

First Tool Qualification Symposium
April 9th – 10th, 2013
Munich, Germany

Structure of the TI Compiler Qualification Kit

Texas Instruments Compiler Group
Thomas Suchyta
Trevor Jones

Outline

- Introduction to the TI compiler
- Our approach to qualifying the TI compiler to safety standards
- A model based process to compiler qualification
- The qualification kit's validation and testing framework

Introduction to the TI Compiler

- TI develops C/C++ compilers for many of our embedded processor families: ARM, C6000, C28x, MSP430
- The compiler group resides in Houston and Dallas and has been developing compilers for nearly 30 years
- Extensive experience in compiling for high performance architectures (VLIW, DSP, SIMD, μ C, RISC, CISC)
- The TI compiler generates highly optimized code for each target, including state-of-the-art whole program optimization
- Each compiler release undergoes an intensive and comprehensive validation and benchmarking process
- Future development work will include support for parallel processing through openMP and openCL

TI's Compiler Qualification Kit Project

- Worked with TÜV Nord, Validas AG, ACE in the development of compiler qualification kits for ARM, C28x, C6x
 - TÜV NORD (www.tuev-nord.de) – One of the largest technical service providers in Europe
 - Validas AG (www.validas.de) – A consulting company in the field of software quality for embedded systems. Tool qualification to safety standards is their main focus.
 - Associated Compiler Experts (www.ace.nl) – A supplier of products and services for professional compiler development and designers of the SuperTest compiler test and validation suite.
 - ACE SuperTest Qualification Kit test suite, a subset of the ACE SuperTest C Test and Validation Suite, will be included in the TI compiler qualification kit



TI's Compiler Qualification Kit Project

- New safety standards such as ISO 26262, DO-178C and IEC 61508 require customers document, analyze, and evaluate all software tools used in development.
 - Based on the evaluation, customers may be required to complete a tool qualification.
- Qualification methods:
 - Increase confidence from use
 - Evaluation of the tool development
 - **Validation of the software tool**
 - Development in compliance with a safety standard
- The qualification kit it will provide a way to qualify the compiler through validation
 - Validation must show that a tool satisfies its requirements using systematic tests. Furthermore, reaction to abnormal usage conditions has to be tested.

Model Based Approach to Compiler Qualification

- The compiler qualification kit uses the model based process developed by Validas
 - This process has been approved by TÜV Nord to meet the requirements of ISO 26262 and IEC 61508.
- The process uses the Validas Tool Chain Analyzer tools to provide a model of the TI compiler that includes:
 - Compiler use cases
 - Compiler features
 - Compiler artifacts (inputs and outputs)
 - Potential compiler errors
 - Possible error mitigations
 - Validation test cases
 - Documentation generation

Model Based Approach to Compiler Qualification

1. Tool (Chain) definition modeling

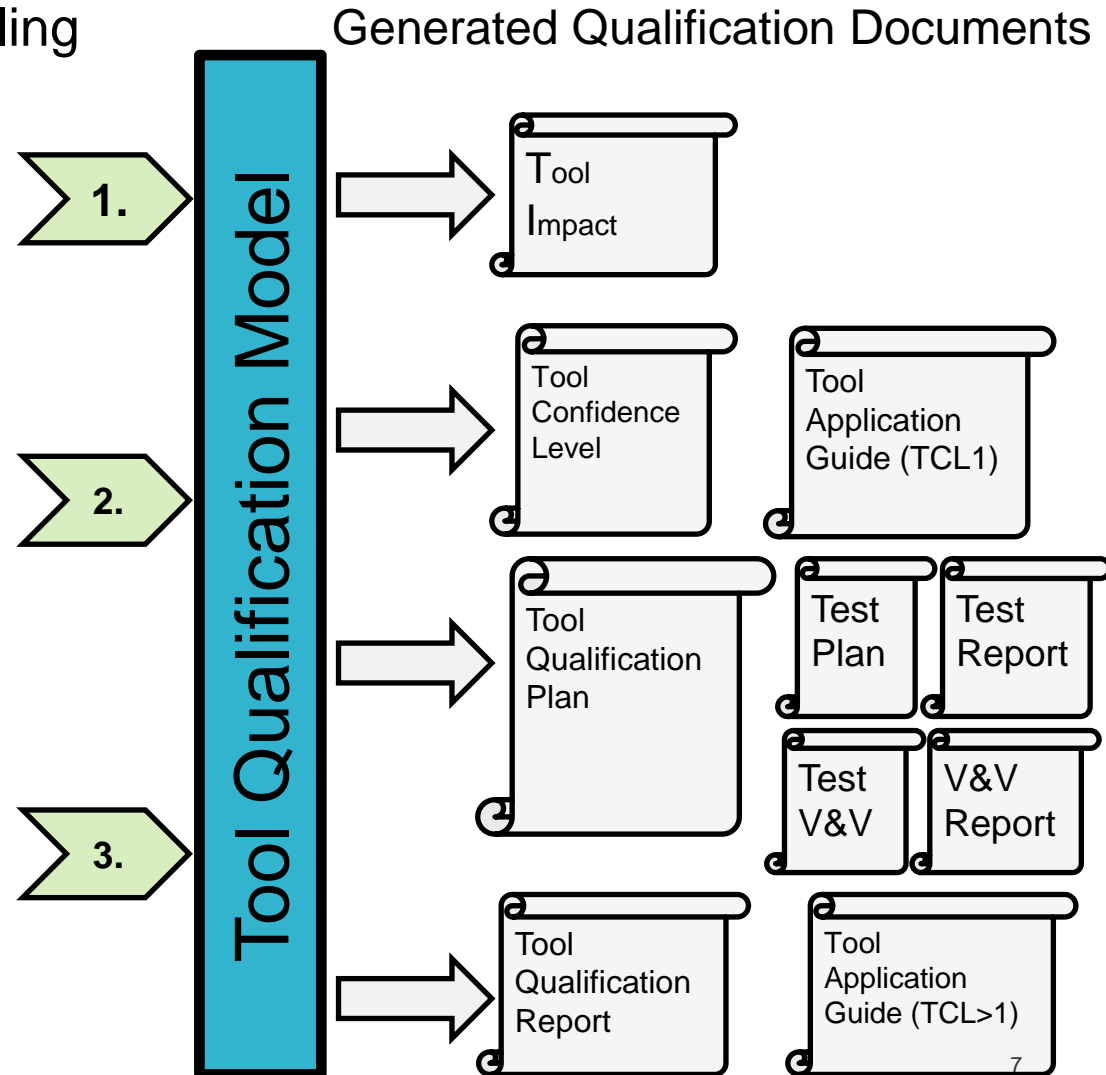
- Use-Cases,
- Relevant tool features,
- Artifacts
- Documentation

2. Tool (Chain) classification modeling

- Potential errors
- Available checks & restrictions
- Documentation

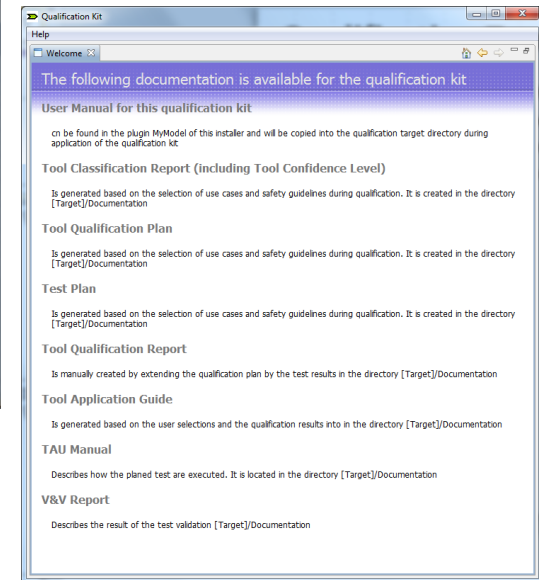
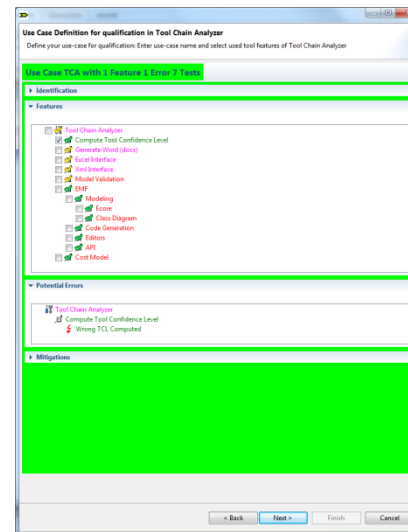
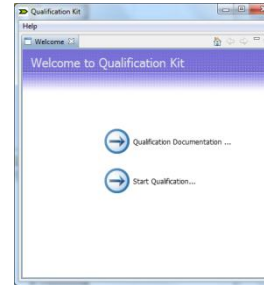
3. Tool (Chain) qualification modeling

- Qualification cost optimization
- Qualification planning
- Qualification tests:
 - Identification,
 - Validation Plan
 - Execution Plan
- Documentation



Model Based Approach to Compiler Qualification

- A flexible approach to tool qualification since it does not limit the use cases
 - It also provides flexibility in choosing error mitigations and validation test cases.
- Validas Tools provide a way to work through the entire process
 - Guides you through the qualification process
 - Helps selecting features (dependencies)
 - Helps selecting mitigation measures
 - Generates documents based on selections
 - Shows qualification status

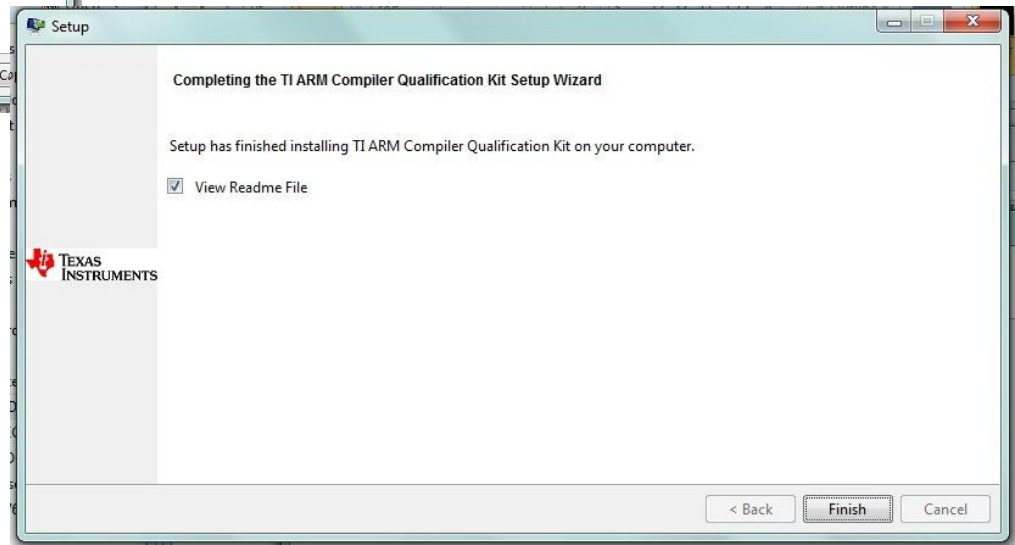


Structure of the TI Compiler Qualification Kit

- The TI qualification kit will provide a framework that will allow the running of the validation test cases
 - The test cases are chosen through the model based qualification process
- This testing framework platform:
 - Is a Windows and Linux command line tool
 - Requires Perl and Code Composer Studio Version 5
 - Test cases are executed on a hardware emulation environment

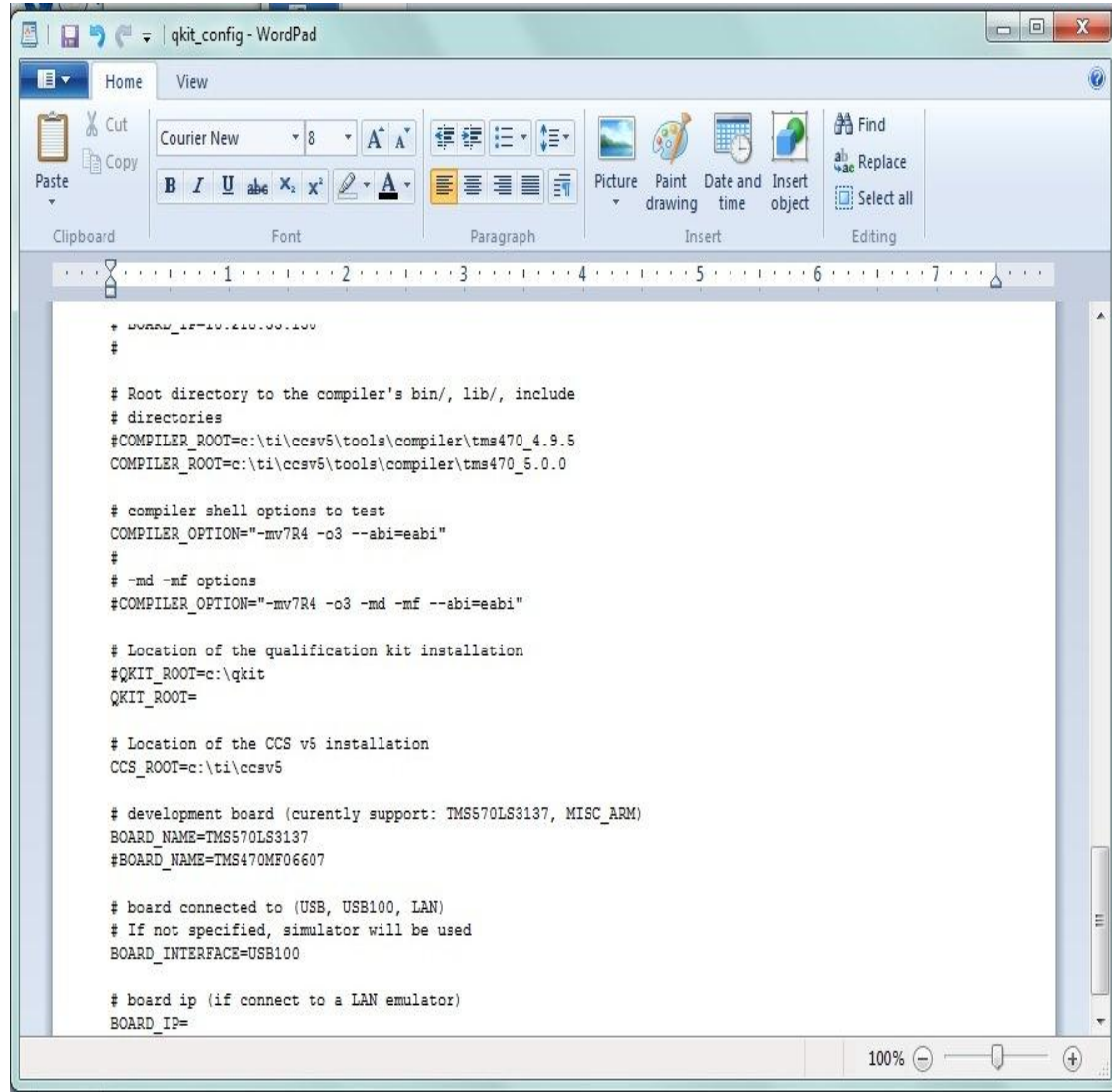
Compiler Qualification Kit Testing Framework

- Installation of the kit is typical of any other program



Compiler Qualification Kit Testing Framework

- A single text file controls the configuration of the testing framework
- To configure edit:
 - the compiler location
 - CCS installation
 - compile options to test
 - hardware emulation environment



```
qkit_config - WordPad
#
# Root directory to the compiler's bin/, lib/, include
# directories
#COMPILER_ROOT=c:\ti\ccsv5\tools\compiler\tms470_4.9.5
COMPILER_ROOT=c:\ti\ccsv5\tools\compiler\tms470_5.0.0

# compiler shell options to test
COMPILER_OPTION="-mv7R4 -o3 --abi=eabi"
#
# -md -mf options
#COMPILER_OPTION="-mv7R4 -o3 -md -mf --abi=eabi"

# Location of the qualification kit installation
#QKIT_ROOT=c:\qkit
QKIT_ROOT=

# Location of the CCS v5 installation
CCS_ROOT=c:\ti\ccsv5

# development board (currently support: TMS570LS3137, MISC_ARM)
BOARD_NAME=TMS570LS3137
#BOARD_NAME=TMS470MF06607

# board connected to (USB, USB100, LAN)
# If not specified, simulator will be used
BOARD_INTERFACE=USB100

# board ip (if connect to a LAN emulator)
BOARD_IP=
```

Compiler Qualification Kit Testing Framework

- Execution of the test cases is through a Perl driver program

```
ca: Command Prompt
C:\Ti_Development\qkit>
C:\Ti_Development\qkit>
C:\Ti_Development\qkit>perl qkit_driver.pl
set QKIT_ROOT=C:\Ti_Development\qkit
Creating canonicalizer...
copy(C:\ti\ccsu5\tools\compiler\tms470_5.0.0\bin\armcl.exe, .\canonicalize.exe)
Deriving compiler version...
Loading test information.....done (loaded 1 tests in 00:00)
Building test commands...done

:compiler_cio_001:$-/:L7/26: Adding fixed option set '-q'
:compiler_cio_001:$1/2:info:
:compiler_cio_001:$1/2:info: ===== Running option set 1 of 2 : '-q' =====
:compiler_cio_001:$1/2:L12/26: cleaning/deleting 'test1.obj'
:compiler_cio_001:$1/2:L16/26: cleaning/deleting 'test1.out'
:compiler_cio_001:$1/2:L10/26: %sh CMD> armcl --silicon_version=4 --abi=eabi -q
test1.c
:compiler_cio_001:$1/2:L11/26: ensuring that RC=0
:compiler_cio_001:$1/2:L12/26: ensuring that 'test1.obj' now exists
:compiler_cio_001:$1/2:L14/26: %sh CMD> armcl --silicon_version=4 --abi=eabi -q
test1.obj -z -lnk.cmd -lrtsv4_0_be_eabi.lib -o test1.out
:compiler_cio_001:$1/2:L15/26: ensuring that RC=0
:compiler_cio_001:$1/2:L16/26: ensuring that 'test1.out' now exists
:compiler_cio_001:$1/2:L18/26: %sh CMD> load470.bat test1.out
:compiler_cio_001:$1/2:L18/26:
:compiler_cio_001:$1/2:L18/26: ***** DSS Generic Loader *****
:compiler_cio_001:$1/2:L18/26:
:compiler_cio_001:$1/2:L18/26: START: 21:10:16 GMT-0500 <CDT>
:compiler_cio_001:$1/2:L18/26:
:compiler_cio_001:$1/2:L18/26: Configuring Debug Server for specified target...
:compiler_cio_001:$1/2:L18/26: Done
:compiler_cio_001:$1/2:L18/26: TARGET: Cortex-R4 CPU Functional Simulator, Big E
ndian_0
:compiler_cio_001:$1/2:L18/26: Connecting to target...
:compiler_cio_001:$1/2:L18/26: Resetting target...
:compiler_cio_001:$1/2:L18/26: testEnv.outFiles: test1.out
:compiler_cio_001:$1/2:L18/26: Loading test1.out
:compiler_cio_001:$1/2:L18/26: Done
:compiler_cio_001:$1/2:L18/26: Target running...
:compiler_cio_001:$1/2:L18/26: Interrupt to abort . . .
:compiler_cio_001:$1/2:L18/26: Start IO test
:compiler_cio_001:$1/2:L18/26: a = 5
:compiler_cio_001:$1/2:L18/26: s = string test
:compiler_cio_001:$1/2:L18/26: c = c
:compiler_cio_001:$1/2:L18/26: f = 1.6
:compiler_cio_001:$1/2:L18/26: End IO test
:compiler_cio_001:$1/2:L18/26: NORMAL COMPLETION: 0 cycles
:compiler_cio_001:$1/2:L18/26:
:compiler_cio_001:$1/2:L18/26: END: 21:10:20 GMT-0500 <CDT>
:compiler_cio_001:$1/2:L19/26: ensuring that RC=0
:compiler_cio_001:$1/2:L20/26: ensuring that the last command's output contains
string "Start IO test"
:compiler_cio_001:$1/2:L21/26: ensuring that the last command's output contains
string "a = 5"
:compiler_cio_001:$1/2:L22/26: ensuring that the last command's output contains
string "s = string test"
```

```
ca: Command Prompt
:compiler_cio_001:$-/:EOF: 1 Passed ==> -q
:compiler_cio_001:$-/:EOF: 2 Passed ==> -mv7R4 -o3 --abi=eabi
:compiler_cio_001:$-/:EOF:
:compiler_cio_001:$-/:EOF: Setting file .qkitresults

perl ./report --qkit -f -t TMS470 compiler_cio_001

- Qualification kit passing tests -

TEST DESCRIPTION PASSED STATUS
-----
compiler_cio_001: Printf format tests 2/2 Passed

DESCRIPTION: Test various printf formats.
FEATURE: Compiler - C I/O
1. Passed ==> -q
2. Passed ==> -mv7R4 -o3 --abi=eabi
-----
1 Tests
1 tests passed
0 tests failed
*****
** TI Compiler Qualification Kit Final Results **
** Time stamp: 28032013_211009 **
*****

- Qualification kit passing tests -

TEST DESCRIPTION PASSED STATUS
-----
compiler_cio_001: Printf format tests 2/2 Passed

DESCRIPTION: Test various printf formats.
FEATURE: Compiler - C I/O
-----
1 Tests
1 tests passed
0 tests failed

C:\Ti_Development\qkit>
```

Compiler Qualification Kit Testing Framework

- Validation reports are generated after test case execution

```
*****
** TI ARM Compiler Qualification Kit      **
** Version 1.0.1A                        **
** Copyright 2012                         **
**                                         **
** Starting Time: 2012-12-14 11:19:08    **
*****
```

```
OS Used:      Windows
Compiler Path: C:\arm_502\bin
Compiler Include: C:\arm_502\include
Compiler Library: C:\arm_502\lib
Loader CCXML Used: TM9570L83137_xds560v21an.ccxml
```

```
-----
- Qualification kit passing tests -
-----
```

```
TEST DESCRIPTION          PASSED
STATUS
-----
-----
compiler_cio_001: Printf format tests
                           2/2
Passed
```

Passed

2/2

DESCRIPTION: Test linker splitting of output sections across memory regions with separate load and run placements.

FEATURE: Linker - create output sections, link section in specific memory regions

lnk_region_004: Linker memory region tests

2/2

Passed

DESCRIPTION: Test that memory regions are correct and output sections are in correct memory regions.

FEATURE: Linker - link sections in specific memory regions

59 Tests

59 tests passed [including 3 skipped tests]
0 tests failed

Summary

- TI Compiler Qualification Kit is a tool that aids in qualifying the toolset to safety standards through validation
- The kit uses a TÜV Nord ISO 26262 and IEC 61508 approved model based qualification process developed by Validas
- Current development schedule
 - ARM beta kit June 2013
 - C28x, C6x beta kits 3Q 2013
 - See this TI wiki page for latest qualification kit news and schedule:
http://processors.wiki.ti.com/index.php/Compiler_Qualification_Kit
- Questions?
 - Project lead: Thomas Suchyta t-suchyta1@ti.com