

KF32 系列

工具链 misra 规则检查使用说明 V1.0

2022 年 05 月

目录

目录	2
修改记录	3
1. 绪论	4
2. 工具集成与功能选项说明	4
2.1 虚拟使能开关 “MISRA-2012 ENABLE”	5
2.2 信息存在下配置库方法检测--CHECK-LIBRARY	5
2.3 配置检测--CHECK-CONFIG	5
2.4 问题调试分析--BUG-HUNTING	5
2.5 部分特性使能选项--ENABLE=<ID>	5
2.6 选项语言标准-STD=<ID>	6
2.7 显示格式模板--TEMPLATE='<TEXT>'	6
2.8 错误信息位置 --TEMPLATE-LOCATION='<TEXT>'	6
2.9 配置输出路径--CPPCHECK-BUILD-DIR=<DIR>	7
2.10 加载函数信息的配置--LIBRARY=<CFG>	7
2.11 其他相关选项	7
2.11.1 附加脚本	7
2.11.2 文件或文件夹排除与匹配	8
2.11.3 源文件包含头文件路径	8
2.11.4 特殊控制	8
2.11.5 输出控制选项	9
2.11.6 禁止匹配	9
2.11.7 更多选项	10
3. 基于项目与统计功能介绍	10
3.1 示例控制台输出显示	10
3.2 集成局限性	11
3.3 整体分析与查看	11
3.3.1 这里参考输入	11
3.3.2 示例项目输出如下:	11
3.3.3 输出文件结构示意	12
3.3.4 文件结果报告查看	13

修改记录

序号	日期	版本	变更及说明	其他
1	2022-05-13	V1.0	初稿	
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

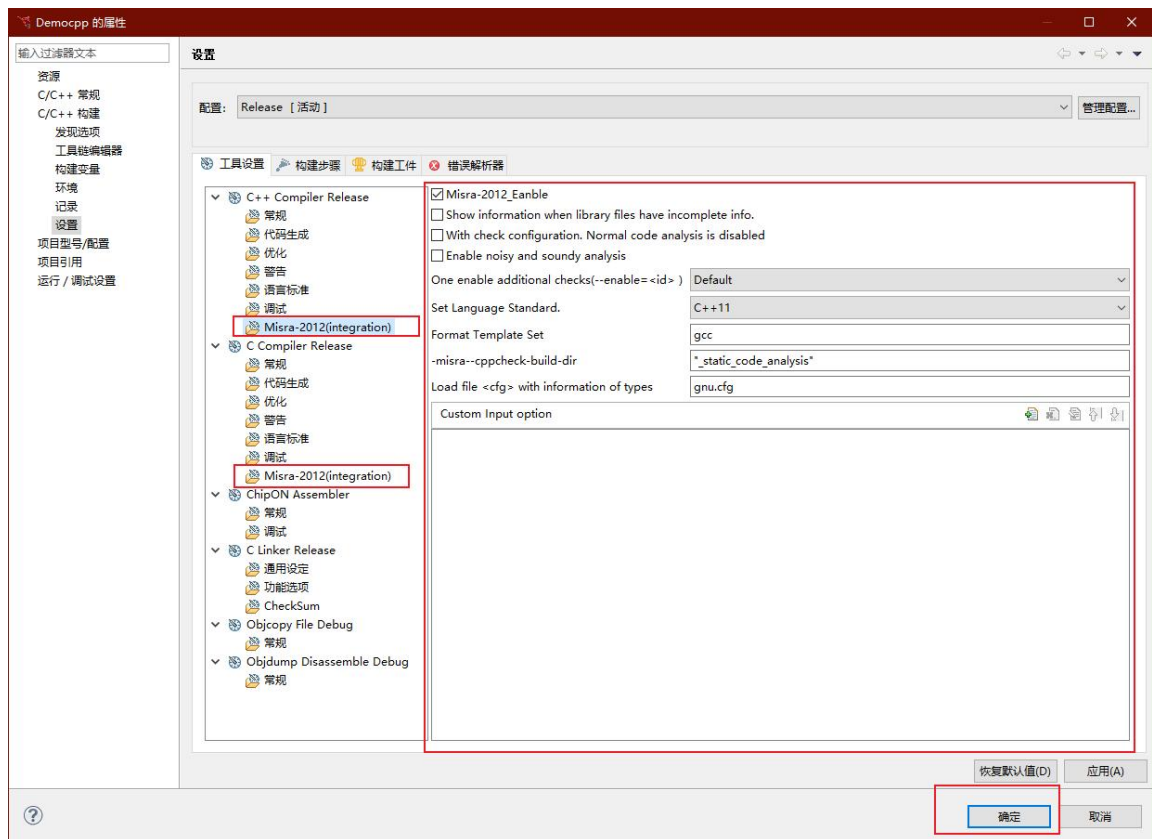
1. 绪论

misra 为工业标准的 C 编程规范，当前 ChipON 工具集成第三方功能程序 cppcheck，对应其当前标准为 misra-2012。

默认工具不开启该功能检测，当配置使能后，该集成方案将检测结果输出到开发环境下。功能控制选项寄生在编译器选项交互选项中。

2. 工具集成与功能选项说明

如下图所示，该配置与项目属性进行绑定，在项目属性的工具选择配置页面，选择使能并可选项配置或添加自定义选项。



该功能选项与 C 编译器或 C++编译器进行绑定，即与编译工具联合进行源文件的单一文件编译分析模式。这些选项非编译器选项，由后台程序进行区分与组装实现代码的静态分析。同时基于 makefile 构建系统的增量构建模型下的静态代码分析。

2.1 虚拟使能开关 “Misra-2012 Enable”

使能基于编译过程单个文件静态分析与 misra 规则匹配功能的开关。

2.2 信息存在下配置库方法检测--check-library

当库文件的信息不完整时，显示信息消息。

2.3 配置检测--check-config

检查 cppcheck 配置。 此标志禁用常规代码分析。

2.4 问题调试分析--bug-hunting

启用噪声和声音分析。 正常的 Cppcheck 分析是关闭的。

2.5 部分特性使能选项--enable=<id>

使额外的检查。如果用逗号分隔，可以给出多个 id。 参见--std.

可用的 id 为：

- * all 所有

使能所有检查。建议只在扫描整个程序时使用，因为这将启用 unusedFunction。

- * warning 警告

启用警告消息

- * style 样式

启用所有编码风格检查。 所有严重性为“样式”、“警告”、“性能”和“可移植性”的消息都被启用。

- * performance 性能

使性能信息

- * portability 可移植性

使可移植性的消息

- *information 信息

使信息消息

- * unusedFunction 未使用函数

检查未使用的功能。 建议只在扫描整个程序时启用此功能。

- * missingInclude 缺失头文件

如果缺少包含则发出警告。 有关详细信息，可查看--check-config。

2.6 选项语言标准-std=<id>

可用的选项有:

- * c89 C 代码与 C89 兼容
- * c99 C 代码与 C99 兼容 (工具默认)
- * c11 C 代码是 C11 兼容的(默认)

- * c++ 03 c++代码与 c++ 03 兼容
- * c + + 11 c++代码是 c++ 11 兼容的 (工具默认)
- * c + + 14 c++代码与 c++ 14 兼容
- * c + + 17 c++代码是 c++ 17 兼容的
- * c + + 20 c++代码是 c++ 20 兼容的 (默认)

2.7 显示格式模板--template='<text>'

格式化错误消息。 可用字段:

- {file} 文件名称
- {line} 行号
- {column} 列号
- {callstack} 显示一个调用堆栈。 例: [file.c: 1] - > [file.c: 100)
- {inconclusive:text} 如果警告是不确定的, 则返回文本
- {severity} 严重程度, 典型如 error 或者 warning
- {message} 警告消息
- {id } 警告标识
- {cwe} cwe id(常见缺陷枚举)
- {code} 显示原始的代码
- \t 插入制表符
- \n 插入换行符
- \r 插入回车

示例格式:

```
' {file}: {line}, {severity}, {id}, {message}'    或  
' {file}({line}):({severity}) {message}'       或  
' {callstack} {message}'
```

预定义模板: gcc(默认)、cppcheck1(原始默认)、vs、edit。

2.8 错误信息位置 --template-location='<text>'

格式错误消息位置。 如果没有提供, 则不会显示额外的位置信息。

可用字段:

{file}	文件名称
{line}	行号
{column}	列号
{info}	位置信息
{code}	显示原始的代码
\t	插入制表符
\n	插入换行符
\r	插入回车

举例格式(如 gcc): ' {file}:{line}:{column}: note: {info}\n{code}'

2.9 配置输出路径--cppcheck-build-dir=<dir>

Cppcheck 工作文件夹。 优点:

- *全程序分析
- *速度分析; 如果文件的散列没有改变, Cppcheck 将重用结果。
- *一些有用的调试信息, 例如用于执行 clang/clang-tidy/addons 的命令。

2.10 加载函数信息的配置--library=<cfg>

加载包含类型和函数信息的<cfg>文件。 有了这些信息, Cppcheck 就能更好地理解您的代码, 从而获得更好的结果。 与 Cppcheck 一起分发的 std.cfg 文件会自动加载。 有关库文件的更多信息, 请阅读 Cppcheck 手册。

2.11 其他相关选项

提供其他选项列表, 可以根据需要进行添加的增添处理, 部分选项仅基于文件或项目的扫描多文件情形适用。

2.11.1 附加脚本

--addon=<addon> 集成为 misra 规则检测

--addon-python=<python interpreter> 你可以在 addon json 文件中或通过命令行选项指定 python 解释器。 如果不存在, Cppcheck 将首先尝试 “python3”, 然后尝试 “python”。 ChipON 工具分发提供最小运行需要的 python.exe python38.dll, 其版本为 3.8.8. 另外附加 xml 到 html 解析需要的 pygments。

2.11.2 文件或文件夹排除与匹配

`--config-exclude=<dir>` 需要排除配置检查的路径(前缀)。定义在头文件(而不是源文件)中匹配前缀的预处理器配置将不会被考虑计算。

`--config-excludes-file=<file>` 包含 `config-exclude` 列表的文件

`-i <dir or file>` 给出一个源文件或源文件目录以排除在检查之外。这只适用于源文件,因此源文件包含的头文件是不匹配的。目录名与路径的所有部分匹配。

`--file-filter=<str>` 可以多次使用,示例:`--file-filter=*bar.cpp` 只分析以 `bar.cpp` 结尾的文件。

`--file-list=<file>` 指定要检测文本文件的文件。每行添加一个文件名。当文件为 '-' 时,将从标准输入中读取文件列表。

2.11.3 源文件包含头文件路径

`-I <dir>` 给出搜索包含文件的路径。给出几个 `-I` 参数来给出几个路径。第一个给定的路径首先搜索所包含的头文件。如果路径相对于源文件,则不需要这样做。使用 IDE 的配置,默认搜索文件路径使用编译配置的文件路径。

`--includes-file=<file>` 指定目录路径以搜索文本文件中包含的头文件。每行添加一个包含路径。第一个给定的路径首先搜索所包含的头文件。如果路径相对于源文件,则不需要这样做。

`--include=<file>` 强制在选中的文件之前包含一个文件。

2.11.4 特殊控制

`--inconclusive` 允许 Cppcheck 报告,即使分析是不确定的。这个选项存在误报。每个结果在你知道它是好是坏之前都必须仔细调查。

`--inline-suppr` 支持内联排除。可以通过在要抑制的警告之前的行中放置一个或多个注释来使用它们,比如: `'// cppcheck-suppress warning id'`。

2.11.5 输出控制选项

-E 在 stdout 上打印预处理器输出，不做任何进一步处理。

--output-file=<file> 将结果写入文件，而不是写入标准错误。如配置 -xml 选项后后期可生成分析的 html 文件。

--xml 将结果以 xml 格式写入错误流(stderr)。

--dump 转储每个翻译单元的 xml 数据。转储文件的扩展名为 .dump，包含 ast、tokenlist、symbol database、valueflow。

--errorlist 以 XML 格式打印所有错误消息的列表。

--exitcode-suppressions=<file> 当应该显示某些消息但不应导致非零退出码时使用。

2.11.6 禁止匹配

--suppress=<spec> 禁止匹配<spec>的警告。格式为：[error id]:[filename]:[line]，其中文件与行号是可选项。如果[error id]是通配符'*'，则所有错误 id 匹配。

--suppressions-list=<file> 禁止文件中列出的警告。每个 suppression 的格式与上面的<spec>相同。

--suppress-xml=<file.xml> 屏蔽 xml 文件中列出的警告。XML 文件应遵循章节中指定的特定格式。XML 格式是可扩展的，将来可能使用更多的属性进行扩展。

如：

```
<?xml version="1.0"?>
<suppressions>
  <suppress>
    <id>uninitvar</id>
    <fileName>src/file1.c</fileName>
    <lineNumber>10</lineNumber>
    <symbolName>var</symbolName>
  </suppress>
</suppressions>
```

--inline-suppr 内嵌通过源码注释的方法屏蔽，如

```
// cppcheck-suppress misra-c2012-11.4
```

```
// cppcheck-suppress comparePointers
```

2.11.7 更多选项

`--max-configs=<limit>` 在跳过文件之前要检入的最大配置数。默认设置是“12”。如果和“`--force`”一起使用，最后一个选项是有效的。

`-f, --force` 强制检查文件中的所有配置。 如果与‘`--max-configs=`’一起使用，则最后一个选项是有效的。

`-q, --quiet` 不要显示进度报告。

`-D<ID>` 定义预处理器的符号。 除非使用了`--max-configs`或`--force`，否则 Cppcheck 只会在使用`-D`时检查给定的配置。 例如：“`-DDEBUG = 1 -D__cplusplus`”。

`-U<ID>` 未定义预处理器的符号。 使用`-U`显式隐藏某些`#ifdef <ID>`代码路径。 例如：“`-UDEBUG`”

`-j <jobs>` 启动<jobs>线程以同时执行检查。

`--max-ctu-depth=N` 整个程序分析的深度最大。 缺省值为 2。 较大的值意味着可以发现更多的错误，但也意味着分析将变慢。

3. 基于项目与统计功能介绍

3.1 示例控制台输出显示

```
Building file: ../_config/startup.c
..\_config\startup.c:36:15: warning: Comparing pointers that point to different
objects [comparePointers]
    while( begin <  end) {
                ^
..\_config\startup.c:9:21: note: Variable declared here.
extern unsigned int __data_start__;
Building file: ../main.cpp
..\main.cpp:13:24: warning: A compatible declaration shall be visible when an
object or function with external linkage is defined [misra-c2012-8.4]
volatile unsigned char i=3;
```

```
^
..\main.cpp:18:6: warning: Both operands of an operator in which the usual
arithmetic conversions are performed shall have the same essential type category
[misra-c2012-10.4]
    if(i==3)
    ^
```

3.2 集成局限性

集成使用基于编译系统的增量调用执行检测，再次编译时因不触发原有文件的构建，因此 cppcheck 将不执行，因此无法显示以往的规则检查结果信息。

3.3 整体分析与查看

如局限部分描述，默认配置分析结果存放与固定目录，因此文件的解析结果将根据变动重新启动分析，未变动时读取原始的检测信息。

集成开发环境构建系统提供如下的接口，使能编译前调用命令行或编译后调用命令行从而实现更多的控制，通过&& 或 ； 间隔支持多条命令的添加，如制作库，elf 文件信息处理等。

3.3.1 这里参考输入

```
@cppcheck --platform=kungfu32_r1.xml --template=gcc
--enable=information --cppcheck-build-dir="_static_code_analysis"
--addon="misra.json" --xml ../
--output-file=_static_code_analysis/build_log.xml && python "C:/Program Files
(x86)/ChipON IDE/KungFu32/ChipONCC32/common/cppcheck-htmlreport"
--file=_static_code_analysis/build_log.xml --report-dir=._static_code_html
--source-dir=.
```

注：其实字符@控制命令行内容不输出显示在控制中。可使用--quiet 屏蔽过程进度信息的输出，如 1/5 files checked 2% done

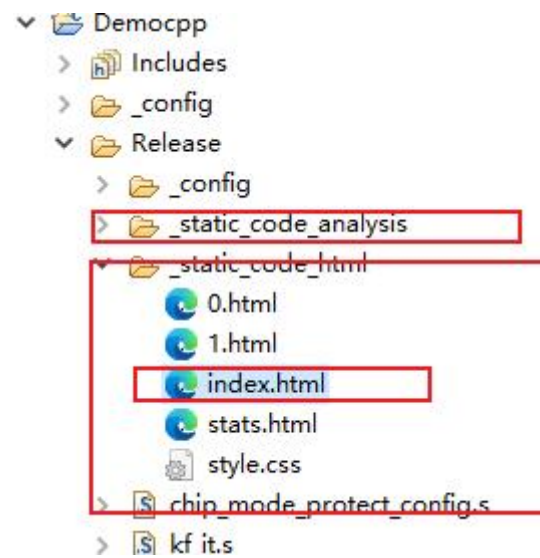
3.3.2 示例项目输出如下：

```
Checking ..\_config\chip_mode_protect_config.c ...
1/5 files checked 2% done
Checking ..\_config\startup.c ...
```

```
2/5 files checked 7% done
Checking ..\_config\vector.c ...
3/5 files checked 90% done
Checking ..\kf_it.c ...
4/5 files checked 98% done
Checking ..\main.cpp ...
5/5 files checked 100% done
Parsing xml report.
Creating ./_static_code_html directory
Processing errors
    ..\_config\startup.c
    ..\main.cpp
Creating index.html
Creating style.css file
Creating stats.html (statistics)
```

Open './_static_code_html/index.html' to see the results.

3.3.3 输出文件结构示意



3.3.4 文件结果报告查看

