C++ static code analysis: Results of ~ and << operations on operands of underlying types unsigned char and unsigned short should immediately be cast to the operand's underlying type

2 minutes

When ~ and << are applied to small integer types (`unsigned char` or `unsigned short`), the operations are preceded by integral promotion, and the result may contain high-order bits which have not been anticipated.

Noncompliant Code Example

```
unsigned char port = 0x5aU;
unsigned char result_8;
unsigned short result_16;
unsigned short mode;
result_8 = (~port) >> 4; // Noncompliant; '~port' is 0xFFA5 on a 16-bit machine but 0xFFFFFFA5 on a 32-bit machine. Result is 0xFA for both, but 0x0A may have been expected.
result_16 = ((port << 4) & mode) >> 6; // Noncompliant; result_16 value depends on the implemented size of an int.
```

Compliant Solution

```
result_8 = ((unsigned char)(~port)) >> 4; // Compliant
result_16 = ((unsigned short)((unsigned short) port << 4) & mode) >> 6; // Compliant
```

See

- MISRA C 2004, 10.5 - If the bitwise operators ~ and << are applied to an operand of *underlying type* unsigned char or unsigned short, the result shall be immediately cast to the *underlying type* of the operand.
- MISRA C++ 2008, 5-0-10 - If the bitwise operators ~ and << are applied to an operand with an *underlying type* of unsigned char or unsigned short, the result shall be immediately cast to the *underlying type* of the operand.