Qualcomm Technologies, Inc.

Qualcomm® Hexagon™ C++ Library

Supplement

80-N2040-43 Rev. A

March 30, 2018

Revision history

Revision	Date	Description
A	March 2018	Initial release

Contents

1 Introduction

1.1 Purpose

This document describes how to select the C++ library to use of the two that are provided with the Hexagon Tools and the reasons for selecting one library over the other.

NOTE: This document is a supplement to the *Hexagon C++ Library User Guide* (80-N2040-14) and is meant to be used in conjuction with that document.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `cp armcc armcpp`.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Usage Details

The C++ libraries are distributed with the Hexagon Tools for use in writing C++ programs. The libraries are used to support C++ language features. They also include the containers, file I/O, strings, exceptions, smart pointers, and other features described by the C++ standard. Two different libraries are distributed with the Hexagon Tools: libstdc++ and libc++.

2.1 Using the C++ library

By default, `hexagon-clang++` uses the C++03 standard and selects the libstdc++ library for compilation and linkage. If the more recent standards are specified, such as C++11 (`std=c++11`), C++14 (`std=c++14`), etc., `hexagon-clang++` will select the libc++ library for compilation and linkage.

You can use the `stdlib=libc++` or `stdlib=libstdc++` arguments with `hexagon-clang++` to select the appropriate standard library.

2.2 Differences between libstdc++ and libc++

libstdc++ is the “historical” C++ library, and it supports only C++98 and C++03 language modes. It can be used with standalone mode to write and debug C++ programs. This standalone mode is often used in conjunction with the Hexagon simulator. Refer to *Hexagon Stand-alone Application User Guide* (80-N2040-22).

libc++ was introduced to support C++11 and beyond. libc++ requires support in the form of OS include paths for headers and libraries. This means that many programs built with libc++ will not successfully compile without explicit `-I/path/to/os/include` arguments, and they will not link without explicit arguments for the linker as well.

The libc++ standard library is distributed as two distinct libraries, libc++ and libc++abi. When `-stdlib=libc++` is used, the compiler/linker is configured to resolve to each of those respective includes/libraries. If you use the `-nostdlib` argument to disable the automatic inclusion of libc++, you should know that the C++ code will require linkage with both libc++ and libc++abi.

2.3 Compiling with C++11

When compiling source files in C++11 mode, you must use libc++. You must also provide the necessary include paths to find the QuRT headers.

For example:

```
hexagon-clang++ -std=c++11 -I/path/to/qurt/install/variant/include/qube -
I/path/to/qurt/install/variant/include/posix -
```

```
I/path/to/qurt/install/variant/include/qurt -c -o new_program.o
new_program.cpp
```

2.4 System configuration limitations

2.4.1 Simulator

Some library features include time/date referenced by the epoch (“wall clock”) and interval times that can be used in delays (e.g., `std::this_thread::sleep`) or time-bounded waits (`try_lock_for()`, `wait_for()`, etc.). In a normal system, these are often driven by a timer tick interrupt that elapses at a given interval. When using `hexagon-sim` with QuRT, these features depend on co-simulators (“cosims”) used to model this external device. The qtimer cosim may be necessary to simulate a real hHexagon processor. Refer to the section on cosimulation in *Hexagon Simulator System API User Guide* (80-N2040-18) for details. Refer to the qtimer cosim example included with the Hexagon tools for an example of using the qtimer cosim. This example is found in <Hexagon_tools>/Examples/cosims/qtimer_test.

2.4.2 Target

Note that QuRT will not provide a real UTC reference, so wall clock times will not match other processors in the SoC or in the outside world.

A References

A.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
Hexagon C++ Library User Guide	80-N2040-14
Hexagon Stand-alone Application User Guide	80-N2040-22
Hexagon Simulator System API User Guide	80-N2040-18