Mario Krizic

Cross-Compiling Forte for Ev3 using Docker

Note: This guide is probably not optimal as I only have limited experience with the tools used here. In case there is a better solution, please edit this document accordingly. This guide is meant for Windows (64 bit) users, but the steps may also be helpful for Linux. The commands in this guide start with '>'. Do not copy this character, as this will cause all sorts of errors.

The following steps are based on the guide found here: https://www.ev3dev.org/docs/tutorials/using-docker-to-cross-compile/:

- First get the latest Docker installation: https://www.docker.com/products/docker-engine
- Open the command line and run the following command: > docker pull ev3dev/debian-stretch-cross
- As long filenames are tedious to work with, rename it with the command: > docker tag ev3dev/debian-stretch-cross ev3cc

This image contains the toolchain you will need to compile forte for the Ev3. I recommend to set the available memory in the Docker settings to 4GB and the Swap to 2GB.

Now that you have the docker image, create a folder on your host (4diac is used here) and download the forte file from git using the following link: https://git.eclipse.org/c/4diac/org.eclipse.4diac.forte.git

Use "> git clone <remote> name" to get this repository on your local machine. There should be a folder with the given name in your .git folder now.

```
C:\Users\Mario>git clone https://git.eclipse.org/r/4diac/org.eclipse.4diac.forte forte
Cloning into 'forte'...
remote: Counting objects: 21, done
remote: Total 12125 (delta 0), reused 12125 (delta 0) eceiving objects: 100% (12125/12125), 8.
Receiving objects: 100% (12125/12125), 8.12 MiB | 166.00 KiB/s, done.
Resolving deltas: 100% (5001/5001), done.
Checking out files: 100% (1717/1717), done.
```

It is recommended using the 1.10.x branch to get the latest version. First we cd into the forte folder and then we use the checkout command:

```
C:\Users\Mario\forte>git checkout 1.10.x
Switched to a new branch '1.10.x'
Branch '1.10.x' set up to track remote branch '1.10.x' from 'origin'.
```

And just to be safe, we can perform a pull request to get the latest changes. It might me necessary to enter your git email and or name with the commands:

"> git config –global user.email <Your Email>"

“> git config –global user.name <Your Name>”

Now you can pull with the command “> git pull origin master”. This should get all the latest changes.

If any errors arise during the compilation of forte, you may look for the latest release (for example 1.10.2 as I am writing this) and repeat this process for that branch.

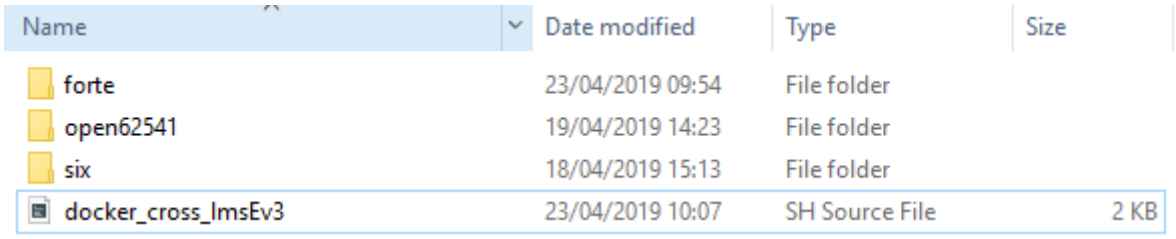
```
C:\Users\Mario\forte>git pull origin master
From https://git.eclipse.org/r/4diac/org.eclipse.4diac.forte
 * branch            master     -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 tests/core/interalvartests.cpp | 96 ++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 96 insertions(+)
 create mode 100644 tests/core/interalvartests.cpp
```

Compile Forte with OPC_UA for Ev3 using docker_cross_lmsEv3.sh

I have written a script file, which will automatically do all steps for you, but in case it does not work, you can compile it manually by using the steps below.

First, create a folder containing the six (the python module), the forte source and the open62541 source.

It is important to rename them accordingly, otherwise the script will not run. It also makes it easier to support upcoming releases. Add the docker_cross_lmsEv3.sh file into the folder.



Open the command line and run > docker run --rm -it -v <path_containing_files>:<forte_container_path> -w <forte_container_path> ev3cc

First, you will need to run these commands:

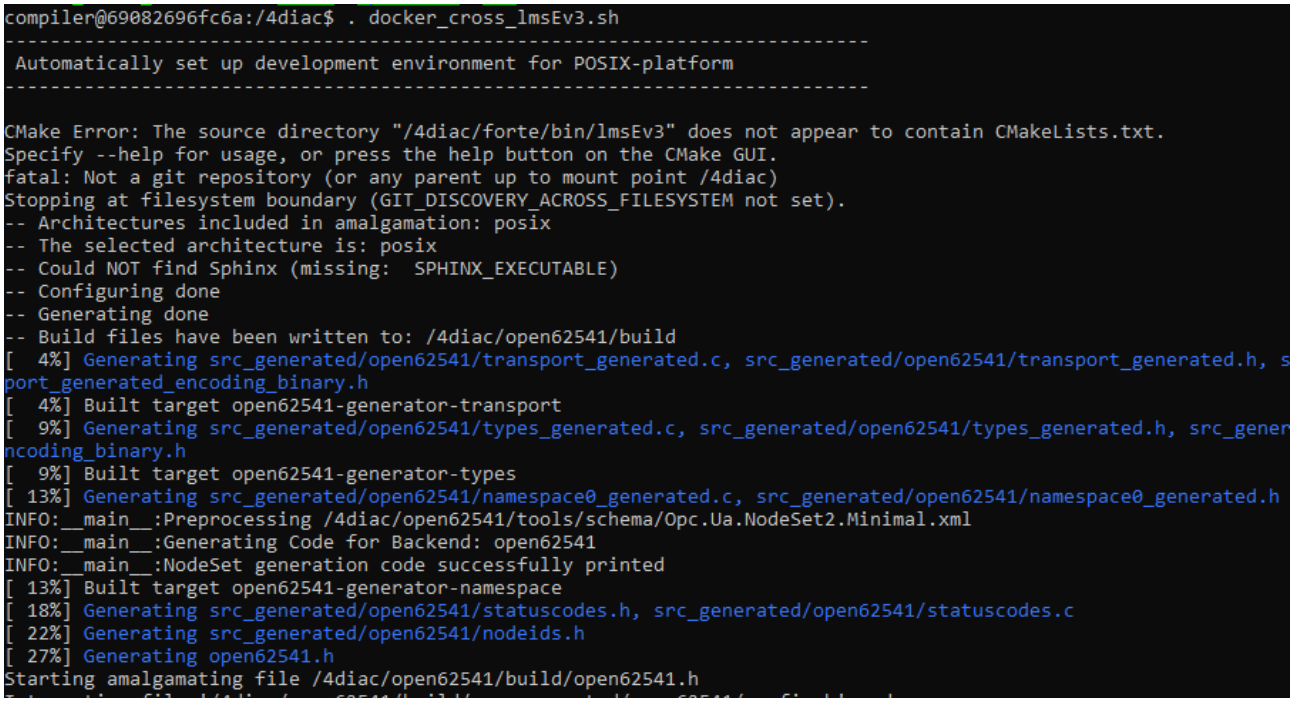
> sudo apt-get update

> sudo apt-get install python

Go to the folder containing the setup.py file for the six module, in my case 4diac/six/six-1.12.0 and run > sudo python setup.py install

Now, go back to the 4diac folder containing the docker_cross_lmsEv3.sh script and type > . docker_cross_lmsEv3.sh

This will compile open and forte.



You can find the compiled forte executable in the /4diac/forte/bin/lmsEv3/src folder.

Compile Forte

Open the command line again and run > docker run --rm -it -v <forte_host>:<forte_container> -w <forte_container> ev3cc

--rm removes the container after it is closed.

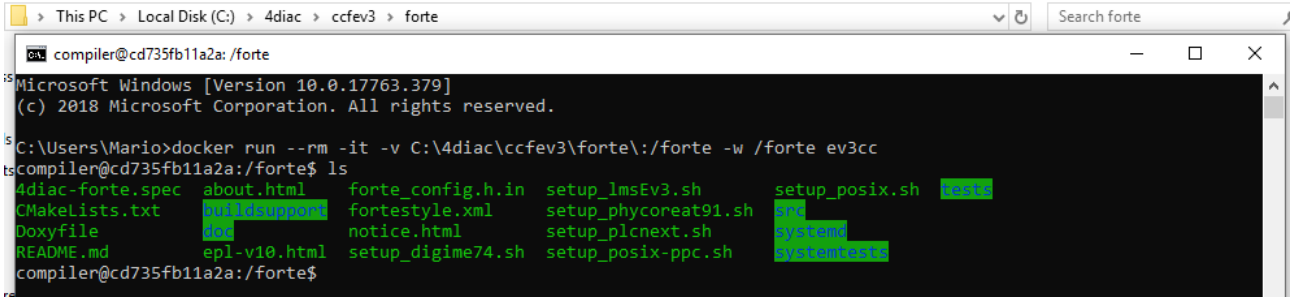
-it starts an interactive session

-v creates a shared folder inside the container and use <host_path>:<container_path> as a parameter.

-w is the workspace directory. When you execute this command you will start in this folder. In our case this is /forte

ev3cc is the image from which the container will be generated.

The command may look something like this:



As the scripts currently contain some errors, you will need a good editor like notepad++. Open the setup_lmsEv3.sh file on your host system using you favourite editor.

The first error you will have to fix is the DCMAKE_TOOLCHAIN_FILE parameter.

Change it to: “/home/compiler/toolchain-armel.cmake”. Then, you will need to convert the line endings to unix. On notepad++ this is done by Edit → EOL Conversion → Unix(LF).

The File should now look something like this:

```
#!/bin/bash
echo "------------------------------------------------------------------------------"
echo "Automatically set up development environment for POSIX-platform"
echo "------------------------------------------------------------------------------"
echo ""
echo "Includes 64bit-datatypes, float-datatypes, Ethernet-Interface,"
echo "ASN1-encoding, ..."
echo ""
echo "To include tests set directories for boost-test-framework and "
echo "set FORTE_TESTS-option to 'ON'"
echo ""
echo "------------------------------------------------------------------------------"

export forte_bin_dir="bin/lmsEv3"

#set to boost-include directory
export forte_boost_test_inc_dirs=""
#set to boost-library directory
export forte_boost_test_lib_dirs=""

if [ ! -d "$forte_bin_dir" ]; then
  mkdir -p "$forte_bin_dir"
fi

if [ -d "$forte_bin_dir" ]; then
  
  echo "For building forte go to $forte_bin_dir and execute \"make\""
  echo "forte can be found at ${forte_bin_dir}/src"
  echo "forte_tests can be found at ${forte_bin_dir}/tests"
  
  
  cd "./$forte_bin_dir"
  
  cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE="/home/compiler/toolchain-armel.cmake" -DFORTE_MODULE_LMS_EV3=ON -DFORTE_
else
  echo "unable to create ${forte_bin_dir}"
  exit 1
fi
```

If you are still getting errors, try replacing the file with the setup_lmsEv3.sh file on my GitHub.

Now run the script and the build should be configured and generated.

```
compiler@cd735fb11a2a:/forte$ ./setup_lmsEv3.sh
------------------------------------------------------------------------------
 Automatically set up development environment for POSIX-platform
------------------------------------------------------------------------------

 Includes 64bit-datatypes, float-datatypes, Ethernet-Interface,
 ASN1-encoding, ...

 To include tests set directories for boost-test-framework and
 set FORTE_TESTS-option to 'ON'

------------------------------------------------------------------------------
For building forte go to bin/lmsEv3 and execute "make"
forte can be found at bin/lmsEv3/src
forte_tests can be found at bin/lmsEv3/tests
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/arm-linux-gnueabi-gcc
-- Check for working C compiler: /usr/bin/arm-linux-gnueabi-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/arm-linux-gnueabi-g++
-- Check for working CXX compiler: /usr/bin/arm-linux-gnueabi-g++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
```

Then cd into the bin/lmsEv3 folder and run make.

```
compiler@cd735fb11a2a:/forte$ cd bin/lmsEv3
compiler@cd735fb11a2a:/forte/bin/lmsEv3$ make
[  1%] Generating src_gen/pctimeha_gen.cpp
[  1%] Generating src_gen/forte_thread_gen.cpp
[  1%] Generating src_gen/forte_sync_gen.cpp
[  1%] Generating src_gen/forte_sem_gen.cpp
[  1%] Generating src_gen/genforte_printer_gen.cpp
[  1%] Generating src_gen/timespec_utils_gen.cpp
[  1%] Generating src_gen/fdselecthand_gen.cpp
[  1%] Generating src_gen/bsdsocketinterf_gen.cpp
[  1%] Generating src_gen/timerha_gen.cpp
[  2%] Generating src_gen/devlog_gen.cpp
```

This will generate the forte file in the bin/lmsEv3/src folder.

Compile Forte with OPC_UA:

Once you completed the previous steps, you can easily compile forte with OPC_UA with the following steps:

Download open62541 from https://github.com/open62541/open62541 and add it into the same folder as the forte files.

Next, as the installation of pip will probably delete your arm compilers, download the python six module files from https://pypi.org/project/six/#files (tar.gz) and put them into the same folder where open62541 and forte are.

Start a new docker session and mount the forte parent folder like this:

```
C:\Users\Mario>docker run --rm -it -v C:\4diac\ccfev3\:/4diac -w /4diac ev3cc
compiler@e573f574d67f:/4diac$
```

First, install python using:

> sudo apt-get update

> sudo apt-get install python

Unzip the six folder and navigate to the folder that contains the setup.py file. Then run the command > sudo python setup.py install

```
compiler@e573f574d67f:/4diac/six/six-1.12.0$ ls
CHANGES  MANIFEST.in  README.rst  documentation  setup.py      six.py   test_six.py
LICENSE  PKG-INFO     build       setup.cfg      six.egg-info  six.pyc
compiler@e573f574d67f:/4diac/six/six-1.12.0$ sudo python setup.py install
/usr/lib/python2.7/distutils/dist.py:267: UserWarning: Unknown distribution option: 'python_requires'
  warnings.warn(msg)
/usr/lib/python2.7/distutils/dist.py:267: UserWarning: Unknown distribution option: 'tests_require'
  warnings.warn(msg)
running install
running build
running build_py
running install_lib
copying build/lib.linux-x86_64-2.7/six.py -> /usr/local/lib/python2.7/dist-packages
byte-compiling /usr/local/lib/python2.7/dist-packages/six.py to six.pyc
running install_egg_info
Writing /usr/local/lib/python2.7/dist-packages/six-1.12.0.egg-info
compiler@e573f574d67f:/4diac/six/six-1.12.0$
```

Now we can compile open62541. Go to the open62541 folder and make a new build directory (> mkdir build && cd build), then run:

> cmake -DBUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=Debug -DUA_ENABLE_AMALGAMATION=ON ..

This will cause an error, where CMake does not know the path to the compiler, but now we have a cmake cache. Open the open62541/build folder (you can do this on your host machine as well) and then you will need to perform some edits in the CmakeCache.txt file.

Set the paths to the C/C++ compiler by setting the values of these two fields to point to them:

CMAKE_C_COMPILER:FILEPATH=/usr/bin/arm-linux-gnueabi-gcc

CMAKE_CXX_COMPILER:FILEPATH=/usr/bin/arm-linux-gnueabi-g++

Then run the previous cmake command again and it should configure and generate successfully.

Finally run make -j to complete the open62541 setup.

```
compiler@e573f574d67f:/4diac/open62541/build$ cmake -DBUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=Debug -DUA_ENABLE_AMALGAMATION=ON ..
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/arm-linux-gnueabi-gcc
-- Check for working C compiler: /usr/bin/arm-linux-gnueabi-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/arm-linux-gnueabi-g++
-- Check for working CXX compiler: /usr/bin/arm-linux-gnueabi-g++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Found PythonInterp: /usr/bin/python (found version "2.7.9")
-- Could NOT find Git (missing:  GIT_EXECUTABLE)
-- Could NOT find Sphinx (missing:  SPHINX_EXECUTABLE)
-- Configuring done
-- Generating done
-- Build files have been written to: /4diac/open62541/build
compiler@e573f574d67f:/4diac/open62541/build$ make -j
Scanning dependencies of target open62541-generator-statuscode
Scanning dependencies of target open62541-generator-namespace
Scanning dependencies of target open62541-generator-types
Scanning dependencies of target open62541-generator-transport
[ 11%] [ 16%] Generating src_generated/ua_namespace0.c, src_generated/ua_namespace0.h
[ 22%] [ 22%] Generating src_generated/ua_statuscode_descriptions.c
```

Next, create, then go to the forte build folder and run the following command:

> cmake -DCMAKE_BUILD_TYPE=Debug -DFORTE_ARCHITECTURE=Posix -DFORTE_MODULE_CONVERT=ON -DFORTE_COM_ETH=ON -DFORTE_MODULE_IEC61131=ON -DFORTE_COM_OPC_UA=ON -DFORTE_COM_OPC_UA_INCLUDE_DIR=/4diac/open62541/build -DFORTE_COM_OPC_UA_LIB_DIR=/4diac/open62541/build/bin -DFORTE_COM_OPC_UA_LIB=libopen62541.so ..

This will again cause an error. Edit the CmakeCache.txt file in the forte/build folder to use the armel gcc and g++ compilers analog to the way described above with open. And run the cmake command again.

Finally run make again and forte should be in the forte/build/src folder.

```
compiler@e573f574d67f:/4diac/forte/build$ cmake -DCMAKE_BUILD_TYPE=Debug -DFORTE_ARCHITECTURE=Posix -DFORTE_MODULE_CONVERT=ON -DFORTE_COM_ETH=ON -DFORTE_MODULE_IEC61131=ON -DFORTE_COM_OPC_UA=ON -DF
ORTE_COM_OPC_UA_INCLUDE_DIR=/4diac/open62541/build -DFORTE_COM_OPC_UA_LIB_DIR=/4diac/open62541/build/bin -DFORTE_COM_OPC_UA_LIB=libopen62541.so ..
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/arm-linux-gnueabi-gcc
-- Check for working C compiler: /usr/bin/arm-linux-gnueabi-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/arm-linux-gnueabi-g++
-- Check for working CXX compiler: /usr/bin/arm-linux-gnueabi-g++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
No Log Level was specified. Setting Log Level to Debug default LOGINFO.
-- FORTE_EXTERNAL_MODULES_DIRECTORY:
Building executable
-- Configuring done
-- Generating done
-- Build files have been written to: /4diac/forte/build
compiler@e573f574d67f:/4diac/forte/build$ make
[  1%] Generating src_gen/pctimeha_gen.cpp
[  1%] Generating src_gen/forte_thread_gen.cpp
[  1%] Generating src_gen/forte_sync_gen.cpp
[  1%] Generating src_gen/forte_sem_gen.cpp
```

How to proceed:

When finished compiling forte for Ev3, you will want to transfer your forte files to the brick. For Windows use WinSCP. You can find the installation here: https://winscp.net/eng/docs/guide_install

Drag and drop the forte files here, then connect with PuTTY via SSH to the brick. Find the forte file and run it with “> ./forte”. You can find PuTTY in case you don’t have it yet here: https://www.putty.org/

```
robot@ev3dev:~/forte/bin/lmsEv3/src$ ./forte
INFO: T#6: FORTE is up and running
INFO: T#19: Using default bootfile location: forte.fboot
INFO: T#23: Boot file forte.fboot could not be opened. Skipping...
```

Sometimes you may get some errors. Use the command “>file forte” to verify that you have compiled forte for the correct architecture (ARM). If the architecture is correct, try using sudo or modify the permissions before proceeding with further troubleshooting.

```
compiler@e573f574d67f:/4diac/forte/build/src$ file forte
forte: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=3aa2578b0f3aad60bcdd3413df560e2f46a1a039, not
stripped
```

This guide is currently incomplete, as it is missing custom function blocks and some other features. They will be added as soon as possible.

Troubleshooting:

The following sections contains some infos on errors I have encountered and how to resolve them.

If forte reports some errors in the opc_ua handler or opc_ua_client_handler, build forte without opc_ua first and repeat the steps. Some configurations may be missing.

Check the folder hierarchy and make sure you are calling the scripts from the correct locations.

Calling the docker script may cause an error saying that the folder does not appear to contain a CmakeLists.txt file. You can ignore this message.