C

C Compiler Validation for Embedded Targets

Qualifying Compilers for Use in Safety-Critical Projects

First Edition
by
Mrs ***Olwen Morgan*** *CITP, MBCS*
and
Eur Ing ***Chris Hills*** *BSc , C. Eng., MIET, MBCS, FRGS, FRSA*

The Art in Embedded Systems comes through Engineering discipline.

Contents

C Compiler Validation for Embedded Targets

Qualifying Compilers for Use in Safety-Critical Projects

These notes have been prepared for those investigating validation services for C cross-compilers for embedded targets. Some may find validation unfamiliar and the aim of this document is to explain the essentials so that it is clear what to expect from Phaedrus Systems compiler validation services.

What is compiler validation?

Compiler validation is essentially highly controlled, repeatable and reproducible testing of a compiler using a validation suite - a recognised set of test programs. The purpose of such testing is to provide a reliable indication of how well a compiler complies with the standard for the language that it implements.

Whilst that is the simple explanation there is a lot of subtle complexity and rigour involved, coupled with a deep understanding of the C language standard (ISO 9899) and laboratory standard testing methods. For example Phaedrus Systems uses one of the worlds best C Language Compiler Test Suites and custom hardware test rigs. Our validator has several decades experience of working on high SIL projects, compiler validation and critical systems. Compiler validation is not something that should be undertaken by developers on a development PC.

Also there is a difference between general compiler validation and on-target validation. Phaedrus Systems understands the critical differences and has adapted the test suites and test rig accordingly.

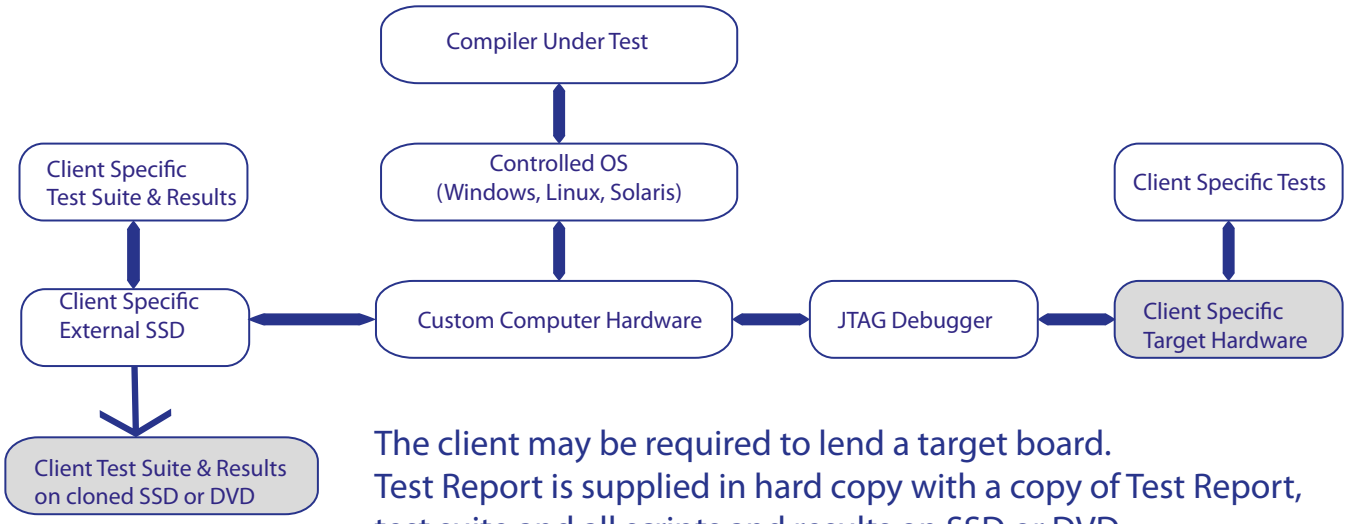
Furthermore, not only must the documentation of the results of the tests be of a standard for the Project Certification Body but of a standard suitable, if necessary, to be used as evidence in a Court of Law.

The selection of a suitable test suite is paramount. Not everything called a "Test Suite" is suitable for validating a C compiler to the required standard: there are very few Test Suites that are acceptable to the Certifying Bodies. Phaedrus Systems' validator has experience of the leading C compiler test suites.

Who needs compiler validation?

Typically compiler validation is sought in connection with the development of safety-critical, security-critical or mission-critical systems. Until recently, compiler validation has been recommended but rarely formally required for the development of safety-related software

Phaedrus Systems On-Target Compiler Validation Test System



The client may be required to lend a target board.
Test Report is supplied in hard copy with a copy of Test Report, test suite and all scripts and results on SSD or DVD.

complying with safety standards such as IEC 61508 Part3, ISO 26262 (Automotive), EN 50128 (Rail), ISO 60880 (Nuclear,), ISO 62034 (Medical), and/or comparable provisions in similar industry-specific standards. Most validations up to 2014 have been for large-scale system developments. From 2014 this is beginning to change as a result of technical advances in microcontroller design.

MCU's and software are being increasingly used in critical systems, software is increasing in volume and complexity and is controlling more critical functions. Consequently Certification Bodies are starting (as of 2015) to request that compiler validation is done on the actual target MCU used on the project, with the appropriate compiler switches and flags set.

Various microcontroller manufacturers now offer safety-rated microcontrollers. An increasingly popular form is the dual-core lockstep microcontroller for which regulatory bodies are beginning to issue safety certifications covering the hardware. These microcontrollers offer very high safe failure fractions through their physical and logical design. One weakness, however, is that both cores run the same object-code image. For software to match the safe failure fraction achieved by the hardware, it is now becoming more important for the critical parts of development tool chains to be proven fit for purpose.

To date many compilers used in embedded system development have been accepted as "proven in use", that is they have a history of reliable operation. This only works for compilers supplied as object code, where, for example there 10,000 users of the same binary and with a fixed set of components. "Proven in use" would not hold for a compiler supplied in source form, for example GCC, where the users build from source code or deploy a variable set of components.

Developments in hardware safety are causing a shift towards the use of formally validated compilers where previously validation would not have been required. Developers can now expect regulatory bodies to require compiler validation more and more often for critical system developments, and increasingly the requirement will be for on-target validation.

Why the compiler developer's testing may not be enough

Most compiler suppliers make extensive use of recognised test suites, such as SuperTest, Perennial or Plum-Hall, in their product development. This however, can really only establish that a compiler is qualified for a range of critical uses. It does not establish that a compiler is fit for purpose in any particular use. For embedded C cross-compilers it is important to understand why this is.

The C language standard (ISO 9899) makes clear that an implementation is a specific configuration of hardware platform and compiler software used under particular compiler invocation options - the specific set of switches and options used when setting the target MCU.

However thoroughly a compiler vendor may test his

compiler, it is likely that:

(a) The testing will have been done on-host (using a simulator or emulator rather than real target hardware) because the test suite may not fit on the target or on-target testing may not be flexible enough or can take too much time.

(b) Testing has not been done with exactly the same compiler options that are used in a specific embedded development project.

NOTE: - Embedded C Compilers may be using a variant of ISO-C 9899:1990, 1995, 1999 or 2011 with extensions and/or restrictions when running on the specific target.

Consequently the compiler developer's testing cannot test the exact implementation that the embedded developer will be using. This is not a minor issue: a small change in the compiler options can have a huge impact on the generated object code. Compiler validation services fill this gap by performing tests under the implementation options used for a particular development project and as far as possible on actual target hardware.

What does a validator do?

By "validator" we mean a qualified and suitably experienced engineer who performs validation testing. The validator's task includes the following:

(a) Setting up and configuring a test host (usually a PC) to a known and repeatable initial state under a specified version of the host operating system. This is not as simple as it sounds and Phaedrus Systems Validators build custom PC's specifically for Compiler Validation as standard PC's are not suitable. Also the PC operating system has to be maintained to a known state.

(b) Installing a specified version of the compiler to be tested. This will also include specific settings, flags and switches all of which are required to be fully documented.

(c) Installing suitable test driver software. Again this has to be rigorously documented and recorded using sound methods. How the test software is set up will be crucial to the running of the tests.

(d) Preliminary testing to ensure that the programs of the test suite can be satisfactorily run and their results are correctly gathered by the test driver software.

(e) Running the tests on-host to provide assurance of the integrity of the host platform configuration.

(f) Running the tests on a specified externally connected target via a debugging adaptor. The parameters of the debugging adaptor need to be recorded and understood.

(g) Analysing any test failures to determine their causes (together with any test re-runs required). Test failures may or may not be significant and in some cases can be expected with a specific implementation on a specific target.

(h) Preservation of the test transcript produced by the test tool. All data must be recorded and archived.

(i) Preparation and issue of a test report to the rigour and standards required.

(j) Issue of a test certificate.

These steps are conceptually very simple but the key constraints are that they must be both repeatable and reproducible. To be repeatable it must be possible for the any qualified tester using the same test system to get the identical results on successive test runs. To be reproducible it must be possible for a different tester using a different but technically compatible test system to get the same results on separate test runs. In practice this means that testing must be controlled to the same standards that are applied to certification testing laboratories.

Testers must be specially trained in order to be qualified to undertake testing to this degree of rigour. Most systems engineers, even very experienced ones, are not actually qualified to this level and most in-house test systems seen by the authors would not be fit-for-purpose in compiler validation.

On-host and on-target validation times

On modern PC hardware a test suite containing 10's of thousands of tests can usually be run on-host several times in a day.

On-target testing takes considerably longer because the total elapsed time is dominated by the time required to upload tests to the target environment and download the results from that environment. Depending on it's design a large test suite may take several days, or even weeks to run on-target if the turn-around times are long,. Indeed there may be occassions where it is not possible at all: some test suites are not designed in such a way that they cannot even be be used on-target.

An evolving practice in performing validations for clients is that a first validation is conducted on the host. Then a trial validation is conducted on the required target simply to find out how long it will actually take to run on a specific target (MCU, clock speed, memory etc. all play a part). Clients for validation services should therefore be prepared for providers of such services to take a little while before giving definite elapsed time estimates for validation work. For the moment it is the validation provider's responsibility to ensure that:

(a) Validation clients understand that for a previously untested compiler/target combination, initial estimates will be subject to uncertainty,

(b) Validators are themselves prepared to undertake investigative work to resolve any time uncertainties before quoting specific terms and costs to clients.

It is clearly in the best interests of both clients and providers that they make allowance for initial time uncertainties until experience of running a large test suite has accumulated to provide a better basis for estimation

Front-end and back-end testing

Different test suites test different aspects of a compiler's behaviour. Basic conformance suites test that the compiler observes the standard syntax of the language and can correctly compile at least one instance of each language construct.

Tests that do this are very good exercisers of a compiler's front end. Among contemporary C compilers, it is now relatively unusual to find serious errors in their front ends.

By comparison the back-ends (i.e. the code generators) of compilers are less well tested. This is particularly true of C compilers where many compiler options bear on technical matters at the object-code level. To exercise the back-end needs either a large test suite (such as SuperTest) whose design is based on a boundary-value test coverage strategy, or a set of tests generated by a random stress test generation tool.

In practice, and in aid of repeatability and reproducibility, the most controllable approach is the use of a large fixed test suite specifically designed to cover an appropriately strong domain of boundary-value test cases. The strength of such test suites is that they provide a strenuous exercise of a code generator in how it handles arithmetic, which is generally of more concern to embedded system developers than front-end compliance.

Validation reports and transcripts

Validation reports must contain at least the following elements:

(a) A technical description of the steps taken to configure the test system and run the tests,

(b) Resolution of any test failures,

(c) Transcripts of all test runs as produced by the test driver software,

(d) A certificate signed by the tester confirming that the tests have been done as described and that the test transcripts are authentic.

For ease of presentation test transcripts are conventionally presented on digital media with all other elements of the test report as printed items.

It is normal practice for validators to retain copies of results for reference purposes subject to appropriate Non-Disclosure Agreements (NDAs). Phaedrus Systems does this by archiving the SSD used for the tests including all test documentation and test results. At any time all evidence of the tests run can be produced and, if needed, the tests can be repeated in an identical environment.

Currently the authors also recommend to clients that the results of the tests should be made available to compiler and test suite developers, again under NDA, as this is likely to be of help to them in eliminating any bugs found by on-target testing.

A further convention is that the validation provider retains all Intellectual Property Rights (IPR) in test reports and transcripts. The validator then has the legal right to require that any copies presented for regulatory approval are as produced by the tester and do not have any parts removed by the client. (This has been known to happen and "adjusted" test reports supplied to the Regulators for Approval and Certification.)

On-host and on-target validation times

Depending on the scope of service offered, validation providers may offer further advice in dealing with regulatory authorities regarding the documentation of compiler validation procedures in safety cases.

It may happen that errors are found during testing for which the developer needs to take mitigation actions

during project development. A qualified validator should have the expertise to advise on such mitigations and how they can be tested. Robustness of mitigation can be demonstrated by re-running relevant failing test programs from the test suite and then running the programs that test the mitigation.

When this is done under laboratory conditions, the advising validator must not participate in any retests where he/she has had a role in specifying what the mitigation tests should be. Ideally this should also be avoided in testing that is not subject to laboratory-level certification procedures. For the moment, however, a pragmatic approach is acceptable provided that the validator's role in post validation advice is subject to agreement between the client and the validation provider.

Conclusion

Phaedrus Systems hopes that prospective validation clients will find these notes helpful and will be glad to answer any specific queries that they may arise

C Compiler Validation for Embedded Targets

First edition July2016

Phaedrus Systems Library

The Phaedrus SystemsLibrary is a collection of useful technical documents on development. This includes project management, integrating tools like QA·C to IDE's, the use of debuggers, coding tricks and tips. The Library also includes the QuEST series.

Copies of this paper (and subsequent versions) with the associated files, will be available with other members of the Library, at:

http://library.phaedsys.com

PhaedruS SystemS

The Art in Embedded Systems comes through Engineering discipline.