GUIDELINE TO USE ASSERT/SC_ASSERT IN SCHEAP G4

CODING RULE

DATE: JAN/15/2021
CORESW – RH850 MODELING GROUP
RENESAS ELECTRONICS CORPORATION

© 2021 Renesas Electronics Corporation. All rights reserved.
RENESAS CONFIDENTIAL
BIG IDEAS FOR EVERY SPACE

AGENDA

© 2021 Renesas Electronics Corporation. All rights reserved.
RENESAS CONFIDENTIAL

BIG IDEAS FOR EVERY SPACE RENESAS

SOME FACTS ABOUT ASSERT/SC_ASSERT

- The assert macro is typically used to identify logic errors during **program development** [1].
- The assert macro can be turned off without modifying your source files by using a **NDEBUG** command-line option [1, 5].

<assert.h>

```
#ifdef NDEBUG

    #define assert(expression) ((void)0)

#else

    _ACRTIMP void __cdecl _wassert(
        _In_z_ wchar_t const* _Message,
        _In_z_ wchar_t const* _File,
        _In_   unsigned       _Line
        );

    #define assert(expression) (void)(                                                \
            (!!(expression)) ||                                                       \
            (_wassert(_CRT_WIDE(#expression), _CRT_WIDE(__FILE__), (unsigned)(__LINE__)), 0) \
        )

#endif
```

<systemc-2.3.1a_MOD_vc140_64bit\src\sysc\utils\sc_report.h>

```
#ifdef NDEBUG

#define sc_assert(expr) \
 ((void) 0)

#else

#define sc_assert(expr) \
 ((void)((expr) ? 0 :   \
     (SC_REPORT_FATAL( ::sc_core::SC_ID_ASSERTION_FAILED_, #expr ), 0)))

#endif // NDEBUG
```

- In one project the asserts literally caused a 3x slowdown. But they helped to uncover some really pesky bugs [2].
- Asserts are for catching the errors that can't possibly happen [2].

© 2021 Renesas Electronics Corporation. All rights reserved.

BACKGROUND

- The SLD group developed and tested the models in **Debug mode** – in which **NDEBUG** is NOT defined => assert is turned **on**. That's why the SLD used assert/sc_assert (for checking the validity of pointers).
- The RHM group is now developing and testing the models in **Release mode** – in which **NDEBUG** is defined => assert is turned **off**. So, using assert/sc_assert might be prone to bug. “If ...else“ or other ways should be used instead.
- Countermeasure for using assert/sc_assert:

1. Coder’s mindset should be correct about assert/sc_assert – this document explain the correct ways of using assert/sc_assert.
2. Add a checkpoint in PR checklist - Code Review Checklist (15.20) to force code reviewer to check for assert/sc_assert usage.
3. Recommend tester to run regression test in debug mode.

© 2021 Renesas Electronics Corporation. All rights reserved.

DO USE - INPUTS GENERATED BY YOUR OWN CODE

- **Assert/sc_assert** should only be used on inputs generated by **your own code**.
- **Exception** should be used on **external inputs**.
- Rule of thumb: verify **private** functions' arguments with **asserts/sc_assert**, and using **exceptions** for **public/protected** functions' arguments [3].

© 2021 Renesas Electronics Corporation. All rights reserved.

DO USE - INTERNAL INVARIANTS

- **Assert/sc_assert** should be used on ‘Internal Invariants’ [6].

Before assertions were available, many programmers used **comments** to indicate their assumptions concerning a program's behavior. For example, you might have written something like this to explain your assumption about an else clause in a multiway if-statement:

```
if (i % 3 == 0) {
    ...
} else if (i % 3 == 1) {
    ...
} else { // We know (i % 3 == 2)
    ...
}
```

You should now use an **assertion** whenever you would have written a **comment** that asserts an invariant. For example, you should rewrite the previous if-statement like this:

```
if (i % 3 == 0) {
   ...
} else if (i % 3 == 1) {
    ...
} else {
    assert i % 3 == 2 : i;
    ...
}
```

© 2021 Renesas Electronics Corporation. All rights reserved.

DO USE - CONTROL-FLOW INVARIANTS

- **Assert/sc_assert** should be used on ‘Control-Flow Invariants’ [6].

Place an assertion at any location you assume will not be reached. The assertions statement to use is:

```
assert false;
```

For example, suppose you have a method that looks like this:

```
void foo() {
    for (...) {
      if (...)
        return;
    }
    // Execution should never reach this point!!!
}
```

Replace the final comment so that the code now reads:

```
void foo() {
    for (...) {
      if (...)
        return;
    }
    assert false; // Execution should never reach this point!
}
```

© 2021 Renesas Electronics Corporation. All rights reserved.

DO USE - PRECONDITIONS

- **Assert/sc_assert** should be used on ‘Preconditions’ [6].

Preconditions — what must be true when a method is invoked.

Use an assertion to test a **nonpublic** method's precondition that you believe will be true no matter what a client does with the class. For example, an assertion is appropriate in the following "helper method" that is invoked by the previous method:

```
/**
 * Sets the refresh interval (which must correspond to a legal frame rate).
 *
 * @param  interval refresh interval in milliseconds.
*/
 private void setRefreshInterval(int interval) {
  // Confirm adherence to precondition in nonpublic method
  assert interval > 0 && interval <= 1000/MAX_REFRESH_RATE : interval;

  ... // Set the refresh interval
 }
```

© 2021 Renesas Electronics Corporation. All rights reserved.

DO USE - POSTCONDITIONS

- **Assert/sc_assert** should be used on ‘Postconditions’ [6].

Postconditions — what must be true after a method completes successfully.

You can test postcondition with assertions in both **public** and **nonpublic** methods. For example, the following public method uses an assert statement to check a post condition:

```
 /**
   * Returns a BigInteger whose value is (this-1 mod m).
   *
   * @param  m the modulus.
   * @return this-1 mod m.
   * @throws ArithmeticException  m <= 0, or this BigInteger
   *has no multiplicative inverse mod m (that is, this BigInteger
   *is not relatively prime to m).
   */
public BigInteger modInverse(BigInteger m) {
   if (m.signum <= 0)
     throw new ArithmeticException("Modulus not positive: " + m);
   ... // Do the computation
   assert this.multiply(result).mod(m).equals(ONE) : this;
   return result;
}
```

© 2021 Renesas Electronics Corporation. All rights reserved.

DO USE - CLASS INVARIANTS

- **Assert/sc_assert** should be used on ‘Class Invariants’ [6, 7].

Class invariants — what must be true about each instance of a class..

```
// Returns true if this tree is properly balanced
private boolean balanced() {
  ...
}
```

Assert at the end of any method of class:

```
assert balanced();
```

© 2021 Renesas Electronics Corporation. All rights reserved.

DONT USE - PRECONDITIONS ON PUBLIC METHODS

- **Assert/sc_assert** should **NOT** be used on ‘Preconditions on Public Methods’ [6].

By convention, preconditions on **public** methods are enforced by explicit checks that throw particular, specified exceptions. For example:

```
/**
  * Sets the refresh rate.
  *
  * @param  rate refresh rate, in frames per second.
  * @throws IllegalArgumentException if rate <= 0 or
  * rate > MAX_REFRESH_RATE.
*/
public void setRefreshRate(int rate) {
  // Enforce specified precondition in public method
  if (rate <= 0 || rate > MAX_REFRESH_RATE)
    throw new IllegalArgumentException("Illegal rate: " + rate);
    setRefreshInterval(1000/rate);
  }
```

© 2021 Renesas Electronics Corporation. All rights reserved.

SOME ‘PERSONAL’ IDEAS – CAN BE TRUE OR FALSE

- Assertions are only used by lazy programmers who don't want to code up error handling. If you know an error is possible, handle it. If it's not possible, then there is no reason to assert [2, 4].

© 2021 Renesas Electronics Corporation. All rights reserved.

REFERENCES

1. assert Macro, _assert, _wassert
2. Why should I use asserts?
3. When to use assertions and when to use exceptions?
4. When should assert() be used?
5. Assertion (software development)
6. Programming With Assertions
7. Assert use cases

© 2021 Renesas Electronics Corporation. All rights reserved.

Renesas.com

© 2021 Renesas Electronics Corporation. All rights reserved.
RENESAS CONFIDENTIAL