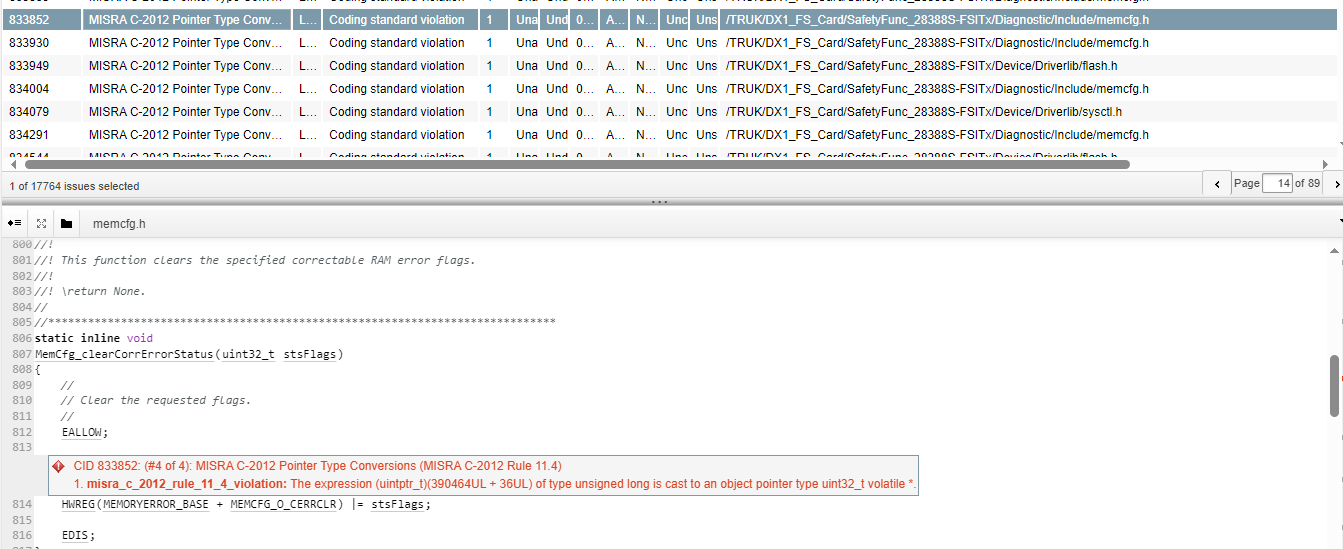
****

# MISRA C-2012 Pointer Type Conversions

**Category Advisory**

CID 833852: (#4 of 4): MISRA C-2012 Pointer Type Conversions (MISRA C-2012 Rule 11.4)

1. **misra\_c\_2012\_rule\_11\_4\_violation:** The expression (uintptr\_t)(390464UL + 36UL) of type unsigned long is cast to an object pointer type uint32\_t volatile \*.

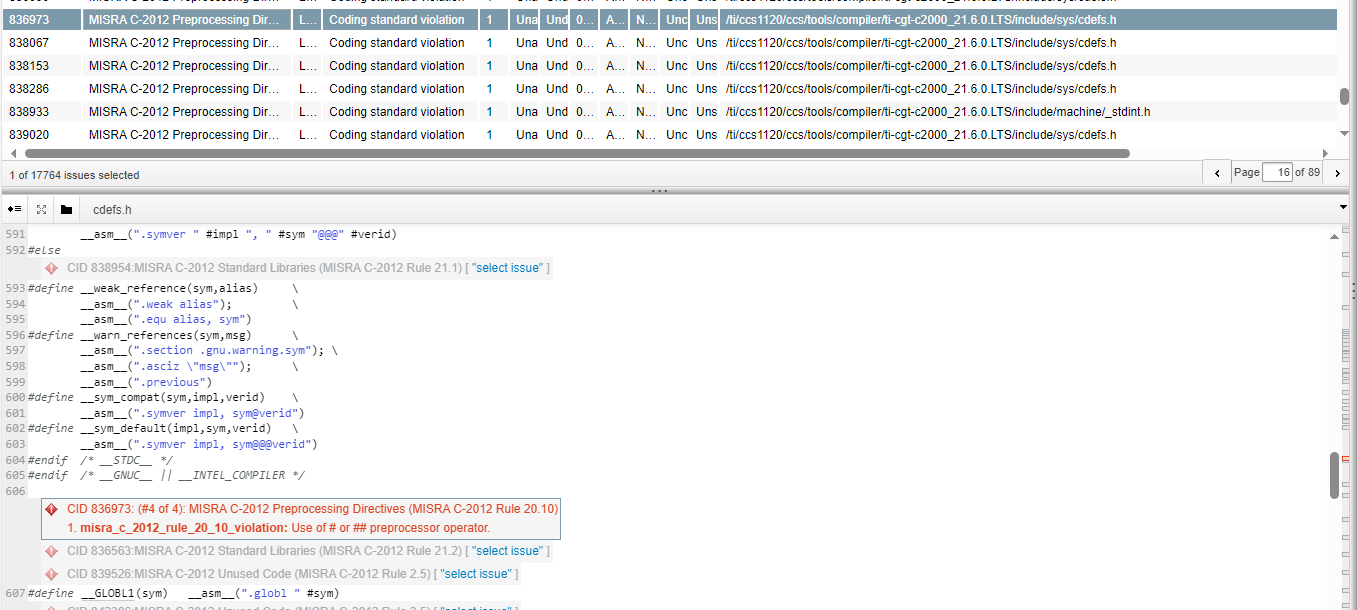
****

# MISRA C-2012 Preprocessing Directives

**Category Advisory**

CID 836973: (#4 of 4): MISRA C-2012 Preprocessing Directives (MISRA C-2012 Rule 20.10)

**misra\_c\_2012\_rule\_20\_10\_violation:** Use of # or ## preprocessor operator.

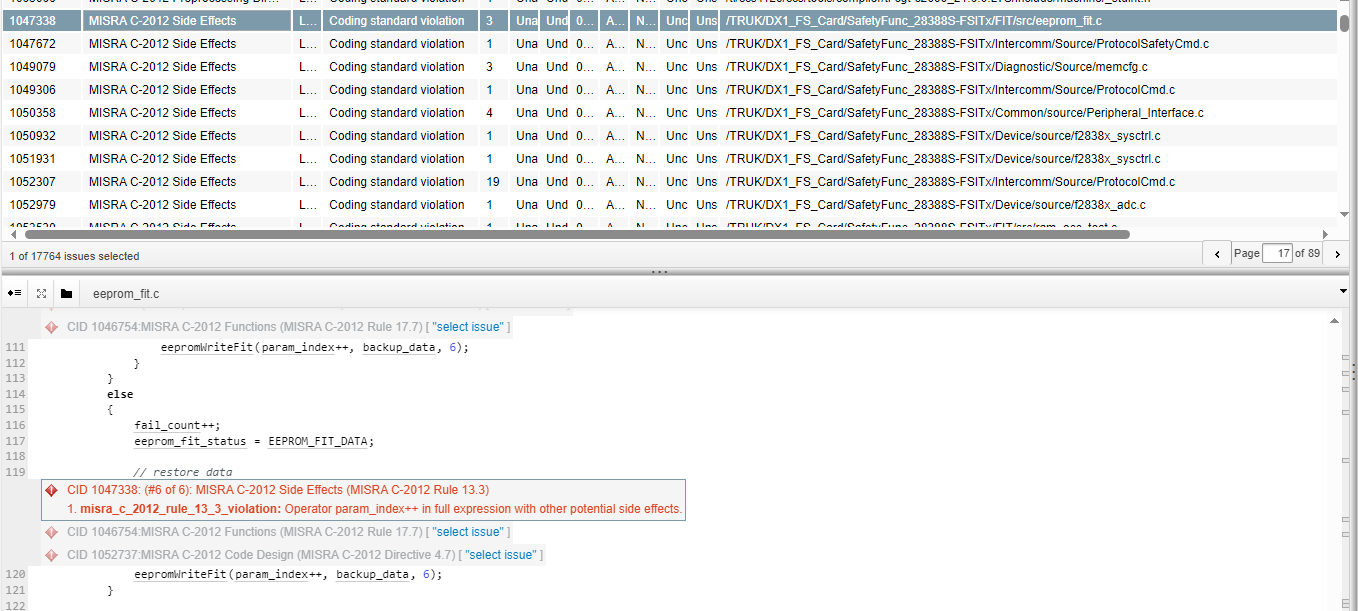
****

# MISRA C-2012 Side Effects

**Category Advisory**

CID 1047338: (#6 of 6): MISRA C-2012 Side Effects (MISRA C-2012 Rule 13.3)

**misra\_c\_2012\_rule\_13\_3\_violation:** Operator param\_index++ in full expression with other potential side effects.

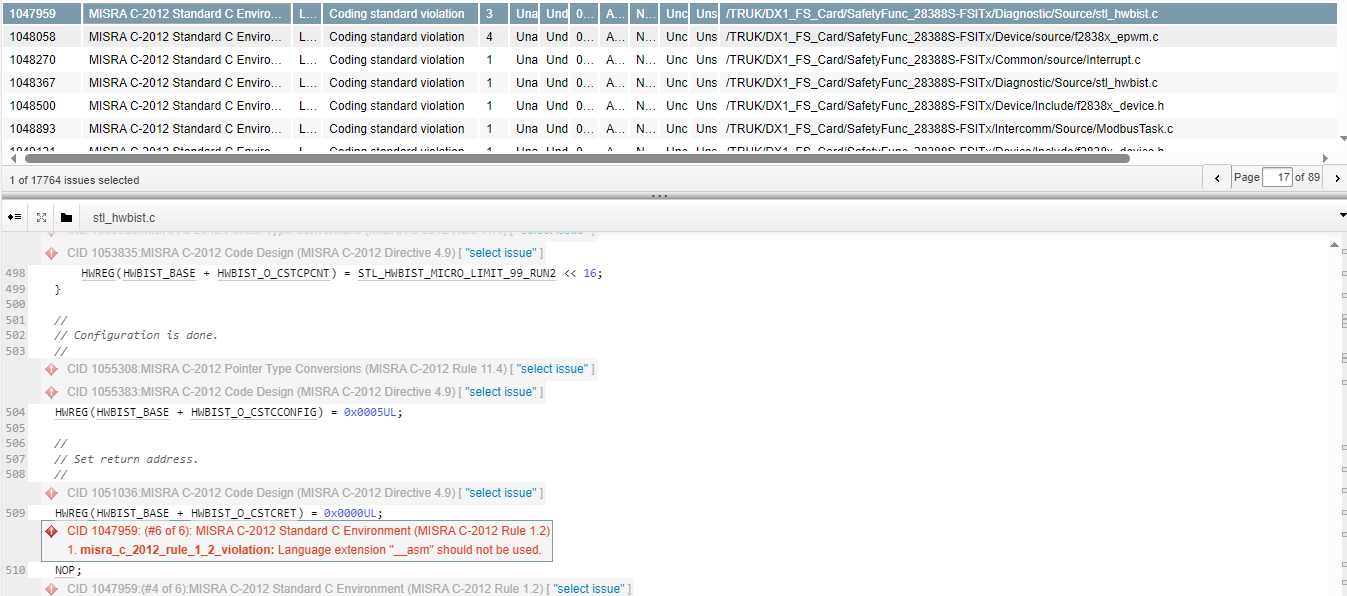
****

# MISRA C-2012 Standard C Environment

**Category Advisory**

CID 1047959: (#6 of 6): MISRA C-2012 Standard C Environment (MISRA C-2012 Rule 1.2)

**misra\_c\_2012\_rule\_1\_2\_violation:** Language extension "\_\_asm" should not be used.

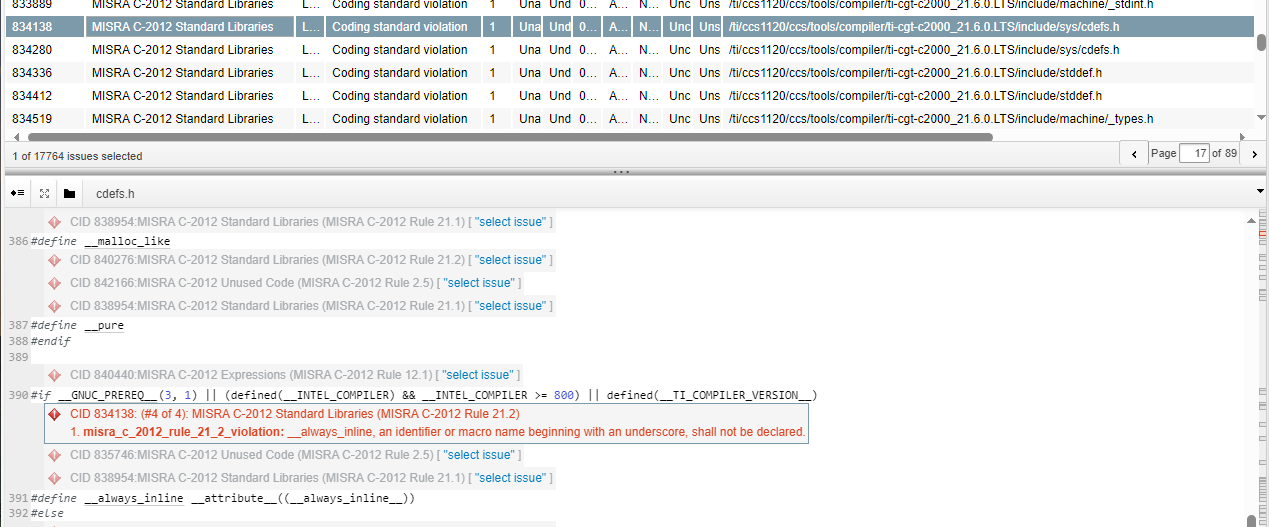
****

# MISRA C-2012 Standard Libraries

**Category Require**

CID 834138: (#4 of 4): MISRA C-2012 Standard Libraries (MISRA C-2012 Rule 21.2)

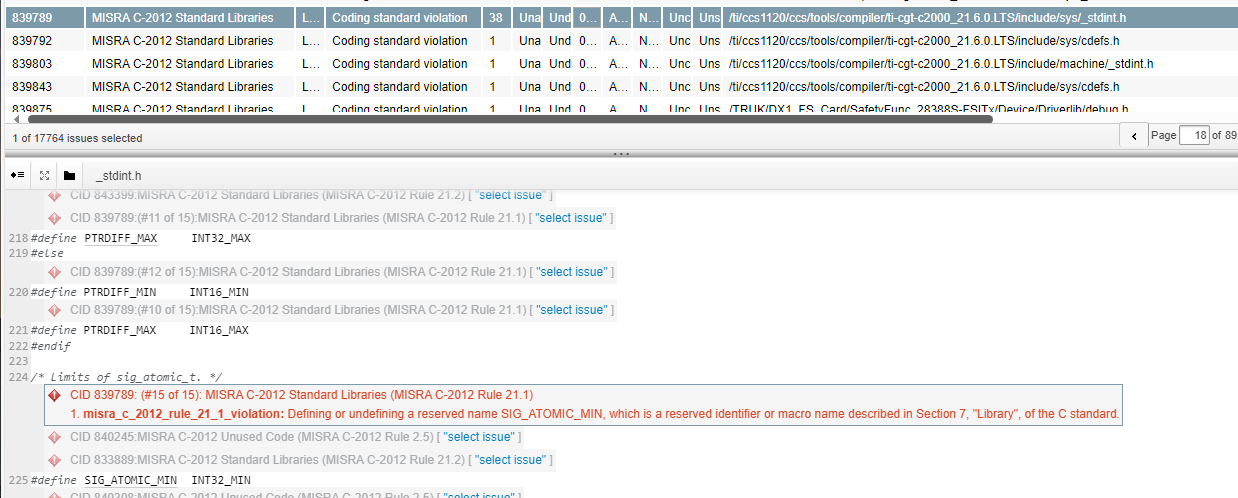
**misra\_c\_2012\_rule\_21\_2\_violation:** \_\_always\_inline, an identifier or macro name beginning with an underscore, shall not be declared.

****

CID 839789: (#15 of 15): MISRA C-2012 Standard Libraries (MISRA C-2012 Rule 21.1)

1. **misra\_c\_2012\_rule\_21\_1\_violation:** Defining or undefining a reserved name SIG\_ATOMIC\_MIN, which is a reserved identifier or macro name described in Section 7, "Library", of the C standard.

INTMAX\_MAX是C 标准第 7 节“库”中描述的保留标识符或宏名称，不得声明

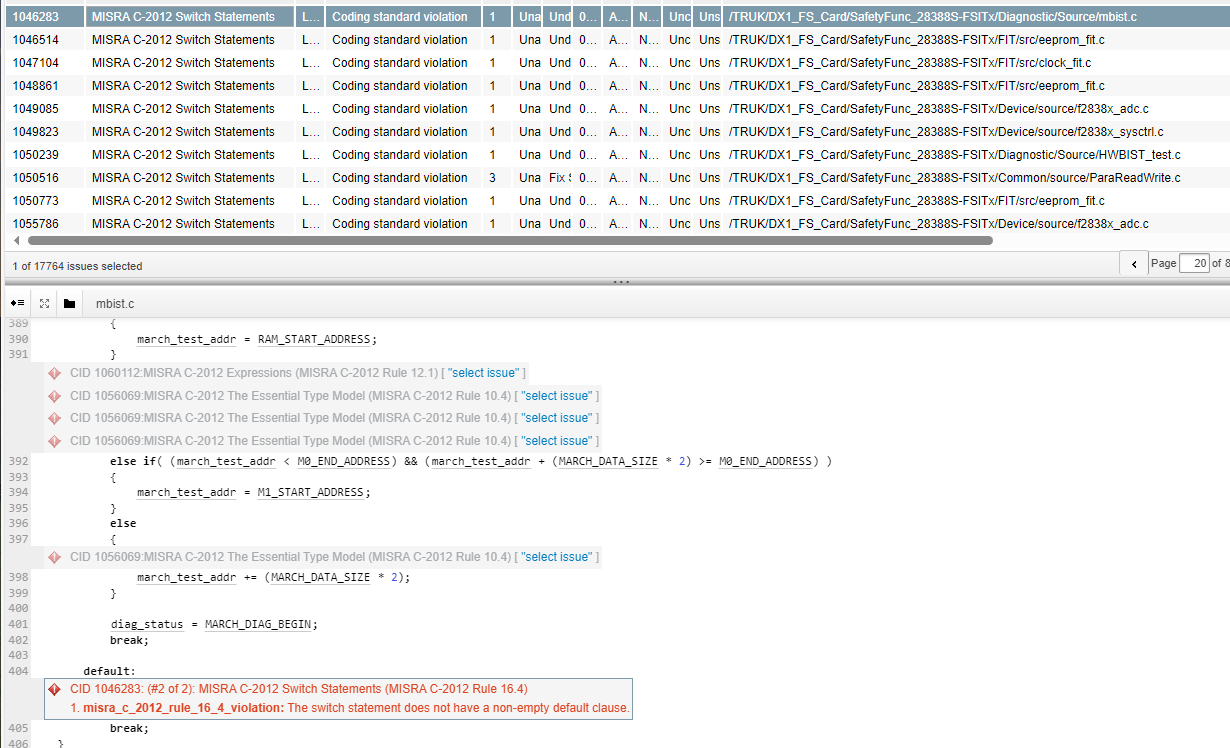
****

# MISRA C-2012 Switch Statements

**Category Required**

CID 1046283: (#2 of 2): MISRA C-2012 Switch Statements (MISRA C-2012 Rule 16.4)

. **misra\_c\_2012\_rule\_16\_4\_violation:** The switch statement does not have a non-empty default clause

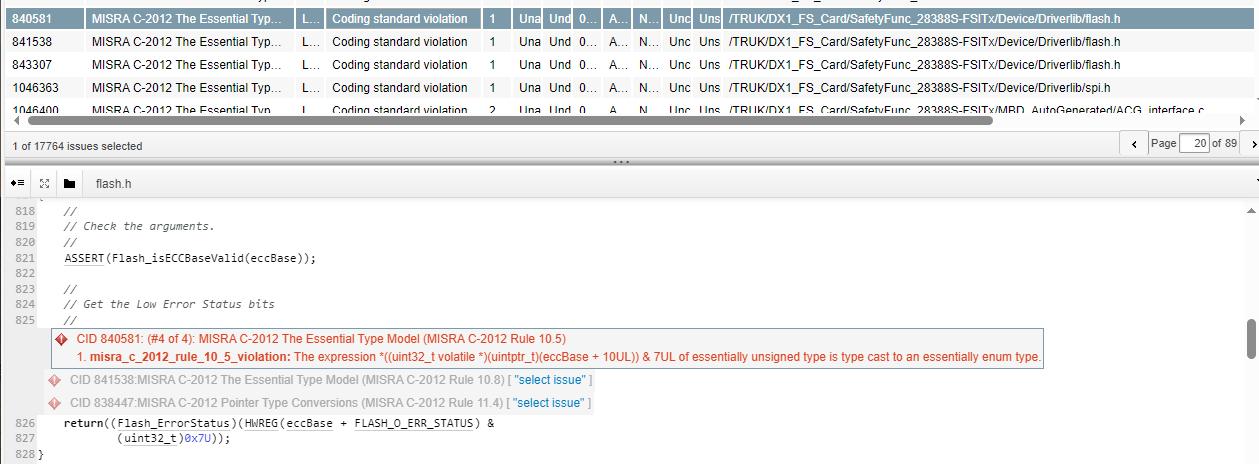
****

# MISRA C-2012 The Essential Type Model

**Category Advisory**

CID 840581: (#4 of 4): MISRA C-2012 The Essential Type Model (MISRA C-2012 Rule 10.5)

**misra\_c\_2012\_rule\_10\_5\_violation:** The expression \*((uint32\_t volatile \*)(uintptr\_t)(eccBase + 10UL)) & 7UL of essentially unsigned type is type cast to an essentially enum type.

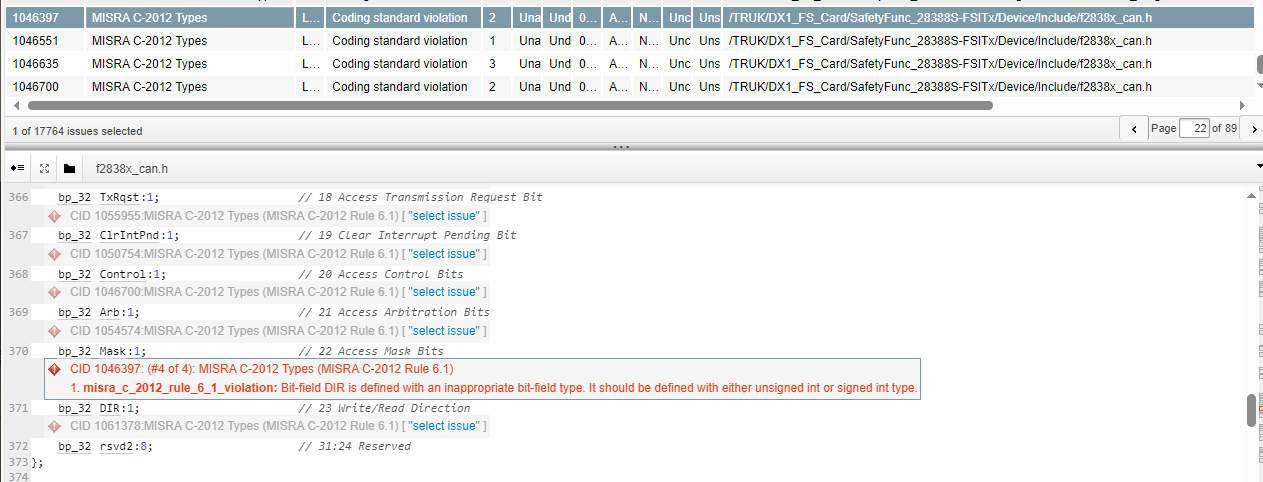
****

# MISRA C-2012 Types

**Category Required**

CID 1046397: (#4 of 4): MISRA C-2012 Types (MISRA C-2012 Rule 6.1)

**misra\_c\_2012\_rule\_6\_1\_violation:** Bit-field DIR is defined with an inappropriate bit-field type. It should be defined with either unsigned int or signed int type.

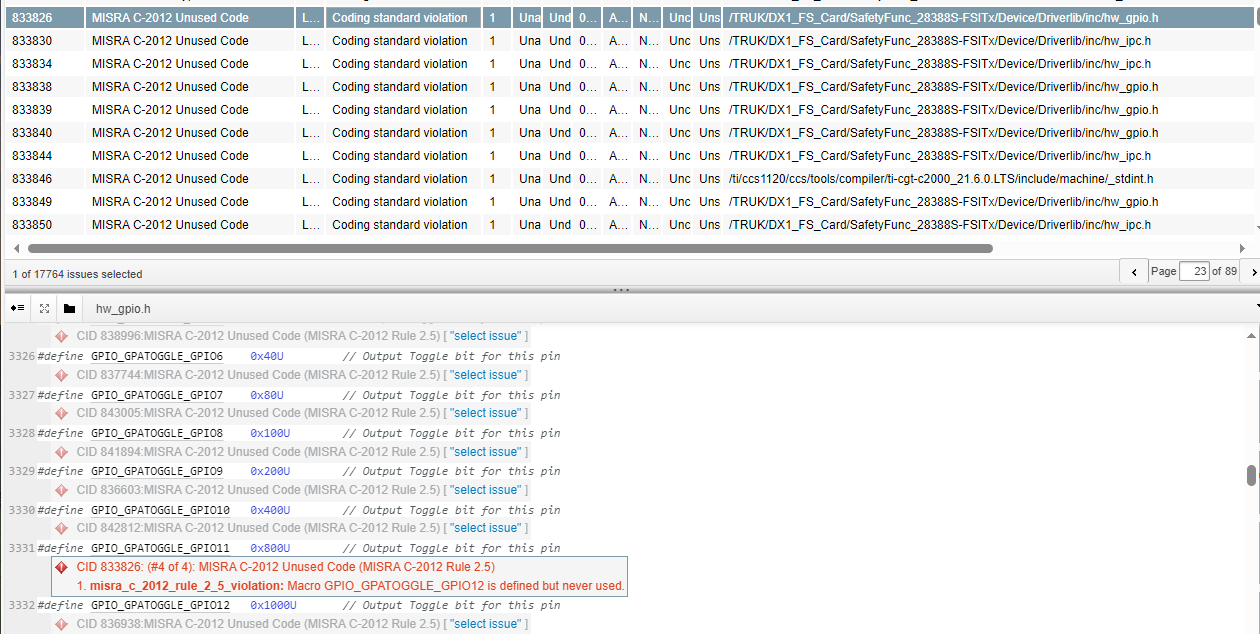
****

# MISRA C-2012 Unused Code

**Category Advisory**

CID 833826: (#4 of 4): MISRA C-2012 Unused Code (MISRA C-2012 Rule 2.5)

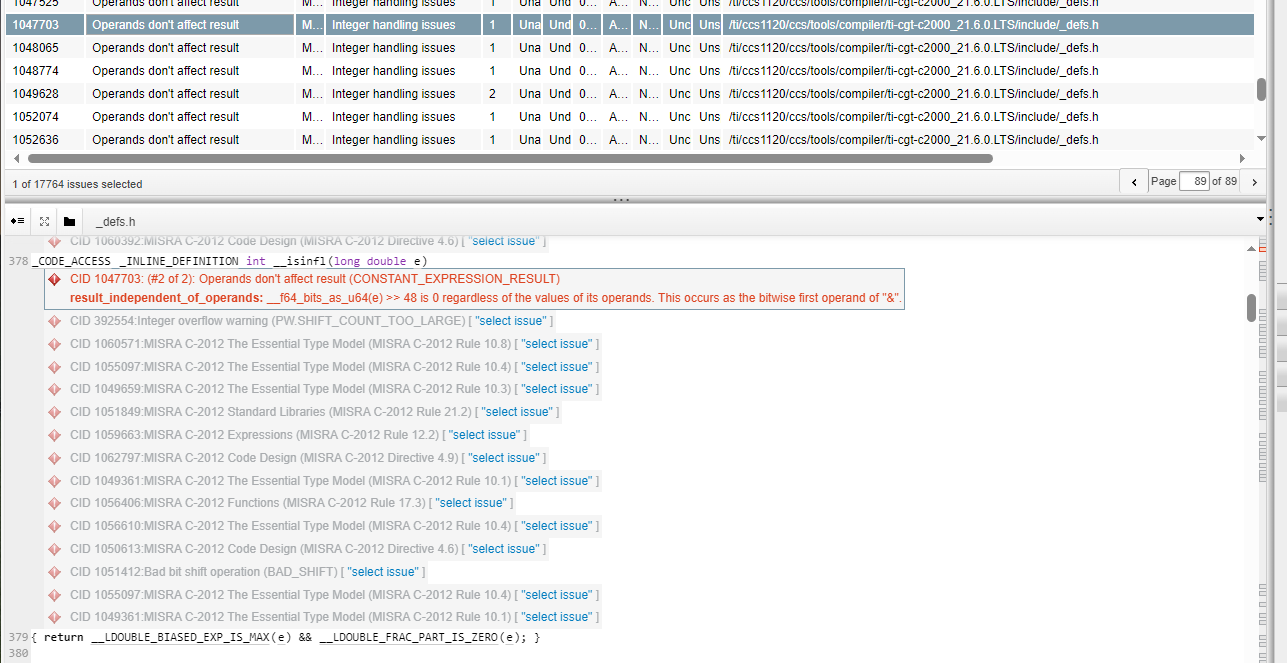
**misra\_c\_2012\_rule\_2\_5\_violation:** Macro GPIO\_GPATOGGLE\_GPIO12 is defined but never used

****

# Operands don't affect result

CID 1047703: (#2 of 2): Operands don't affect result (CONSTANT\_EXPRESSION\_RESULT)

**result\_independent\_of\_operands:** \_\_f64\_bits\_as\_u64(e) >> 48 is 0 regardless of the values of its operands. This occurs as the bitwise first operand of "&".

****