



**Documentation**

**Performing QAC-tests with GenericScons**

**ITK Engineering AG**   
www.itk-engineering.de  
Im Speyerer Tal 6  
76761 Rülzheim  
Tel: +49-7272-7703-0  
Fax: +49-7272-7703-100

1. General

|  |  |
| --- | --- |
| Document state: | draft |
| Date: | 20 February 2015 |
| Author: | André Fischer |
| Projekttitel: | GenericScons |

Change history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Date | Vers. | Stelle | Grund | Kürzel |
| 1 | 13.06.2014 | 0.1.0 |  | Erstfassung | AnFi |
| 2 | 05.09.2014 | 0.1.1 |  | Update | AnFi |
| 3 | 12.09.2014 | 0.2.0 |  | Update | AnFi |
| 4 | 25.09.2014 | 0.2.1 |  | Update | AnFi |
| 5 | 06.10.2014 | 0.2.2 |  | Correction in 2.3.2 | AnFi |
| 6 | 16.10.2014 | 0.3.0 |  | Add information about new feature | AnFi |
| 7 | 31.10.2014 | 0.3.1 |  | Add information about copy behavior of GenericScons | AnFi |
| 8 | 10.11.2014 | 0.4.0 |  | Update paragraph 2.3.3.  Add Paragraphs 3.1.3, 3.2.2  Add Figures Table | AnFi |
| 9 | 10.02.2015 | 0.5.0 |  | Update on new GenericScons options for QAC | AnFi |
| 10 | 20.02.2015 | 0.6.0 |  | Remove minor bug in Section 2.3.2 | AnFi |
| 11 | 26.08.2015 | 0.7.0 |  | Add section 3.1.5 Update section 3.1.3 & 4.2 Remove section 5.3 | Vishal Singh |

1. Terms, abbreviations and definitions
   1. Terms
   2. Abbreviations
   3. Definitions

„xxx“ typically stands for a component name, e.g. “cipp”.

1. Contents

[I General 2](#_Toc428527025)

[II Terms, abbreviations and definitions 2](#_Toc428527026)

[II.1 Terms 2](#_Toc428527027)

[II.2 Abbreviations 2](#_Toc428527028)

[II.3 Definitions 2](#_Toc428527029)

[III Contents 3](#_Toc428527030)

[IV Figures 4](#_Toc428527031)

[1. Introduction 5](#_Toc428527032)

[2. Prerequisites 5](#_Toc428527033)

[2.1 QAC installation 5](#_Toc428527034)

[2.1.1 QAC 5](#_Toc428527035)

[2.1.2 QACPP 5](#_Toc428527036)

[2.2 Shares 6](#_Toc428527037)

[2.3 Adjusting GenericScons 6](#_Toc428527038)

[2.3.1 Activation of QAC 6](#_Toc428527039)

[2.3.2 (Optional) Modification of 01\_Source\_Code\algo\xxx\algo\_lib\_file\_list.scfg and 01\_Source\_Code\algo\xxx\algo\_config.scfg 6](#_Toc428527040)

[2.3.3 Integration of QAC-tests into visual studio solution 8](#_Toc428527041)

[3. How to QAC within GenericScons 10](#_Toc428527042)

[3.1 Via Command Line 11](#_Toc428527043)

[3.1.1 Generation of QAC-project 11](#_Toc428527044)

[3.1.2 Generation of Compliance Report 11](#_Toc428527045)

[3.1.3 Analysis of single source files 11](#_Toc428527046)

[3.1.4 Additional command line options 12](#_Toc428527047)

[3.1.5 QAC option –j N – Multi-thread build 12](#_Toc428527048)

[3.2 Via Visual Studio 12](#_Toc428527049)

[3.2.1 Generating the compliance report for the project 13](#_Toc428527050)

[3.2.2 Analyzing single source files 14](#_Toc428527051)

[4. The QAC results 14](#_Toc428527052)

[4.1 QAC/QACPP project files: 14](#_Toc428527053)

[4.1.1 <algo>\_<C or CPP>\_<arch>.prj; e.g. PV\_C\_ti\_c674x.prj 15](#_Toc428527054)

[4.1.2 <algo>\_AnalyserPers\_arch>.p\_a; e.g. PV\_AnalyserPers\_C\_ti\_c674x.p\_a 15](#_Toc428527055)

[4.1.3 filelist\_<arch>.lst 15](#_Toc428527056)

[4.1.4 settings\_<arch>.via 15](#_Toc428527057)

[4.2 QAC analysis results 15](#_Toc428527058)

[4.3 The compliance report 15](#_Toc428527059)

[5. Typical errors 16](#_Toc428527060)

[5.1 QAC analysis does not work 16](#_Toc428527061)

[5.2 QAC warning messages 18](#_Toc428527062)

1. Figures

[Figure 1: External Tools in Visual Studio 10](#_Toc428527063)

[Figure 2: QAC-Targets in Visual Studio 13](#_Toc428527064)

[Figure 3: QAC-Output in Visual Studio 14](#_Toc428527065)

[Figure 4: QAC-Errors on the command line 16](#_Toc428527066)

[Figure 5: How to analyze a project in QAC GUI 17](#_Toc428527067)

[Figure 6: Analysis results in QAC-GUI 17](#_Toc428527068)

[Figure 7: How to start the Message Explorer in QAC-GUI 18](#_Toc428527069)

[Figure 8: GenericScons warning for missing directories 19](#_Toc428527070)

[Figure 9: GenericScons warning because of missing prj-template 19](#_Toc428527071)

# Introduction

GenericScons supports the generation of QAC- and QACPP compliance reports. This manual describes the necessary prerequisites for and usage of this feature.

# Prerequisites

## QAC installation

Copy QAC, QACPP-2.5 and QAW folders present under [\\LUL4GCNG\tools](file:///\\LUL4GCNG\tools) to local machine **C:\tools\**

OR

With the help of installation kit, you will find an installation guide for the installation. Please make sure that the individual components are installed to the correct directories. The following directories need to be present:

**C:\tools\QAC\**  **C:\tools\QAC\QAR-1.1\**  **C:\tools\QACPP-2.5\**  **C:\tools\QACPP-2.5\QAR-1.1\**  **C:\tools\QAW\**

Detailed description of installation steps:

Note: Please use the option Run As Administrator to install.

### QAC

1. Install QAC to the following directory *C:\Tools\QAC* ­otherwise the batch files won’t work!

Use this directory without subdirectories for versions.

1. Run the file “QAC-7.2-F-Win.exe”. Password: ”enforcer”.

Use the option “complete” during installation.

1. Run the file “MCM-4.6-QAC-7.2-Win.exe”. Password: ”jackhobbs”.
2. Run the file “M2CM-2.6-QAC-7.2-Win.exe”. Password: ”kellyholmes”.
3. Run the file “QAR-1.1-Win.exe”. Install it into: C:\Tools\QAC\QAR-1.1. You have to specify the QAR directory in the dialog to install it there.
4. Run QAC and enter license server: 30002@ls\_rb\_prflexd\_eu\_1.conti.de

### QACPP

1. Install QACPP to the following directory *C:\Tools\QACPP-2.5* – otherwise the batch files won’t work!
2. Run the file” QACPP-2.5-Win.exe”.Password: ”automobile”. Use the option “complete” during installation.
3. Run the file “MCPP-1.1-QACPP-2.5-Win.exe”. Password: ”segment”. Must be installed under: C:\Tools\QACPP-2.5\mcpp .
4. Run the file “QAR-1.1-Win.exe”. Install it into C:\Tools\QACPP-2.5\QAR-1.1. You have to specify the QAR directory in the dialog to install it there.
5. Copy->Paste “QAW” under C:\Tools\
6. License server for QAC/QACPP: [30002@ls\_rb\_prflexd\_eu\_1.conti.de](mailto:30002@ls_rb_prflexd_eu_1.conti.de)
7. Python is required: Python(x,y)-2.6.2.0.exe. Please restart your computer after installation.

## Shares

The following shares need to be provided in the project:

<http://mks-psad:7001/si/viewproject?projectName=/nfs/projekte1/REPOSITORY/Base%5fDevelopment/05%5fAlgorithm/ETK%5fEngineeringToolKit/04%5fEngineering/SCT%5fSconstools/project.pj>

CP 1.48 or higher

as *02\_Development\_Tools\scons\_tools.*  
  
<http://mks-psad:7001/si/viewproject?projectName=/nfs/projekte1/REPOSITORY/Tools/QAC/project.pj>

CP 1.35 or higher

as *05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\common.*  
  
<http://mks-psad:7001/si/viewproject?projectName=/nfs/projekte1/REPOSITORY/Tools/QACPP/project.pj>

CP 1.23 or higher

as *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common.*

## Adjusting GenericScons

### **Activation of QAC**

In order to activate the qac feature within GenericScons, add the following line to *04\_Engineering\03\_Workspace\algo\<component>\sconstruct\_config.scfg*:

generate\_qac = True

This is not needed any more from GenericScons ≥ AL\_ETK\_SCT\_01.06.00, since it is activated then per default.

### **(Optional) Modification of 01\_Source\_Code\algo\xxx\algo\_lib\_file\_list.scfg and 01\_Source\_Code\algo\xxx\algo\_config.scfg**

(see also corresponding templates in scons\_templates\01\_Source\_Code\algo\xxx)

* Inclusion of additional sources to be analyzed:

GenericScons advises QAC to analyze only those sources which are used to build ecu libraries. In some cases it is necessary to also analyze further source files, e.g. for the wrapper. This can be accomplished by including the following variable in algo\_lib\_file\_list.scfg and storing the corresponding sources there.

# algo wrapper sources to be analysed by qac

xxx\_wrapper\_sources = """

../00\_Custom/xxx\_wrp/xxx\_wrp\_main.c

"""

This source need to be provided to GenericScons in algo\_config.scfg. So include the following lines to algo\_config.scfg:

# wrapper source list for qac

wrapper\_src = Split(xxx\_wrapper\_sources)

Note: This variable is optional. If there are no additional sources to be analyzed, everything works fine if this variable is not defined.

* Definition of exclude paths:

Generally not all header files which are included via include paths need to be analyzed by QAC, but only those which “belong” to the individual component. E.g., some components include sources from cct and they don’t need to analyze those. In this respect there is a possibility to exclude the sources of certain directories, so then those sources are read by QAC, in order to understand the code, but no messages are given for those. Those “excludes” shall be defined in algo\_lib\_file\_list.scfg with the variable

# exclude paths for qac

qac\_exclude\_paths = """

#../../../01\_Source\_Code/algo/cct

#../../../01\_Source\_Code/algo/cml

#../../../01\_Source\_Code/common

#../../../01\_Source\_Code/common/rte

"""

Note: This variable is optional. If it is not set, per default QAC excludes all directories listed under “xxx\_algo\_common\_include\_paths”.

* QAC-options

Per default Scons passes the QAC / QAR options defined in 02\_Development\_Tools\scons\_tools\scons\_common\_config\common\_config.scfg to QAC(PP). In addition, extra QAC flags can be set in algo\_config.scfg for QAC tests via command line as well as via Visual Studio, please refer to the template in 02\_Development\_Tools\scons\_tools\scons\_templates\01\_Source\_Code\algo\xxx\algo\_config.scfg.

Remark:

When using visual studio for QAC analysis, per default, all warning messages are displayed in the output window. This output behavior may be controlled by introducing, the variable qac\_ide\_options to 01\_Source\_Code\algo\xxx\algo\_config.scfg. Here display options may be included:

-disp : displays warning messages

-su=0 : suppresses all warning messages of level 0

-su=2 : suppresses all warning messages of level 2

-su=3 : suppresses all warning messages of level 3

etc.

So, e.g.:

# qac options using visual studio

qac\_ide\_options = "-disp –su=0 –su=2"

suppresses all warnings of level 0 and 2 and displays all others.

Please note that this display manipulation does not affect in any way the content of the compliance report being generated right after analysis.

### **Integration of QAC-tests into visual studio solution**

It is possible to analyze the source code with visual studio. More precisely, the QAC output is printed to the output window of visual studio and the user can click the warnings in order to jump to the corresponding location in the source code. This is what needs to be done:

1. Visual studio support of GenericScons Release ≥ **AL\_ETK\_SCT\_01.03.01** Please make sure to have moved to the recent visual studio support of GenericScons, see

http://mks-psad:7001/si/viewrevision?projectName=/nfs/projekte1/REPOSITORY/Base%5fDevelopment/05%5fAlgorithm/ETK%5fEngineeringToolKit/04%5fEngineering/SCT%5fSconstools/docs/project.pj&selection=ReleaseNotes%5fAL%5fETK%5fSCT%5f01.03.01.txt

1. Modifications on scons.bat

(The goal of this modification is to assure correct build status output in visual studio.)

Please refer to the template in 02\_Development\_Tools\scons\_tools\scons\_templates\03\_Workspace\algo\xxx. The differences are:

Replace

..\..\..\02\_Development\_Tools\movpy\movpy.exe ..\..\..\02\_Development\_Tools\scons\_tools\scons\_adas\_extensions\sconscript\_setup.py

..\..\..\02\_Development\_Tools\movpy\movpy.exe ..\..\..\02\_Development\_Tools\scons\_tools\scons\scons.py %\* 2>&1 | ..\..\..\02\_Development\_Tools\scons\_tools\gnutools\tee.exe sconsbuild.log

and

endlocal & set SCONS\_ERRORLEVEL=%ERRORLEVEL%

by

REM Copy shared scons scripts to the locations defined in sonscript\_setup\_config.scfg

..\..\..\02\_Development\_Tools\movpy\movpy.exe ..\..\..\02\_Development\_Tools\scons\_tools\scons\_adas\_extensions\sconscript\_setup.py

REM Execute scons script, save returned error level to errorlevel.txt, and redirect scons output to tee.exe for build log.

(..\..\..\02\_Development\_Tools\movpy\movpy.exe ..\..\..\02\_Development\_Tools\scons\_tools\scons\scons.py %\* & call echo %%^^ERRORLEVEL%%>errorlevel.txt) 2>&1 | ..\..\..\02\_Development\_Tools\scons\_tools\gnutools\tee.exe sconsbuild.log

and

endlocal

REM Read the returned errorlevel in errorlevel.txt

set /p SCONS\_ERRORLEVEL=<errorlevel.txt

respectively.

1. Configuration settings for Visual Studio (optional; this is relevant, if visual studio shall be used for analyzing single source files with qac)

Note: These settings have to be configured only once; they will still be present when dealing with another sandbox.

In the Visual Studio please go to Tools -> External Tools…

Then the external tools-window pops up. Here press “Add” and you will be able to integrate execution of an external batch file. This needs to be done:

Title: “qac\_ti\_c674x\_analyze\_file”

Command: $(ProjectDir)..\..\..\..\02\_Development\_Tools\scons\_tools\scons\_common\_scripts\batch\scons\_qac\_ti\_c674x\_analyze\_file.bat

Arguments: $(ProjectFileName) $(ItemPath)

(Please note the space in between those arguments!!!!!!)

Initial directory: $(ProjectDir)

Tick “Use Output window”, see Figure 1.

Please perform accordingly for the remaining cores C66xx, Cortex A8 and Cortex A15: “Title”, “Arguments” and “Initial directory” need to be filled out identically as for the case of C674x. The individual commands read:

Command (C66xx): $(ProjectDir)..\..\..\..\02\_Development\_Tools\scons\_tools\scons\_common\_scripts\batch\scons\_qac\_ti\_c66xx\_analyze\_file.bat

Command (Cortex A8): $(ProjectDir)..\..\..\..\02\_Development\_Tools\scons\_tools\scons\_common\_scripts\batch\scons\_qac\_ti\_cortex\_a8\_analyze\_file.bat

Command (Cortex A15):

$(ProjectDir)..\..\..\..\02\_Development\_Tools\scons\_tools\scons\_common\_scripts\batch\scons\_qac\_ti\_cortex\_a15\_analyze\_file.bat

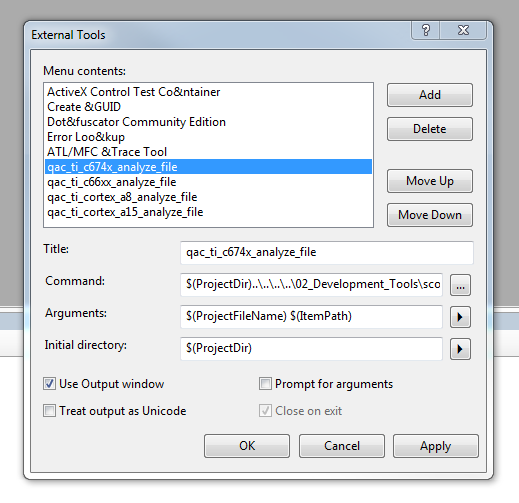
**

Figure 1: External Tools in Visual Studio

# How to QAC within GenericScons

QAC projects, analysis and reports can be generated for DSP, EVE and ARM libraries.

**Remark:** Before a QAC report is generated it is necessary for QAC to have the correct input files stored in specific locations, which GenericScons accounts for everytime a QAC-target is called. Precisely, these are the files being copied:

1. QAC:

* *05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\common\UserMessageFile\QAC.usr.Misra2004\_ADAS* is copied to

*C:\tools\QAC\bin* and to

*C:\tools\QAC\m2cm\messages.*

* *05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\common\UserMessageFile\qacmet.txt* is copied to

*C:\tools\QAC\bin.*

1. QACPP:

* *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common\UserMessageFile\QACPP.usr.Misra2008\_ADAS* is copied to

*C:\tools\QACPP-2.5\bin* and to

*C:\tools\QACPP-2.5\mcpp\messages.*

* *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common\UserMessageFile\qacppmet.txt* is copied to

*C:\tools\QACPP-2.5\bin.*

## Via Command Line

### **Generation of QAC-project**

The following command invoked from *04\_Engineering\03\_Workspace\algo\<component>* only includes the generation of QAC/QACPP project files:

scons.bat <algo>\_qac\_<architecture>\_proj # = project for certain architecture

scons.bat <algo>\_qac\_proj # = all projects for all architectures

where <architecture>=ti\_c674x, ti\_arp32, …; no analysis takes place and no reports are generated. Those project files (to be found in 05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\<algo>) may be used for manually analyzing the source code and generating the compliance reports.

### **Generation of Compliance Report**

For the generation of the QAC project, analyzing the source code and the Compliance Report for <architecture> use the following command from *04\_Engineering\03\_Workspace\algo\<component>*:

scons.bat <algo>\_qac\_<architecture>

For generation of all QAC projects and all Compliance Reports (except for the one for ARP32) use

scons.bat <algo>\_qac

### Analysis of single source files

Analyzing a complete QAC project is quite time-consuming and if one is merely interested in the analysis results of certain source files it is possible to generate a compliance report based on only these files with GenericScons. For this purpose, please call

scons.bat <algo>\_qac\_<architecture> qac\_source=file

where absolute paths to the files shall be given, e.g.:

scons.bat sr\_qac\_ti\_cortex\_a8 qac\_source= H:\sandboxes\regular\SR\_Sig

nRecognition\04\_Engineering\01\_Source\_Code\algo\sr\sr\_buildinfo.c

### Additional command line options

Additional QAC(PP) options can be passed with the option “qac\_options”, e.g.:

scons.bat qac\_options=-disp,-su=0,-su=2,-m-,-html+ <algo>\_qac\_ti\_c674x

Here all eligible QAC(PP) options can be applied, plus, if necessary, the GenericScons-specific QAC(PP)-option “-nostoponfail” which neutralizes the default QAC(PP) flag “-stoponfail”[[1]](#footnote-1). Please handle this option with care!

The format of compliance report can be manipulated with the option “qar\_format”, e.g.:

scons.bat qar\_format=xhtml <algo>\_qac\_ti\_c674x

### QAC option –j N – Multi-thread build

Usage of multi-thread build is activated for qac-targets. More precisely, if “-j N” is invoked for a QAC-target or a number of QAC targets on the command line GenericScons will launch N thread to finish QAC analysis and generate compliance report.

For the generation of the QAC project, analyzing the source code and the Compliance Report for <architecture> with multi-threading use the following command from

*04\_Engineering\03\_Workspace\algo\<component>:*

scons.bat –j 4 <algo>\_qac\_<architecture>

For generation of all QAC project and all Compliance Report with multi-threading use following command from

*04\_Engineering\03\_Workspace\algo\<component>:*

scons.bat –j 4 <algo>\_qac

## Via Visual Studio

Please generate the Visual studio solution with

scons.bat xxx\_ide\_msvc

where “xxx” represents the component name.

### Generating the compliance report for the project

In the visual studio solution choose the desired scons target in the configuration manager

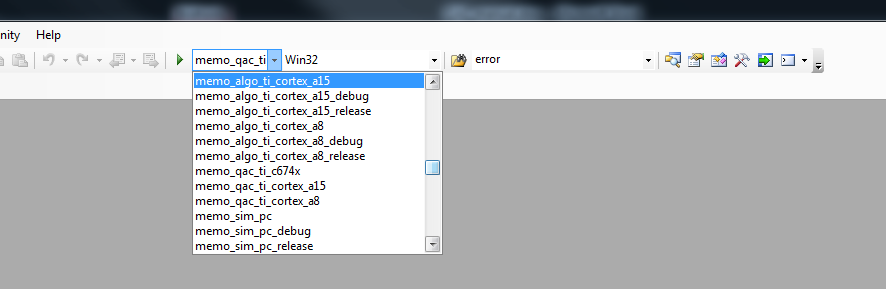


Figure 2: QAC-Targets in Visual Studio

and build the solution. As in the example calling “memo\_qac\_ti\_c674x” makes scons generate the qac-projects, analyse the code and generate the compliance report. During analysis of the code, per default, **all** QAC warnings are displayed in the output window. Those can be clicked in order to jump to the problematic position in the code, see, e.g.:

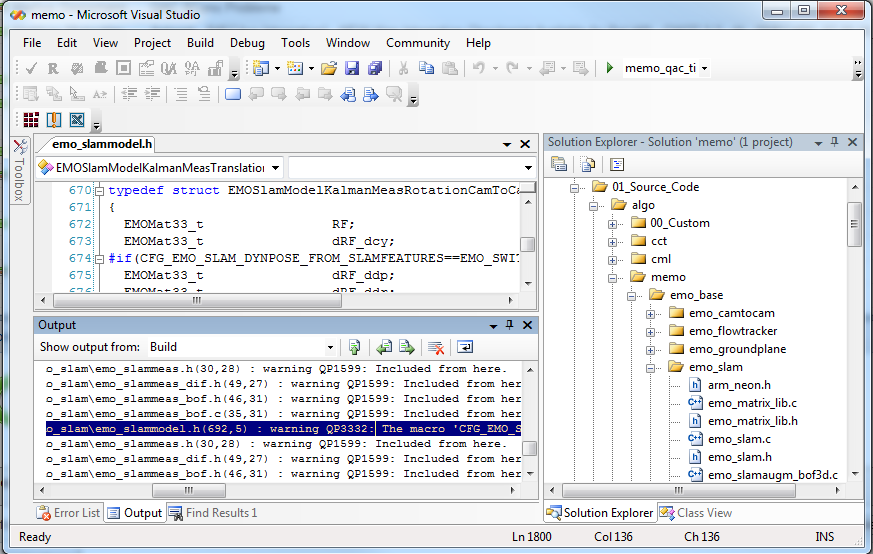


Figure 3: QAC-Output in Visual Studio

Typically there are quite a lot of warning messages, so there is the possibility to suppress warnings of certain levels, please refer to paragraph 2.3.2.

### Analyzing single source files

If the optional setting modifications of Visual Studio are set as described in Paragraph 2.3.3 iii., it is possible to analyze single source files. In order to do so, mark the source file to be analyzed and go to Tools -> qac\_<core>\_analyze\_file, e.g. qac\_ti\_c674x\_analyze\_file. Then this source file is analyzed and a compliance report is generated based on only this very source file. All the results are stored under *05\_Testing\05\_Test\_Environment\algo\modtests\qac(pp)\_tests\debug.* **No** compliance report is copied to 05\_Testing\02\_Reports.

# The QAC results

## QAC/QACPP project files:

The following files are stored in 05\_Testing\05\_Environment\_Tools\algo\modtests\qac\_tests\<algo> resp. 05\_Testing\05\_Environment\_Tools\algo\modtests \qacpp\_tests\<algo>.

### <algo>\_<C or CPP>\_<arch>.prj; e.g. PV\_C\_ti\_c674x.prj

This file is generated from template

*05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\common\qac\_python\_tools\QAC\_prj\_template.txt* resp. *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common\qac\_python\_tools\QACPP\_prj\_template.txt.*

### <algo>\_AnalyserPers\_arch>.p\_a; e.g. PV\_AnalyserPers\_C\_ti\_c674x.p\_a

This file is always generated from a template. By default, template *QAC\_p\_a\_template.txt* resp. *QACPP\_p\_a\_template.txt* is looked for in 05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\xxx resp. 05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\xxx.

If there is no template, it is looked for in *05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\common\qac\_python\_tools* resp. *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common\qac\_python\_tools*.

**Important Note:** Template *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common\qac\_python\_tools\QACPP\_p\_a\_template.txt* contains a bug and should **not** be used as is. Please create a component-specific template without the first line “-il 2”[[2]](#footnote-2).

### filelist\_<arch>.lst

### settings\_<arch>.via

## QAC analysis results

The results are stored in   
*04\_Engineering\04\_Build\algo\<algo>\_qac\<architecture>*. *04\_Engineering\04\_Build\algo\<algo>\_qacpp\<architecture>.*

## The compliance report

The compliance report is stored as **Compliance\_Report\_<algo>\_<arch>.html**

under *05\_Testing\02\_Reports\algo\modtests\qac\_tests\* resp. *05\_Testing\02\_Reports\algo\modtests\qacpp\_tests\*.

# Typical errors

## QAC analysis does not work

In most cases it is not directly possible to generate the compliance report as described above. If you try to generate the report via the command line, error messages like this will most likely occur:

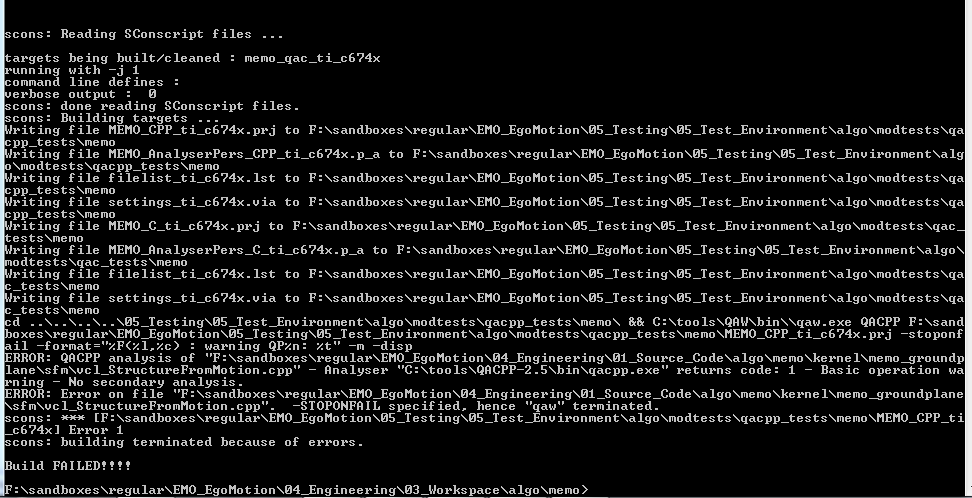
**

Figure 4: QAC-Errors on the command line

This is possibly caused by QAC parser errors during analysis. In this case, please only generate the QAC project files and open the .prj-file with the QAC/QACPP-GUI[[3]](#footnote-3) and analyze those there “by hand” by pressing the encircled button.

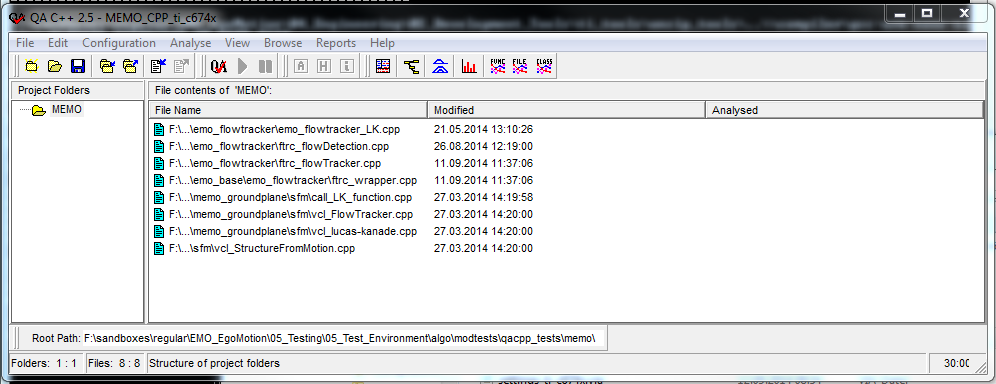


Figure 5: How to analyze a project in QAC GUI

The resulting analysis might look as follows:

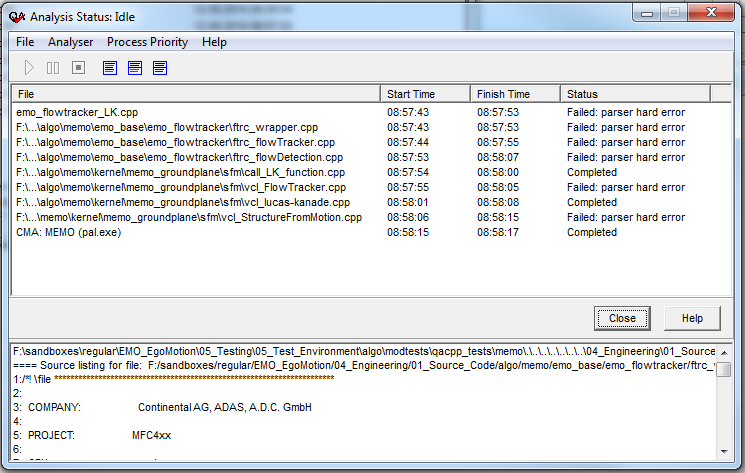


Figure 6: Analysis results in QAC-GUI

Those parser errors can be examined in the message explorer of QAC by pressing the encircled button in the following window:

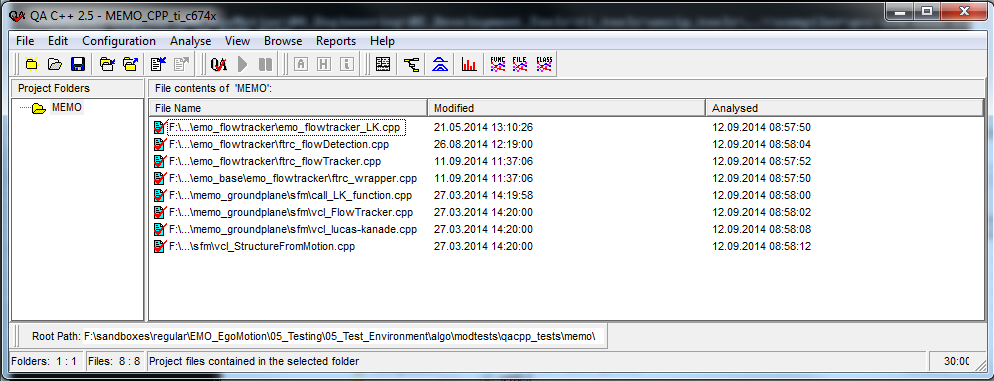


Figure 7: How to start the Message Explorer in QAC-GUI

The message explorer will then highlight the spots where QAC/QACPP had trouble to analyze the code.

**Possible resolutions:**

* Create/modify a component-specific analyzer personality template, see also paragraph 4.1.2. If not done yet, please remove “-il 2” from the template and/or add certain necessary component-specific macros in addition to the existing ones.
* Those errors can be caused by missing function declarations in the compiler includes[[4]](#footnote-4). In this case those compiler includes require an update. This should then be done locally in order to be able to analyze the code and then the QAC responsible should be informed.

As soon as no parser errors occur any more, the compliance report may be printed either in the GUI or the QAC integration in GenericScons can be used as described in Chapter 3.

## QAC warning messages

* If QAC has not been installed or it has not been installed to the required directories, see Section 2.1, the following messages might occur:

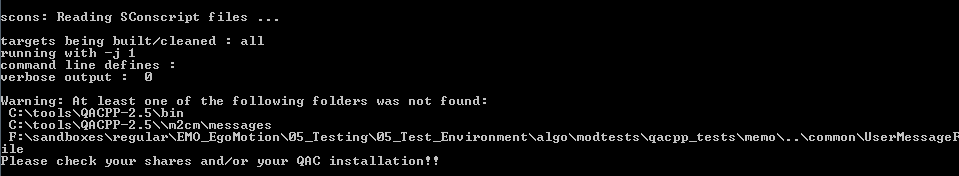


Figure 8: GenericScons warning for missing directories

This error may also occur, if the GenericScons copy action described on page 8 was not successful.

* If the shares under 05\_Testing are not as described in Section 2.2, scons will give possibly in addition to the preceding message the following error:



Figure 9: GenericScons warning because of missing prj-template

If those messages are received, it is still possible to build everything except for QAC-related targets. However, in order to use QAC properly please follow the individual tasks in Chapter 2 or, if this has already been done for your project, please resynchronize the corresponding locations.

1. -stoponfail advises QAC to stop analysis of source files as soon as a parser error occurs. In this case no compliance report is generated. [↑](#footnote-ref-1)
2. This line makes it impossible for GenericScons to analyze the source code. [↑](#footnote-ref-2)
3. Note that C-projects shall be opened with QAC (qacanl.exe) and CPP-projects shall be opened with QACPP (qacppanl.exe). [↑](#footnote-ref-3)
4. The source code typically contains includes from the compiler. Those includes are not suited to be analyzed by QAC, so all necessary functions, variables, etc. are declared in certain different compiler includes being located in *05\_Testing\05\_Test\_Environment\algo\modtests\qac\_tests\common\CompilerInclude* resp. *05\_Testing\05\_Test\_Environment\algo\modtests\qacpp\_tests\common\CompilerInclude.*

   It might be the case that something is missing there. [↑](#footnote-ref-4)