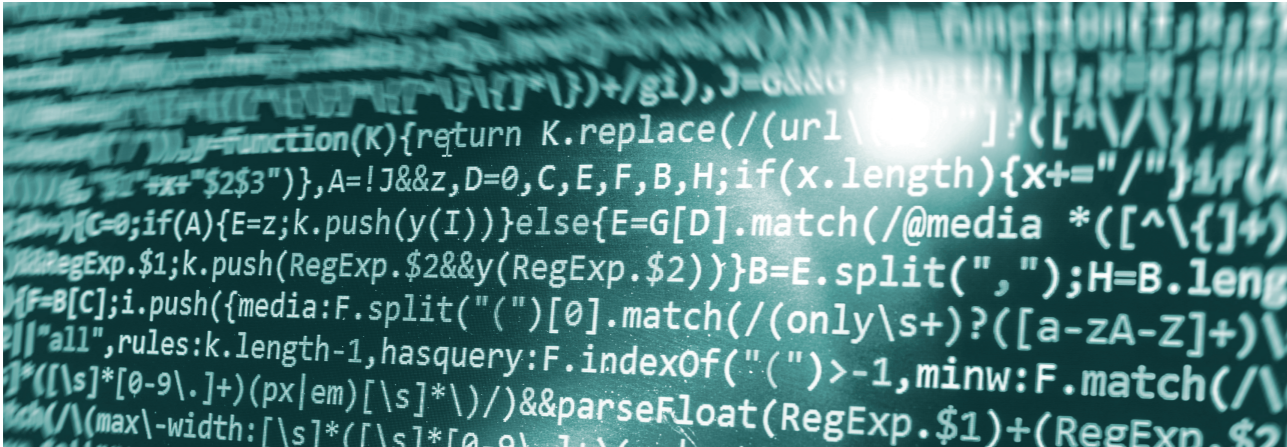




Static code analysis with ECLAIR
Prevent failures, check coding rules, and calculate metrics

> Product Brief

eclair



Static code analysis

During the last years software code quality became increasingly important. Static code analysis analyses source code without executing the code to find potential bugs, violation of coding rules, and security vulnerabilities. So static code analysis improves safety and security of the source code. A There is no effort for test case specification for static code analysis; this is a big advantage. By using static code analysis tools, problems can be addressed early in the development process, where the cost to fix them is still low.

Static code analysis with ECLAIR

ECLAIR is a powerful tool for static code analysis of programs written in C/C++. ECLAIR works on the desktop to find critical defects while software is being coded, in the context of the build environment and compiler. The extreme flexibility of ECLAIR allows it to be tailored for any software verification problem and to any software development process. The coding rules are specified in a very high-level language using a powerful library of services, something that greatly facilitates the adoption of project-specific coding rules.

Run-time errors

ECLAIR detects possible Run-time Errors (RTE), which could lead to crashes, wrong behavior, and security vulnerabilities, e.g. NULL pointer dereference, division by zero, index out of bounds, etc.

```
1 signed long divide_100(signed long a1, signed char b1)
2 {
3
4     return 100 / (a1 - (signed long) b1 - 2);
5 }
6
7 int main(void)
8 {
9     divide_100(5,2);
10    divide_100(5,3);
11    return 0;
12 }
```

The call of divide_100() in line 10 causes a division by zero, whereas the call in line 9 does not.

MISRA and other coding guidelines

ECLAIR checks comprehensively the rules from BARR-C:2018, MISRA-C:2004, MISRA C:2012, and MISRA C++:2008. Support for more coding guidelines is in preparation.

```
14
15     uint32_t v1 = 0;
16     s1 = sizeof(v1++);
```

≡ MC3.R13.6 unevaluated '++' postfix increment operator

≡ MC3.R13.6 'sizeof' expression trait

```
42 int func(int a)
43 {
```

≡ NC3.7.1.e the identifier for parameter 'a' has less than 3 characters

The '++' operator will not be evaluated.
This violates rule 13.6 of MISRA C:2012

The name of the parameter is too short.
This violates rule 7.1e of BARR-C:2018

Metrics

ECLAIR calculates metrics of the «Herstellerinitiative Software» (HIS).

```
8 typedef int s32;
9
10 s32 func9(s32 p1, s32 p2, s32 p3, s32 p4, s32 p5, s32 p6, s32 p7, s32 p8, s32 p9)
11 {
12     return p1+p2+p3+p4+p5+p6+p7+p8+p9;
13 }
```

≡ B.THRESHOLD for function 'func9(s32, s32, s32, s32, s32, s32, s32, s32, s32)', the metric 'HIS.PARAM' (Number of function parameters) has value 9 > 7 (the limit)

The limit for the number of parameters of a function is set to 7. This is violated by func9().

Supported compilers

ECLAIR supports popular native compilers and cross compilers, see list below:

Arm	Green Hills	MPLAB	Tasking
CodeWarrior	HighTec	Microsoft	Texas Instruments
Cosmic	IAR	QNX	Wind River
CrossWorks	Intel	Renesas	Clang/LLVM
GCC	Keil	Softune	...

Development environments

ECLAIR can be used stand alone or integrated into the following development environments:

Eclipse for C/C++	Microsoft Visual Studio
IAR Embedded Workbench	Texas Instruments Code Composer Studio
Keil µVision	...

Supported operating systems

ECLAIR runs on Windows, Unix/Linux, and OS X

More information on ECLAIR can be found at www.hitex.com/eclair

Hitex is providing expertise, trainings, webinars, consulting, services & tools on software quality and software test - please check www.hitex.com

Hitex Head Office, Germany

Hitex GmbH
Greschbachstraße 12
76229 Karlsruhe
Germany

Phone: +49-721-9628-0
Fax: +49-721-9628-149
Email: info@hitex.de

Hitex UK

Hitex (UK) Ltd
Millburn Hill Road
University of Warwick Science Park
Coventry CV4 7HS
United Kingdom

Phone: +44-24-7669-2066
Fax: +44-24-7669-2131
Email: info@hitex.co.uk