NI-DAQmx Linux C Cross-Compile Tips

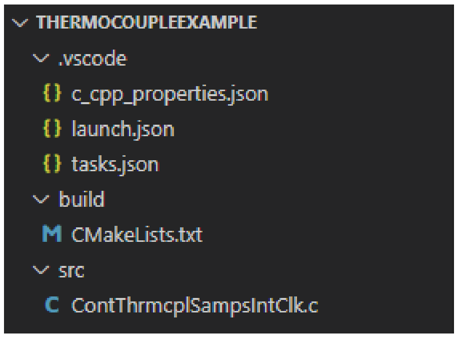
Overview

Before using this document, refer to the NI forum post, "NI Linux Real-Time Cross Compiling: Using the NI Linux Real-Time Cross Compile Toolchain with Visual Studio Code." After working through the examples, choose your desired C code from the "nidaqmx-c-examples" repository. The following steps utilize the "ContThrmcplSamps-IntClk.c" c source file from "\Analog In\Measure Temperature\Cont Thrmcpl Samples-Int Clk".

Note: this example was built with a Windows 10 host machine and cRIO-9040 (x64_Linux).

Steps

1. Install the NI-DAQmx driver on your host computer.
2. Install the NI-DAQmx driver on your NI Linux Real-Time target.
3. Install the correct GNU C & C++ Compile Tools for your NI Linux Real-Time target.
4. Create a copy of the cross-compile project template in a directory of your choosing in the host computer, or use the sample set included in */samplebuildfiles*.
5. Open the directory in VSCode.



```
THERMOCOUPLEEXAMPLE
  .vscode
    c_cpp_properties.json
    launch.json
    tasks.json
  build
    CMakeLists.txt
  src
    ContThrmcplSampsIntClk.c
```

6. Modify the c_cpp_properties.json file, as shown below.

```
{
    "env": {
        "compilerSysroots": "C:/build/18.0/x64/sysroots"
    },
    "configurations": [
        {
            "name": "NI Linux Real-Time x64",
            "compilerPath": "${compilerSysroots}/i686-nilrtsdk-mingw32/usr/bin/x86_64-
nilrt-linux/x86_64-nilrt-linux-gcc",
            "compilerArgs": [
                "--sysroot=${compilerSysroots}/core2-64-nilrt-linux/"
            ],
            "includePath": [
                "${workspaceFolder}/**",
                "C:/Program Files (x86)/National Instruments/NI-
DAQ/DAQmx ANSI C Dev/include",
                "C:/Program Files (x86)/National Instruments/NI-
DAQ/DAQmx ANSI C Dev/lib/msvc"
```

```
            ],
            "cStandard": "c17",
            "cppStandard": "c++17",
            "intelliSenseMode": "gcc-x64"
        }
    ],
    "version": 4
}
```

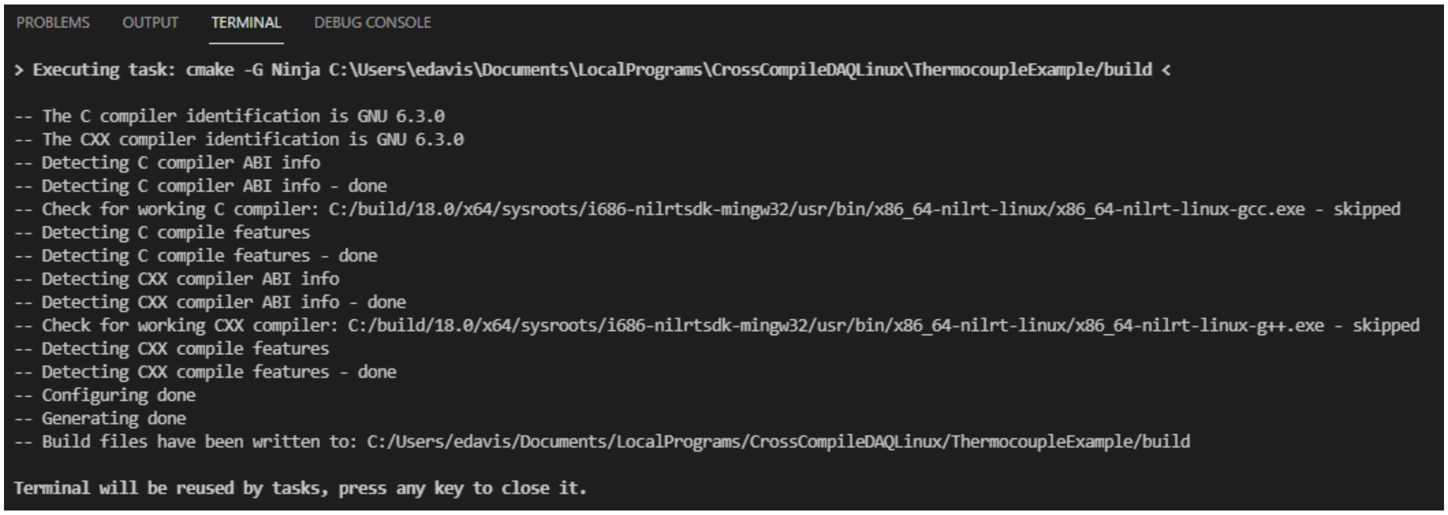
7. Modify the tasks.json file, as shown below.

```
{
    // See https://go.microsoft.com/fwlink/?LinkId=733558
    // for the documentation about the tasks.json format
    "version": "2.0.0",
    "tasks": [
        {
            "label": "CMake Generate Build Files",
            "type": "shell",
            "command": "cmake -G Ninja ${workspaceFolder}/build",
            "options": {
                "cwd": "${workspaceFolder}/build"
            },
            "problemMatcher": []
        },
        {
            "label": "Ninja",
            "type": "shell",
            "command": "ninja",
            "options": {
                "cwd": "${workspaceFolder}/build"
            },
            "problemMatcher": "$gcc"
        },
        {
            "label": "clean",
            "type": "shell",
            "command": "ninja clean",
            "options": {
                "cwd": "${workspaceFolder}/build"
            },
            "problemMatcher": []

        }
    ]
}
```

8. Add your Linux DAQmx example source code to the .src folder.
9. Modify the CMakeList.txt file, as shown below.

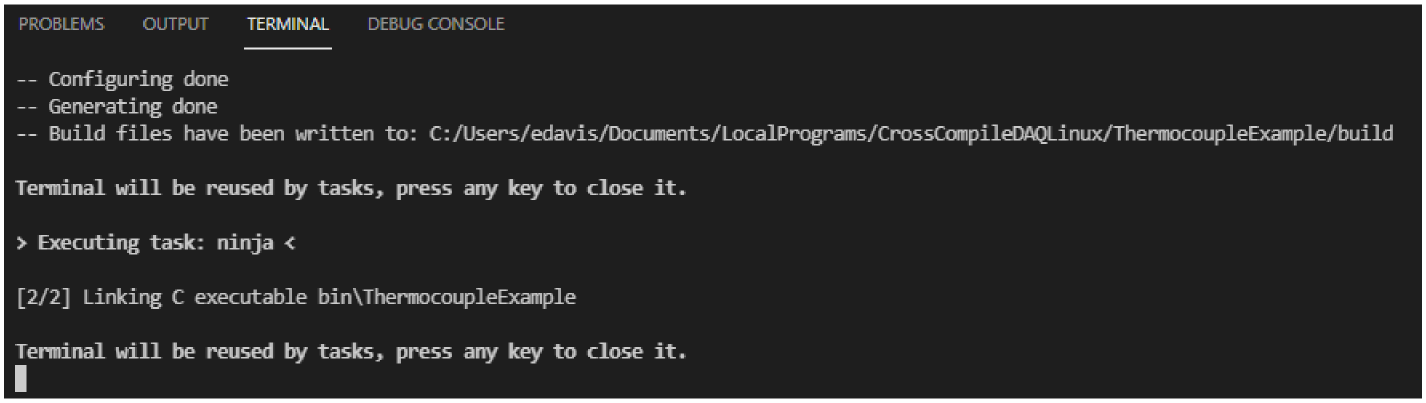
```
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR x86_64)
set(toolchain_path C:/build/18.0/x64/sysroots)
set(CMAKE_C_COMPILER ${toolchain_path}/i686-nilrtsdk-mingw32/usr/bin/x86_64-nilrt-
linux/x86_64-nilrt-linux-gcc.exe)
set(CMAKE_CXX_COMPILER ${toolchain_path}/i686-nilrtsdk-mingw32/usr/bin/x86_64-nilrt-
linux/x86_64-nilrt-linux-g++.exe)
set(CMAKE_SYSROOT ${toolchain_path}/core2-64-nilrt-linux)
set(CMAKE_<LANG>_STANDARD_INCLUDE_DIRECTORIES ${toolchain_path}/core2-64-nilrt-
linux/usr/include/c++/6.3.0 ${toolchain_path}
    /core2-64-nilrt-linux/usr/include/c++/6.3.0/x86_64-nilrt-linux)
set(CMAKE_<LANG>_FLAGS "-Wall -fmessage-length=0")
set(CMAKE_<LANG>_FLAGS_DEBUG "-O0 -g3")
set(CMAKE_<LANG>_FLAGS_RELEASE "-O3")
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_FIND_ROOT_PATH_MODE_PACKAGE ONLY)

# Project specific information
cmake_minimum_required(VERSION 3.7.2)
project(ProjectName) # NOTE: Edit ProjectName to a project name of your choosing
set(EXECUTABLE_OUTPUT_PATH bin)
set(CMAKE_BUILD_TYPE Debug)

set(HEADER_DIR "C:/Program\ Files\ (x86)/National\ Instruments/NI-
DAQ/DAQmx\ ANSI\ C\ Dev/include")
set(DAQMXLIBPATH "C:/Program\ Files\ (x86)/National\ Instruments/Shared/ExternalCompilerSuppo
rt/C/lib64/gcc")
add_executable(ProjectName ../src/SourceCodeName.c ${HEADER_DIR}/NIDAQMX.h) # NOTE: Replace P
rojectName and SourceCodeName to match your chosen project name and source code name
target_include_directories(ProjectName PUBLIC ${HEADER_DIR})
target_link_libraries(ProjectName PUBLIC ${DAQMXLIBPATH}/libnidaqmx.so)
```

10. From the Command Palette (Ctrl + Shift + P), select Tasks: Run Task, and then “CMake Generate Build Files.” This will run the task created to allow Visual Studio Code to invoke CMake, as shown below.



```
> Executing task: cmake -G Ninja C:\Users\edavis\Documents\LocalPrograms\CrossCompileDAQLinux\ThermocoupleExample/build <

-- The C compiler identification is GNU 6.3.0
-- The CXX compiler identification is GNU 6.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/build/18.0/x64/sysroots/i686-nilrtsdk-mingw32/usr/bin/x86_64-nilrt-linux/x86_64-nilrt-linux-gcc.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/build/18.0/x64/sysroots/i686-nilrtsdk-mingw32/usr/bin/x86_64-nilrt-linux/x86_64-nilrt-linux-g++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/edavis/Documents/LocalPrograms/CrossCompileDAQLinux/ThermocoupleExample/build

Terminal will be reused by tasks, press any key to close it.
```

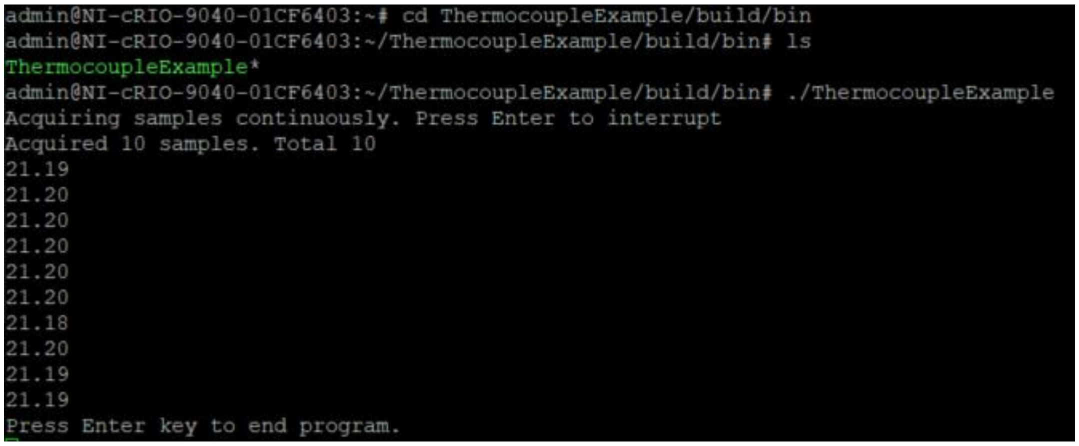
11. From the Tasks: Run Task, select Ninja to build the executable, as shown below.



PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/edavis/Documents/LocalPrograms/CrossCompileDAQLinux/ThermocoupleExample/build

Terminal will be reused by tasks, press any key to close it.

> Executing task: ninja <

[2/2] Linking C executable bin\ThermocoupleExample

Terminal will be reused by tasks, press any key to close it.
```

12. Copy the directory from your host computer to your target.
13. Through SSH, run the executable (located in /build/bin), as shown below.



```
admin@NI-cRIO-9040-01CF6403:~# cd ThermocoupleExample/build/bin
admin@NI-cRIO-9040-01CF6403:~/ThermocoupleExample/build/bin# ls
ThermocoupleExample*
admin@NI-cRIO-9040-01CF6403:~/ThermocoupleExample/build/bin# ./ThermocoupleExample
Acquiring samples continuously. Press Enter to interrupt
Acquired 10 samples. Total 10
21.19
21.20
21.20
21.20
21.20
21.20
21.18
21.20
21.19
21.19
Press Enter key to end program.
```