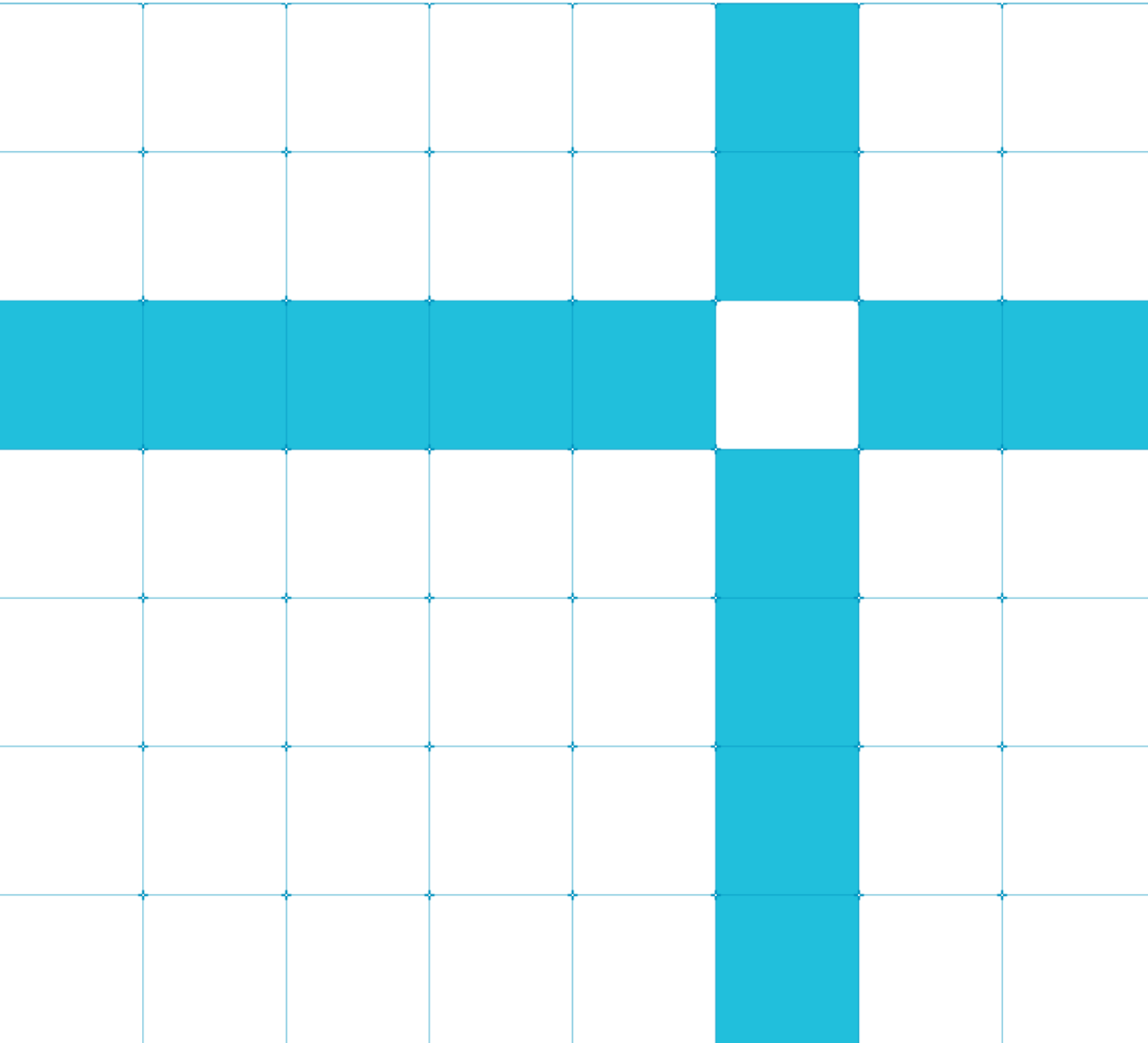
arm

Building EDKII UEFI firmware for Arm Platforms

Version 1.0



Building EDKII UEFI firmware for Arm Platforms

Release Information

Document History

Version	Date	Confidentiality	Change
1.0	25th August 2020	Non-confidential	First release.

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents

1 Overview

This guide shows you how to build Unified Extensible Firmware Interface (UEFI) firmware for the Arm Fixed Virtual Platform (FVP) Model and Juno Development Platform on either a Linux or Windows development PC.

UEFI is a specification that defines an interface between the firmware and an Operating System (OS). UEFI defines the firmware interfaces and boot services that are required for booting a standards-based OS. UEFI also defines run-time services, for example, time, variable that an OS can invoke at runtime.

The TianoCore EFI Development Kit II (EDKII) project provides an implementation of the UEFI firmware. EDKII is an open-source project that provides a feature-rich, cross-platform firmware development environment for UEFI and UEFI Platform Initialization (PI) specifications. UEFI Forum develops and maintains the specifications.

At the end of this guide you will:

- Be familiar with the EDKII development and build environment
- Be able to build the reference firmware for FVP and the Arm Juno development platform
- Have the required EDKII firmware binaries for following the tutorials that are found in Next steps to run the firmware on an FVP or Arm Juno development platform

1.1. Before you begin

There are four different sections in this guide. They all show you how to build firmware on different platforms, so you may not need to read and work through each section. Each section contains its own software and hardware requirements. Check that these requirements are in place before you follow the instructions in that section of the guide.

Before you work through any section of the guide, you must have git installed. How to install git is shown in the following code:

```
$ sudo apt install git
$ git --version
git version 2.17.1
```

You also need to follow the instructions in Set up the development environment.

2 Set up the development environment

This section of the guide shows you how to set up the environment on either a Linux or Windows development PC before starting to build EDKII firmware.

We use the following source repositories in this guide:

Project	Repository	Repository URL	Description
Tianocore	tianocore\edk2	https://github.com/tianocore/edk2.git	The `edk2` repository contains the firmware development environment and required packages for building the UEFI firmware.
Tianocore	tianocore\edk2-platforms	https://github.com/tianocore/edk2-platforms.git	The `edk2`-platforms repository contains the platform workspace and associated modules.
ACPICA	acpica\acpica	https://github.com/acpica/acpica.git	The Advanced Configuration and Power Interface (ACPI) Component Architecture (ACPICA) tools provide an open-source implementation of the iASL compiler.

The following steps show you how to set up your development environment:

1. Launch a terminal window and create a workspace folder on your development PC called `source`. Set the `WORKSPACE` environment variable to point to this folder.

   This can be seen in the following code:

In a Linux bash shell:

```
cd <Directory where you want to work>
mkdir source
cd source
export WORKSPACE=$PWD
```

In a Windows command prompt:

```
cd <Directory where you want to work>
mkdir source
cd source
set WORKSPACE=%CD%
```

2. Clone the source code repositories. In the terminal window, change directory to your workspace (`source`) folder and run the following commands:

```
git clone https://github.com/tianocore/edk2-platforms.git
git clone https://github.com/acpica/acpica.git
git clone https://github.com/tianocore/edk2.git
```

3. Go to the `edk2` folder and update the submodules:

```
cd edk2
git submodule update --init
cd ..
```

3 Build firmware on a Linux host

The steps in this section show you how to build the firmware on a Linux host.

3.1. System requirements

Before you can build the firmware on a Linux host, check that you have:

- The correct PC hardware:
  - 64-bit development PC
  - At least 10GB of free disk space
- The correct Ubuntu version by typing the following in the terminal window:

```
$ uname -srvmpio
Linux 4.18.0-15-generic #16~18.04.1-Ubuntu SMP Thu Feb 7 14:06:04 UTC 2019 x86_64
x86_64 x86_64 GNU/Linux
```

You must have Ubuntu 18.04 desktop.

- The following tools installed on your development PC. The instructions for each tool show you how to install that tool:

Tool	Description	Install instructions

Python 2.7
 	
or
 	
Python 3
	Python interpreter	Python 2.7
		
		`$ sudo apt install python`
		`$ python -V`
		`Python 2.7.15rc1`
		
		Python 3
		
		`$ sudo apt install python3 python3-distutils`

uuid-dev	Required for including uuid/uuid.h	`$ sudo apt install uuid-dev`

Build-essential	This package installs make, gcc, g++.	`$ sudo apt install build-essential`
		`$ make -v`
		`GNU Make 4.1`
		`$ gcc --version`
		`gcc (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0`
		`$ g++ --version`
		`g++ (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0`

bison	A parser generator required by ACPICA tools	`$ sudo apt install bison`

Tool	Description	Install instructions
flex	A fast-lexical analyzer generator required by ACPICA tools	`$ sudo apt install flex`

The following steps show you how to install the required Arm toolchain and how to build the ACPICA tools:

1. Install the required Arm toolchain from GNU-A downloads on Arm developer.

   Go to the relevant section of the page for your development PC architecture and select the little-endian AArch64 ELF bare-metal target (aarch64-elf) GCC cross compiler. For example, for an x86_64 development PC, download gcc-arm-9.2-2019.12-x86_64-aarch64-none-elf.tar.xz. Create a folder called `toolchain` under the workspace folder and extract the toolchain, as shown in the following commands:

```
$ mkdir $WORKSPACE/toolchain
$ cd $WORKSPACE/toolchain
$ wget https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-a/9.2-2019.12/binrel/gcc-arm-9.2-2019.12-x86_64-aarch64-none-elf.tar.xz
$ tar xf gcc-arm-9.2-2019.12-x86_64-aarch64-none-elf.tar.xz
```

2. Run the following commands in the terminal window, to build the ACPICA tools:

Note: The ACPICA tools implement the latest iASL compiler.

```
$ cd $WORKSPACE
$ make -C $WORKSPACE/acpica
```

3.2. Building EDKII firmware

The following steps show you how to build the firmware image:

Note: Run the commands in the terminal window.

1. Use the following commands to set up the environment variables:

```
$ export GCC5_AARCH64_PREFIX=$WORKSPACE/toolchain/gcc-arm-9.2-2019.12-x86_64-aarch64-none-elf/bin/aarch64-none-elf-
$ export PACKAGES_PATH=$WORKSPACE/edk2:$WORKSPACE/edk2-platforms
$ export IASL_PREFIX=$WORKSPACE/acpica/generate/unix/bin/
```

2. Select the Python version that you want to use and set the `PYTHON_COMMAND` environment variable.

For Python 2.7:

```
$ export PYTHON_COMMAND=/usr/bin/python
```

For Python 3:

```
$ export PYTHON_COMMAND=/usr/bin/python3
```

3. Configure the EDKII development environment by running the `edk2setup` bash script that is shown in the following command:

```
$ source edk2/edksetup.sh
```

4. Build the `BaseTools` as shown in the following command:

```
$ make -C edk2/BaseTools
```

5. Follow either Build the firmware for Arm FVP Base AEMv8A-AEMv8A model platform or Build firmware for Arm Juno platform. These show you how to build the firmware for your choice of Arm development platform.

3.3. Build the firmware for Arm FVP Base AEMv8A-AEMv8A model platform

To build the firmware for FVP Base AEMv8A-AEMv8A platform, run the following commands:

```
$ build -a AARCH64 -t GCC5 -p Platform/ARM/VExpressPkg/ArmVExpress-FVP-AArch64.dsc -b
DEBUG

$ build -a AARCH64 -t GCC5 -p Platform/ARM/VExpressPkg/ArmVExpress-FVP-AArch64.dsc -b
RELEASE
```

The firmware binaries are at the following location:

```
$WORKSPACE/Build/ArmVExpress-FVP-AArch64/<DEBUG|RELEASE>_GCC5/FV/FVP_AARCH64_EFI.fd
```

3.4. Build the firmware for Arm Juno platform

To build the firmware for Arm Juno platform, run the following commands:

```
$ build -a AARCH64 -t GCC5 -p Platform/ARM/JunoPkg/ArmJuno.dsc -b DEBUG

$ build -a AARCH64 -t GCC5 -p Platform/ARM/JunoPkg/ArmJuno.dsc -b RELEASE
```

The firmware binaries are at the following location:

```
$WORKSPACE/Build/ArmJuno/<DEBUG|RELEASE>_GCC5/FV/BL33_AP_UEFI.fd
```

4 Build firmware on Linux using Linaro uefi-tools

Linaro provides a set of useful scripts to build the TianoCore EDKII UEFI firmware on a Linux development PC. This section of the guide outlines the additional steps that are required for building firmware using the uefi-tools from Linaro.

4.1. System requirements

Before this can be done, you must have the following:

Tool	Description	Install instructions
Python 2.7 and Python 3	The scripts in the uefi-tools package require Python 2.7, whereas EDKII defaults to use Python3.	`$ sudo apt install python python3 python3-distutils`
Arm cross compiler toolchain	The uefi-tools use the gcc-aarch64-linux-gnu toolchain.	`$ sudo apt install gcc-aarch64-linux-gnu`
ACPICA tools	The uefi-tools use the iASL compiler that is packaged with the ACPICA-tools and distributed with the OS.	`$ sudo apt install acpica-tools`
Cloning the uefi-tools source code repository	In a command window, make sure you are in workspace (`source`) folder.	`git clone https://git.linaro.org/uefi/uefi-tools.git`

Follow either Build the firmware for Arm FVP Base AEMv8A-AEMv8A model platform or Build firmware for Arm Juno platform. These show you how to build the firmware for your choice of Arm development platform.

4.2. Build the firmware for Arm FVP Base AEMv8A-AEMv8A model platform

To build the firmware for FVP Base AEMv8A-AEMv8A platform, run the following commands:

```
$ cd $WORKSPACE
$ ./uefi-tools/edk2-build.sh -b DEBUG fvp -v
$ ./uefi-tools/edk2-build.sh -b RELEASE fvp -v
```

The firmware binaries are at the following location:

```
$WORKSPACE/Build/ArmVExpress-FVP-AArch64/<DEBUG|RELEASE>_GCC5/FV/FVP_AARCH64_EFI.fd
```

4.3. Build the firmware for Arm Juno platform

To build the firmware for Arm Juno platform, run the following commands:

```
$ cd $WORKSPACE
$ ./uefi-tools/edk2-build.sh -b DEBUG juno -v
$ ./uefi-tools/edk2-build.sh -b RELEASE juno -v
```

The firmware binaries are at the following location:

```
$WORKSPACE/Build/ArmJuno/<DEBUG|RELEASE>_GCC5/FV/BL33_AP_UEFI.fd
```

The uefi-tools have additional options for building firmware. To view these options, type the following command in a terminal window:

```
$ ./uefi-tools/edk2-build.sh -h
```

5 Build firmware on Windows using Windows Subsystem for Linux

To build the EDKII firmware using Windows Subsystem for Linux, follow the instructions in Build firmware on a Linux host.

5.1. System requirements

Before you can build the firmware on Windows using Windows Subsystem for Linux, check that you have:

- The correct PC hardware:
  - A x64 development PC with Windows 10 (Version 1809 - OS Build 17763.316).
  - At least 10GB of free disk space.
- The correct Ubuntu version by typing the following command:

```
$ uname -srvmpio
Linux 4.4.0-17763-Microsoft #253-Microsoft Mon Dec 31 17:49:00 PST 2018 x86_64
x86_64 x86_64 GNU/Linux
```

You must have Windows Subsystem for Linux and select Ubuntu 18.04 LTS. You can install this from the Microsoft Store.

6 Build firmware on a x64 Windows host

This section shows you how to build firmware on a x64 Windows host.

6.1. System requirements

Before you can build the firmware on a x64 Windows host, check that you have:

- The correct PC hardware:
  - A 64-bit development PC
  - At least 10GB of free disk space
- That you have at least Windows 10 desktop (Version 1809, OS Build 17763.316) or newer installed.
- The following tools installed on your development PC. The instructions for each tool show you how to install that tool:

Tool	Description	Install instructions
Python 2.7 or Python 3	Python interpreter	Go to https://www.python.org/downloads/windows/ Choose the latest Python 2.7 or 3.8.3 release. Download and run the `Windows x86_64 MSI installer` If needed, add the Python executable to your path by executing the following command: `> set PATH=<Path_to_the_python_executable>;%PATH%`
Git	Git source control tool	Go to https://git-scm.com/download/win Download and run the `64-bit Git for Windows Setup.`
ASL tools	`iASL` compiler and other tools for the ASL language	Go to https://www.acpica.org/downloads/binary-tools Download the `iASL Compiler and Windows ACPI Tools` Extract the content and place it at `C:\ASL\` Check that the compiler is at the right place by executing: `> C:\ASL\iasl.exe -v`

Microsoft Visual Studio 2017 professional	Microsoft IDE and compiler toolchain	Go to https://visualstudio.microsoft.com/downloads/ Download and install Visual Studio 2017 Professional.
echo tool	Echo	See the subsection called Workaround for echo command.

6.2. Arm cross compiler toolchain

The following instructions show you how to install the Arm toolchain Windows (i686-mingw32) hosted cross compilers.

1. Select the latest toolchain for AArch64 bare-metal target (aarch64-none-elf) GCC cross compiler.

For example: Download gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-elf.tar.xz

2. Create a folder called toolchain under the workspace folder, for example `source\toolchain` and extract the toolchain to this folder using an appropriate archiver utility. For example, 7zip. The toolchain folder tree will look like the following:

```
toolchain
+---gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-elf
|   +---aarch64-none-elf
|   +---bin
|   +---include
|   +---lib
|   +---libexec
|   +---share
```

6.3. Workaround for the echo command

EDKII needs a workaround related to the echo command. A script replacing the Windows echo executable must be created, with the name `echo.BAT`. The following instructions show how you can do this:

1. Create a file named `"echo.BAT"` in the folder of your choice.
2. Paste the following lines inside the file:

```
rem %~f0  echo.BAT  %*
rem This file exists to overcome a problem in the EDKII build where
rem build_rule.template invokes a command as:
rem     "$(OBJCOPY)" $(OBJCOPY_FLAGS) ${dst}
rem When OBJCOPY is set to echo, this results in the following error:
rem     "echo" objcopy not needed for m:\...\PCD\Dxe\Pcd\DEBUG\PcdDxe.dll
rem And CMD.EXE fails to find the DOS echo command because of the quotes
@echo %*
@goto :EOF
```

3. Add the file to your `PATH` by executing the following:

```
> set PATH=<Path_to_the_echo_file>;%PATH%
```

6.4. Building EDKII firmware

To build the firmware image, follow these steps:

1. Set up the environment variables:

```
> set GCC5_AARCH64_PREFIX=%WORKSPACE%\toolchain\gcc-arm-9.2-2019.12-mingw-w64-i686-
aarch64-none-elf\bin\aarch64-none-elf-
> set PACKAGES_PATH=%WORKSPACE%\edk2;%WORKSPACE%\edk2-platforms
```

Select the Python version that you want to use, and set the `PYTHON_COMMAND` environment variable to your Python executable:

```
> set PYTHON_COMMAND=<Path_to_your_Python_executable>\<Python_executable>.exe
```

Configure the EDKII development environment by running the `edksetup.bat` script. This can be done with the following command:

```
> call %WORKSPACE%\edk2\edksetup.bat [Rebuild | ForceRebuild]
```

Note: If the `BaseTools` have already been built, the `Rebuild` option can be skipped. Also, the `ForceRebuild` option can be used to do a clean build of the Base tools.

6.5. Build the firmware for Arm FVP Base AEMv8A-AEMv8A model platform

To build the firmware for FVP Base AEMv8A-AEMv8A platform, run the following commands:

```
> build -a AARCH64 -t GCC5 -p Platform\ARM\VExpressPkg\ArmVExpress-FVP-AArch64.dsc -b
DEBUG
> build -a AARCH64 -t GCC5 -p Platform\ARM\VExpressPkg\ArmVExpress-FVP-AArch64.dsc -b
RELEASE
```

The firmware binaries are at the following location:

```
%WORKSPACE%\Build\ArmVExpress-FVP-AArch64\<DEBUG|RELEASE>_GCC5\FV\FVP_AARCH64_EFI.fd
```

6.6. Build the firmware for Arm Juno platform

To build the firmware for Arm Juno platform, run the following command:

```
> build -a AARCH64 -t GCC5 -p Platform\ARM\JunoPkg\ArmJuno.dsc -b DEBUG
> build -a AARCH64 -t GCC5 -p Platform\ARM\JunoPkg\ArmJuno.dsc -b RELEASE
```

The firmware binaries are at the following location:

```
%WORKSPACE%\Build\ArmJuno\<DEBUG|RELEASE>_GCC5\FV\BL33_AP_UEFI.fd
```

7 Related information

Here are some resources related to material in this guide:

- Arm architecture and reference manuals - Find technical manuals and documentation relating to this guide and other similar topics.
- Arm Community - Ask development questions and find articles and blogs on specific topics from Arm experts.
- ACPI Component Architecture
- Arm Fixed Virtual Platforms
- Getting Started with EDKII
- Juno Development Board

8 Next steps

This guide has outlined the steps to build UEFI firmware for Arm development platforms.

To build Trusted Firmware-A (TF-A), a reference implementation of Secure world firmware, see the TF-A Getting Started guide.

The steps to run the firmware on an Arm Fixed Virtual Platform (FVP) or Arm Juno development platform are covered in Arm Development Platforms on TrustedFirmware.