Allied Vision

Application Note

Cross-compiling Vimba X Source Code to ARM

Document V1.0.0
October 2022

Scope of this document

Vimba X offers numerous possibilities to cross-compile source code from a PC to an ARM64 board.
This document describes a tried and tested easy solution.

Introduction

Vimba X runs on several platforms and is source code compatible across all of them. This document refers to a common and uncomplicated solution: Most users develop on a Linux PC with Ubuntu. Nevertheless, you can also use a different distribution.

Note that if you cross-compile any source code from Windows to an ARM64 board with Linux, you generally have to be aware of potential platform dependencies. Although Vimba X is fully compatible, you might have to adapt your code to the different system libraries. This is always the case and independent of the API.

Recommended prerequisites

- PC with Linux and Vimba X for ARM64
- ARM64 board or SOM with Linux and the same Vimba X for ARM64 version as on the PC
- Cross-compilation toolchain, e.g., Linaro (includes all required libraries)

Selecting a cross-compilation toolchain

In contrast to building applications on your host system, cross-compiling requires a compiler that creates binaries for the target instead of the host. Such a compiler and the corresponding C/C++ standard libraries are part of a cross-compilation toolchain. We have tested several toolchains and recommend the Linaro suite.

When using CMake as your build tool, you can provide a CMake toolchain file to specify a cross-compiling toolchain. The Vimba X C and C++ APIs can easily be used in a CMake project. For details, see the examples provided with Vimba X.

If you use a different toolchain than Linaro, note that the C/C++ runtime library versions of this toolchain and on the ARM64 board must be compatible with each other. Vimba X requires a toolchain that targets the GNU triplet 'aarch64-linux-gnu'. A gcc compiler 8.4.0 and the corresponding C and C++ runtime libraries are recommended.

To identify compatible libc versions on your ARM6464 target, use the command:
```
strings /usr/aarch64-linux-gnu/lib/libc.so.6 | grep GLIBC
```

Cross-compiling to ARM

To cross-compile from a Linux PC to an ARM64 board:

1. Make sure the same Vimba X versions are installed on the PC and on the ARM64 board.
2. On your host PC, unpack the toolchain to your hard drive, e.g., /home/foo/linaro. If you use a different toolchain, follow its installation instructions.
3. Create a CMake toolchain file named 'aarch64-linux-gnu.toolchain.cmake'.
   The contents should look similar to this example:

```
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR aarch64)

set(CMAKE_C_COMPILER "/home/foo/linaro/bin/aarch64-linux-gnu-gcc")
set(CMAKE_CXX_COMPILER "/home/foo/linaro/bin/aarch64-linux-gnu-g++")

set(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
set(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
set(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)

set(CMAKE_C_FLAGS "-march=ARM64v8-a")
set(CMAKE_CXX_FLAGS "-march=ARM64v8-a")

# cache flags
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}" CACHE STRING "c flags")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS}" CACHE STRING "c++ flags")
```

4. Configure a Vimba X example with cmake and provide a toolchain file by setting the variable CMAKE_TOOLCHAIN_FILE e.g. on the command line:
   `-DCMAKE_TOOLCHAIN_FILE=/path/to/toolchain/aarch64-linux-gnu.toolchain.cmake`

   The default settings of the Linaro cross-compiler enable it to locate the according standard libraries automatically.
5. Build the project and transfer the generated binaries to the ARM64 board, e.g., by copying them to its SD card.
6. Execute the cross-compiled binaries on the ARM64 board.

Now your application runs on the ARM64 board.