



Application Note

QEP/C PC-Lint™ Compliance

Document Version 1.6
October 2005

Quantum **L^eaps™**, LLC

www.quantum-leaps.com

Copyright © 2002-2005 Quantum **L^eaps**, LLC. All Rights Reserved.

1 PC-Lint™ Source Code Analysis

QEP/C 3 source code is “lint-free”. The compliance was checked against PC-lint™/FlexLint™, which is perhaps the most advanced and widely used static analysis tool from Gimpel Software (<http://www.gimpel.com>). The standard QEP/C distribution includes the <qpc_3>/qep/lint/ subdirectory, which contains the PC-lint™ configuration files used for static analysis of the QEP/C source code.

```
1: au-misra.lnt // MISRA-C:1998 rules [MISRA 98]
2: au-ds.lnt // Dan Saks recommendations
3: co.lnt // generic C/C++ compiler
4:
5: options.lnt -si4 -ss2 -sp4
6: -i.;..\source;..\..\include;..\..\ports\lint
```

Listing 1 std.lnt file PC-lint options file

Listing 1 shows the top-level options for PC-lint™ stored by convention in the std.lnt file. These options include: Motor Industry Software Reliability Association [MISRA 98, 04] rule checker (Listing 1, line 1), Dan Saks recommendation checker (line 2), and a generic C/C++ compiler rules (line 3). The size options for PC-lint™ (Listing 1, line 5) are: size of integer—4 bytes, size of short—2 bytes and size of pointer—4 bytes. The include options (line 6) specify the relative paths to the QEP/C include directories.

```
1: // Assertions
2: -emacro(970, Q_DEFINE_THIS_FILE) // MISRA rule 13
3: -emacro(971, Q_DEFINE_THIS_FILE) // MISRA rule 14
4: -emacro(970, Q_DEFINE_THIS_MODULE) // MISRA rule 13
5: -emacro(971, Q_DEFINE_THIS_MODULE) // MISRA rule 14
6: -emacro(960, Q_ASSERT) // MISRA rule 59
7:
8: // QEP
9: -emacro(960, QEP_ENTER_AND_REC_) // MISRA rule 59
10: -emacro(960, QEP_EXIT_AND_REC_) // MISRA rule 59
11: -emacro(717, QFsm_ctor_) // do ... while(0)
12: -emacro(717, Q_TRAN) // do ... while(0)
13: -emacro(717, Q_TRAN_STA) // do ... while(0)
14: -emacro(956, Q_TRAN_STA) // Non const, non volatile static or external
15:
16: // QS
17: -emacro(506, QS_BEGIN_) // Constant value Boolean [MISRA Rule 52]
18: -emacro(774, QS_BEGIN_) // Boolean within 'if' always evaluates to False
19: -emacro(572, QS_BEGIN_) // MISRA Rule 38, excessive shift value
20: -emacro(778, QS_BEGIN_) // Constant expression evaluates to 0
21: -emacro(912, QS_BEGIN_) // Implicit binary conversion from int to unsigned int
22: -emacro(960, QS_END_) // MISRA Rule 59, left brace expected
23: -emacro(725, QS_END_) // Expected positive indentation
24: -emacro(923, QS_OBJ_) // MISRA Rule 45, cast from pointer to unsigned long
25: -emacro(923, QS_FUN_) // MISRA Rule 45, cast from pointer to unsigned long
26: -DQ_SPY // Enable the Q-Spy instrumentation
27:
28: // Miscellaneous
29: -e546 // Suspicious use of &
```

Listing 2 options.lnt PC-lint options file

Listing 2 shows the project-dependent options for QEP/C stored by convention in the options.lnt file. This file defines options that suppress PC-lint™ warnings in certain contexts. The following bullet-items explain what these options mean and why are they used.

- Listing 2 lines 2 and 4: suppresses PC-lint™ elective note 970 in macros Q_DEFINE_THIS_FILE and Q_DEFINE_THIS_MODULE. Note 970 is elected by the MISRA compliance checker to verify MISRA rule 13, which requires that the basic type char is never used. However, the basic type char represents in these cases a character string of the file name for assertion checking. Therefore, in this particular case MISRA rule 13 is not applicable.
- Listing 2 lines 3 and 5: suppresses PC-lint™ elective note 971 in macros Q_DEFINE_THIS_FILE and Q_DEFINE_THIS_MODULE. Note 971 is elected by the MISRA compliance checker to verify MISRA rule 14, which requires that type char be always declared as signed or unsigned. However, the type char represents in this case a character string of the file name for assertion checking. Therefore, in this particular case MISRA rule 14 is not applicable.
- Listing 2 lines 6, 9, and 10: suppresses PC-lint™ elective note 960 in macros Q_ASSERT, QEP_ENTER_AND_REC_, and QEP_ENTER_AND_REC_. Note 960 is elected by the MISRA compliance checker to verify MISRA rule 59, which requires that the statement following if () .. else be always enclosed in braces. Macros Q_ASSERT, QEP_ENTER_AND_REC_, and QEP_ENTER_AND_REC_ intentionally use an explicit empty statement ((void)0) following the else to avoid the dangling-else problem.
- Listing 2 lines 11-13: suppressing PC-lint™ informational message 717 "do ... while(0)" for macros QFsm_ctor(), Q_TRAN(), and Q_TRAN_STA(). These macros intentionally use do {...} while(0) to group multiple statements together and/or delimit the scope of temporary variables so that these function-like macros can be terminated with the semicolon in a syntactically-correct way.
- Listing 2 line 14: Suppress the elective note 956 "Non const, non volatile static or external variable 'Symbol'" in the macro Q_TRAN_STA(). This check has been advocated by programmers whose applications are multithreaded. Software that contains modifiable data of static duration is often non-reentrant. Static transition indeed contains such static data (the static transition object QTran_), but the code has been carefully designed to avoid concurrency issues, even if a given state machine class gets instantiated multiple times, and each instance executes concurrently with the others. (Please also refer to Section 8.8.1 in "QEP/C Programmer's Manual" [QL 05b])
- Listing 2 lines 17-25: suppress the elective notes for the Q-Spy instrumentation macros in both cases, when the instrumentation is enabled and when it is disabled. When disabled, some Q-Spy macros intentionally use if (0) .. constructs to exclude the instrumentation that might otherwise generate some machine code.
- Listing 2 line 26: enables the Q_SPY instrumentation, which is the tougher case to pass. The code has also been "lintered" with Q_SPY disabled.
- Listing 2 line 29: Suppress a warning message about taking the address of a function name (&QCalc_on). In QP/C, explicitly taking addresses of functions is considered a good programming style because it forces the programmer to distinguish between the pointer-to-function and the function itself. Also, this programming style is more consistent with C++ syntax of pointers-to-functions and pointers-to-member-functions.

The actual “linting” of the source code has been performed with the latest-available version of PC-lint™ (version 8.00s as of this writing) and the latest configuration files, specifically the latest update for MISRA rules support `au-misra.lnt`. Here is the clean lint output (saved in the `_LINT_.TMP` file):

```
--- Module: ..\source\qep.c
--- Module: ..\source\qfsm_dis.c
--- Module: ..\source\qfsm_ini.c
--- Module: ..\source\qhsm_dis.c
--- Module: ..\source\qhsm_in.c
--- Module: ..\source\qhsm_ini.c
--- Module: ..\source\qhsm_top.c
--- Module: ..\source\qhsm_tra.c
```

2 Related Documents

Please also see: “Application Note: QEP/C MISRA Compliance Matrix”, http://www.quantum-leaps.com/doc/AN_QEP_C_MISRA.pdf.

3 References

[C99]	ISO/IEC 9899 C Approved Standards, http://www.open-std.org/jtc1/sc22/open/n2794
[Gimpel 04]	Gimpel Software, “MISRA-C (1998) checking provided by PC-lint/FlexeLint”, http://www.gimpel.com/html/misra.htm
[MISRA 98]	Motor Industry Software Reliability Association (MISRA), MISRA Limited, MISRA-C:1998 Guidelines for the Use of the C Language in Vehicle Based Software, April 1998, ISBN 0-9524156-9-0. See also http://www.misra.org.uk
[MISRA 04]	Motor Industry Software Reliability Association (MISRA), MISRA Limited, MISRA-C:2004 Guidelines for the Use of the C Language in Critical Systems, October 2004, ISBN 0-9524156-2-3 (paperback), ISBN 0-9524156-4-X (PDF). See also http://www.misra.org.uk
[QL 05a]	Quantum Leaps, LLC, “Application Note: QEP/C MISRA Compliance Matrix”, http://www.quantum-leaps.com/doc/AN_QEP_C_MISRA.pdf
[QL 05b]	Quantum Leaps, LLC, “QEP/C Programmer’s Manual”, http://www.quantum-leaps.com/doc/QEP_C_Manual.pdf