



Arm Keil Studio Cloud

1.5

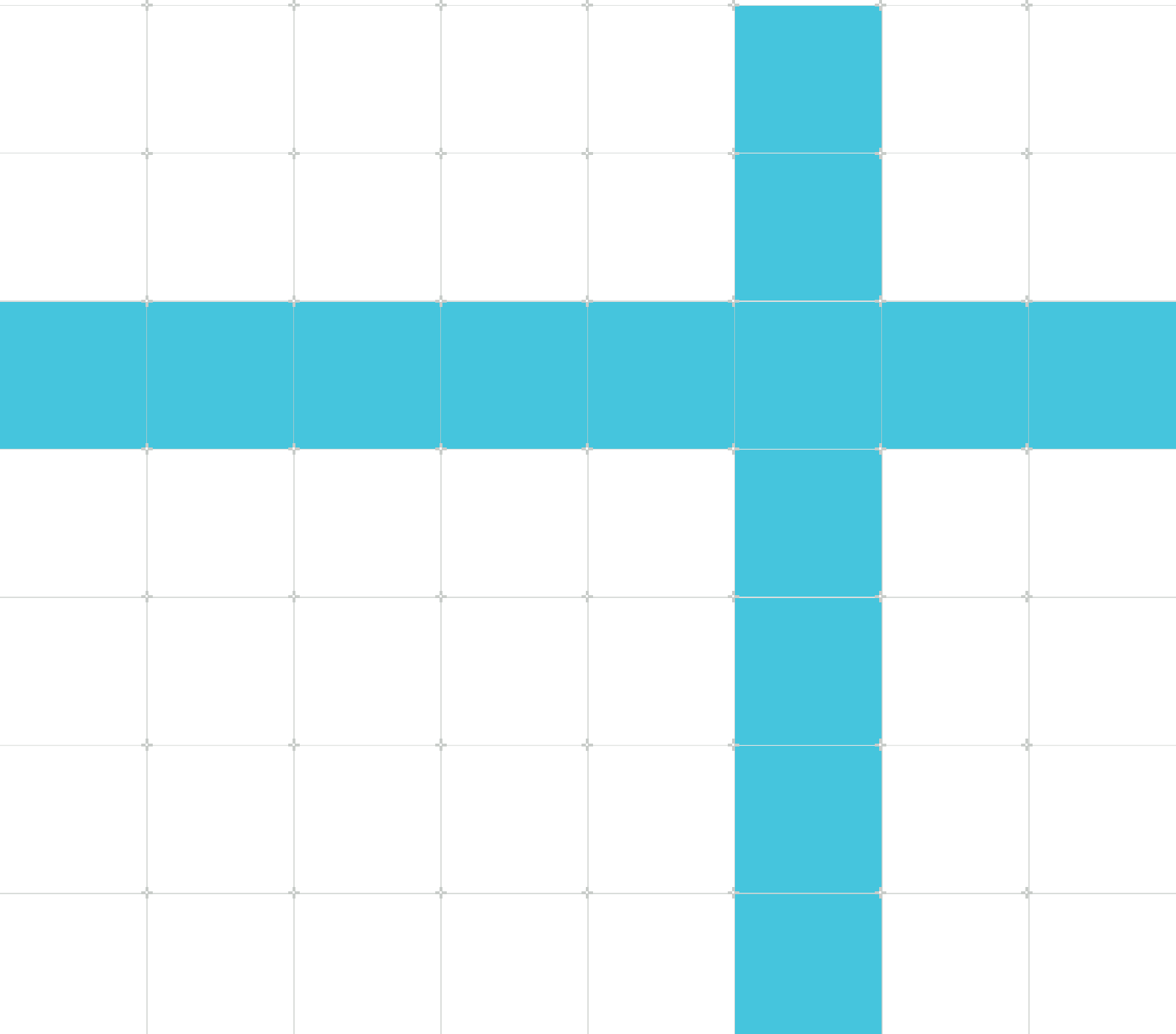
User Guide

Non-Confidential

Copyright © 2021–2022 Arm Limited (or its affiliates).
All rights reserved.

Issue 15

102497_1.5_15_en



Arm Keil Studio Cloud User Guide

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
15-01	25 May 2021	Non-Confidential	Initial release
15-02	10 June 2021	Non-Confidential	1.5.7 updates
15-03	22 July 2021	Non-Confidential	1.5.11 updates
15-04	11 August 2021	Non-Confidential	1.5.13 updates
15-05	19 October 2021	Non-Confidential	1.5.25 updates
15-06	7 December 2021	Non-Confidential	1.5.28 updates
15-07	26 January 2022	Non-Confidential	1.5.31 updates
15-08	27 January 2022	Non-Confidential	1.5.31 documentation release corrections
15-09	8 February 2022	Non-Confidential	Metadata updates
15-10	10 March 2022	Non-Confidential	1.5.36 updates
15-11	7 April 2022	Non-Confidential	1.5.39 updates
15-12	17 May 2022	Non-Confidential	1.5.41 updates
15-13	8 June 2022	Non-Confidential	1.5.42 updates
15-14	12 July 2022	Non-Confidential	1.5.47 updates
15-15	11 August 2022	Non-Confidential	1.5.48 updates

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied,

by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.

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.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction.....	10
1.1 Conventions.....	10
1.2 Other information.....	11
2. Arm Keil Studio.....	12
3. Prerequisites.....	13
3.1 System requirements.....	13
3.2 Access Keil Studio.....	13
4. Manage accounts.....	14
4.1 User Profile view.....	14
5. Tutorials.....	17
5.1 Get started with a CMSIS Blinky example.....	17
5.2 Get started with an Mbed OS Blinky example.....	20
5.3 Export and import a CMSIS project between Keil Studio and Keil MDK.....	21
5.3.1 Export a CMSIS project from Keil Studio and import it in Keil MDK.....	21
5.3.2 Export a CMSIS project from Keil MDK and import it in Keil Studio.....	23
5.4 Work with Git source control.....	25
5.4.1 Configure your project for source control.....	25
5.4.2 Use source control and publish your changes.....	27
5.5 Debug a Blinky example.....	31
5.6 Connect to AWS IoT and send MQTT messages.....	34
5.6.1 Manage your AWS IoT things and certificates.....	34
5.6.2 Configure connection settings in your project.....	36
5.6.3 Send MQTT messages.....	38
6. User interface.....	41
6.1 UI layout.....	41
6.2 Command palette.....	43
6.3 Tips and tricks.....	45
6.3.1 Getting Started page.....	46
6.3.2 Editor.....	46

6.4 Customize the UI.....	53
6.4.1 Preferences.....	53
6.4.2 Keyboard shortcuts.....	55
7. Work with standalone CMSIS projects and CMSIS solutions.....	58
7.1 Standalone CMSIS projects.....	58
7.2 CMSIS solutions.....	58
7.3 Create, import or clone a standalone CMSIS project.....	58
7.3.1 Create a project from a CMSIS example project.....	59
7.3.2 Import a CMSIS example project from the Keil Studio website.....	59
7.3.3 Clone a CMSIS project.....	60
7.3.4 Further resources about CMSIS.....	60
7.4 Manage a standalone CMSIS project and its software components.....	60
7.4.1 Manage Software Components view and Project outline.....	61
7.4.2 Open the Manage Software Components view.....	61
7.4.3 Modify the software components in your project.....	64
7.4.4 Manage your project files from the Project outline.....	64
7.5 Manage a CMSIS solution and its software components.....	66
7.5.1 Manage Software Components view.....	66
7.5.2 Open the Manage Software Components view.....	67
7.5.3 Modify the software components in your project.....	69
8. Work with Mbed projects.....	70
8.1 Create, import or clone an Mbed project.....	70
8.1.1 Create a project from an Mbed example project or an empty Mbed project.....	70
8.1.2 Create a blank Mbed project.....	71
8.1.3 Import an Mbed OS example from your file system.....	72
8.1.4 Import Mbed projects from your Mbed Online Compiler workspace.....	72
8.1.5 Clone an Mbed project.....	72
8.1.6 Next steps.....	73
8.1.7 Further resources about Mbed OS.....	73
8.2 Manage an Mbed project and its libraries.....	73
8.2.1 Mbed OS 5 or 6 project.....	74
8.2.2 Mbed 2 project.....	78
9. Build and run a project.....	81
9.1 Select a build target.....	81

9.2 Build and flash a project to the board.....	81
9.3 Output view.....	82
9.4 Configuring compile-time customizations.....	83
10. IntelliSense.....	84
10.1 Code editing.....	84
10.1.1 Navigate your code.....	84
10.1.2 Edit and refactor your code.....	92
10.2 Code linting.....	98
10.2.1 Enable clang-tidy.....	98
10.2.2 Configure clang-tidy checks.....	99
11. Manage files.....	100
11.1 Search the content of files.....	100
11.1.1 Search the content of multiple files.....	100
11.1.2 Search the content of a specific file.....	101
11.1.3 Include or exclude search patterns.....	101
11.2 Compare files.....	102
11.3 Upload and download files or projects.....	102
11.4 Change file locations.....	102
11.5 Copy the path of a file or folder.....	103
12. Source control.....	104
12.1 Work with Git.....	104
12.1.1 Set credentials for GitHub.....	104
12.1.2 Interface and features reference.....	105
12.1.3 Configure a project for source control and collaboration.....	107
12.1.4 Create or switch branches.....	109
12.1.5 Manage local files.....	110
12.1.6 Synchronize.....	114
12.2 Work with Mercurial.....	115
12.2.1 Credentials.....	116
12.2.2 Interface and features reference.....	116
12.2.3 Configure a project for source control and collaboration.....	118
12.2.4 Create or switch branches.....	119
12.2.5 Manage local files.....	120
12.2.6 Synchronize.....	121

12.3 History view.....	122
13. Monitor and debug.....	123
13.1 Use the Serial Monitor view.....	123
13.2 Debug a project with Keil Studio.....	123
13.2.1 Introduction.....	124
13.2.2 Debug first steps.....	124
13.2.3 Restart or stop the debugger.....	125
13.2.4 Navigate your code using the step buttons.....	125
13.2.5 Set breakpoints.....	125
13.2.6 Set function breakpoints.....	126
13.2.7 Examine threads and call stacks.....	126
13.2.8 Inspect variables.....	127
13.2.9 Inspect registers.....	128
13.2.10 Advanced debugger settings.....	129
13.2.11 Switch to RAM debugging.....	131
13.3 Use the Memory view.....	132
14. Supported and custom targets.....	136
14.1 Supported development boards.....	136
14.2 Supported debug probes.....	136
14.3 Custom targets.....	137
14.3.1 Set up a custom target.....	137
14.3.2 Reset a target.....	138
15. Extensions.....	139
15.1 Install the AWS Toolkit extension.....	139
15.2 Connect to AWS from Keil Studio.....	139
16. Known issues and troubleshooting.....	141
16.1 Known issues.....	141
16.2 Troubleshooting.....	141
16.2.1 Keil Studio does not load.....	141
16.2.2 Cannot log into Keil Studio.....	141
16.2.3 Connected development board or debug probe not found.....	142
16.2.4 Out-of-date firmware.....	142
16.2.5 Connection fails when clicking Run project or Debug project.....	142

16.2.6 Development board not showing serial data.....	144
16.2.7 Cannot make an Mbed project the Active Project.....	145
16.2.8 Code editor shows many red squiggly lines in an Mbed project.....	145
16.2.9 Linker failing with file not found for Mbed OS 15.4.0 and older.....	146
16.3 Report an issue or suggest an enhancement.....	146

1. Introduction

1.1 Conventions

The following subsections describe conventions used in Arm documents.





Glossary



The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm® Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .
 Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
 Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
 Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
 Note	An important piece of information that needs your attention.

Convention	Use
 Tip	A useful tip that might make it easier, better or faster to perform a task.
 Remember	A reminder of something important that relates to the information you are reading.

1.2 Other information

See the Arm website for other relevant information.

- [Arm® Developer](#).
- [Arm® Documentation](#).
- [Technical Support](#).
- [Arm® Glossary](#).

2. Arm Keil Studio

Arm Keil Studio is a free to use, browser-based IDE for the evaluation and development of embedded, IoT, and Machine Learning software for Cortex-M devices. With a cloud-hosted workspace for your code, comprehensive source control integration, and a powerful C/C++ editor, you can edit your projects from any computer, share them with colleagues and export them for desktop usage in Keil μ Vision. You can compile projects using Arm Compiler 6, flash the projects directly to supported development boards, and debug from supported browsers without the need to install any software.

Our goal is to make it quicker and easier for you to evaluate reference designs, reducing the time it takes to get your embedded projects to market, while also providing a point of integration for Arm ecosystem partners who provide professional software, tools, and services.

Arm Keil Studio demonstrates next generation IDE technology and new concepts for CMSIS project formats . We support a range of software examples, showcasing Keil RTX, FreeRTOS, and IoT connectors for Amazon AWS IoT, Microsoft Azure IoT Hub, and Google Cloud. Keil Studio is the successor to the Mbed Online Compiler, and allows you to develop Mbed OS 5 and 6 projects on supported Mbed Enabled boards. Keil Studio also provides limited support for Mbed 2. To get started, you can import Mbed projects from your Online Compiler workspace or mbed.com.

You can access Keil Studio today using an Arm or Mbed account and get started by opening a reference design to evaluate.

To find out which development boards and debug probes are supported, check the [Supported development boards](#) and [Supported debug probes](#) pages.



Keil Studio is a new software tool under constant development, and we regularly publish bug fixes and feature updates. To use flash and debug capabilities, you must use Google Chrome or Microsoft Edge (Chromium).

3. Prerequisites

This chapter presents the system and account requirements you need to access Keil Studio.

3.1 System requirements

To work with development boards over USB, you must use Keil Studio in a desktop browser that supports the WebUSB standard: Google Chrome or Microsoft Edge (Chromium). All other features are supported in the latest versions of the following desktop browsers: Google Chrome, Microsoft Edge, Opera, Safari, and Mozilla Firefox.

Keil Studio supports Mbed OS 5.12 and newer, and Mbed OS 6 and newer. Keil Studio also provides limited support for Mbed 2. See [Manage an Mbed project and its libraries](#) for more details.



If you are moving a project from Mbed OS 5 to Mbed OS 6, note [the deprecated APIs](#).

Installation of udev rules on Linux

On Linux, you must install udev rules to be able to build a project and flash it to your device or debug a project with Keil Studio.

The installation of udev rules requires sudo privileges. See [udev rules for Linux](#) for more information.

3.2 Access Keil Studio

You need an Arm account to work with Keil Studio. If you already have an Mbed account, you can use it to access Keil Studio.

You can create an Arm account from the Keil Studio page at: studio.keil.arm.com.

You can learn how to manage the accounts you use with Keil Studio, see [Manage accounts](#).

4. Manage accounts

This chapter describes how to manage the accounts you use with Keil Studio from the User Profile view.

By default, when you log into Keil Studio, your Keil Studio account gets listed in the User Profile view.

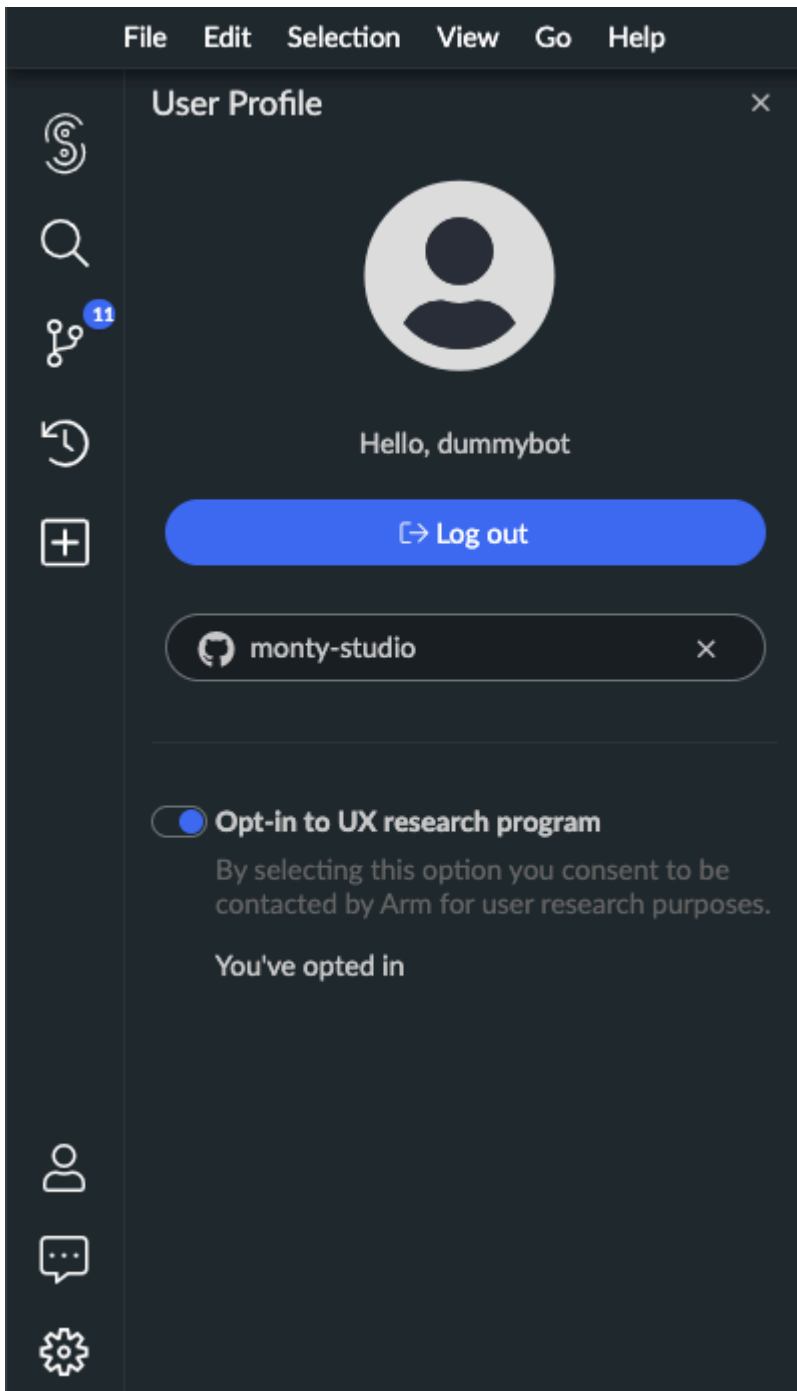
If you have a personal user, organization, or enterprise GitHub account, you can link that account with Keil Studio. Linking your GitHub account with Keil Studio allows you to access and create GitHub repositories from Keil Studio. To learn how to add your GitHub account in Keil Studio, see [Set credentials for GitHub](#).

4.1 User Profile view


The User Profile view lists the accounts which you have associated with Keil Studio.

The accounts that you can view in the User Profile view are:

- Keil Studio account: Allows you to log out of Keil Studio (by default, when you initially log into Keil Studio, your Keil Studio account is listed in the User Profile view).
- GitHub account: Connect or disconnect your GitHub account from Keil Studio.

Figure 4-1: User Profile view

Access the User Profile view

To access the User Profile view, click the Accounts icon  in the Keil Studio activity bar and select Show User Profile.

Features

Feature	Description
Keil Studio Log out button	Logs you out of Keil Studio.
GitHub Connect to GitHub and Disconnect GitHub account buttons	Connect or disconnect your GitHub account with Keil Studio. For more information, see Set credentials for GitHub .
UX research program switch	Opt-in (consent to being contacted by Arm) or opt out (do not consent to being contacted by Arm) of the Keil Studio UX research program.



Disconnecting your GitHub account using the Disconnect GitHub account button does not remove the Keil Studio application from your list of authenticated GitHub applications. You can manage your authenticated GitHub applications from your [account settings in GitHub](#).

5. Tutorials

In this chapter, you will find tutorials to help you learn how to use Keil Studio.

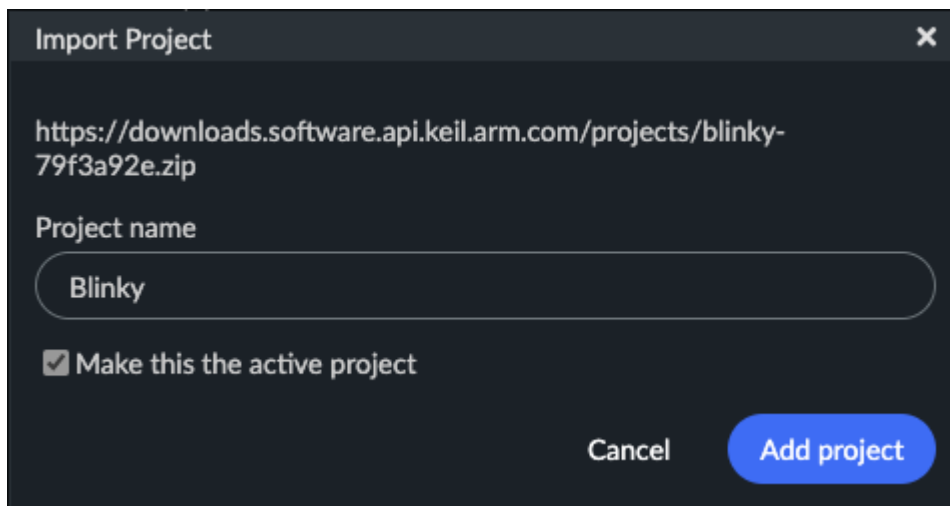
5.1 Get started with a CMSIS Blinky example

This tutorial explains how to start a project in Keil Studio using the CMSIS Blinky example available for the NXP FRDM-K32L3A6 board. Blinky is a simple application that blinks the LED on your development board. Learn how to import a project, and then build and flash Blinky to your board.

Procedure

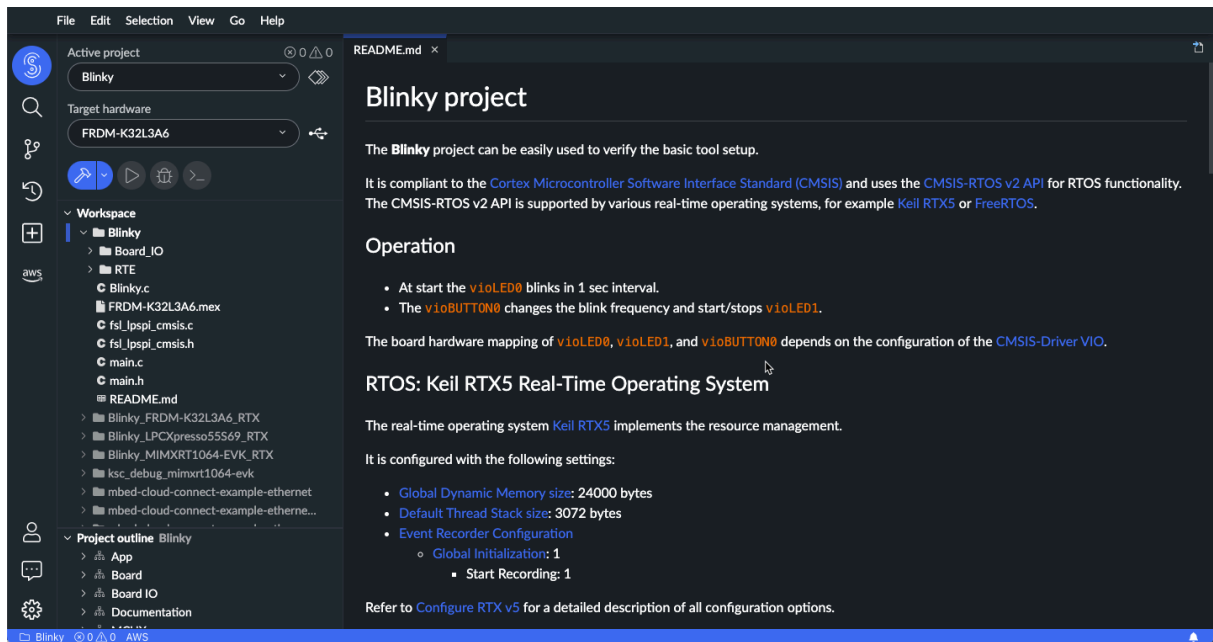
1. Go to [keil.arm.com](https://www.keil.arm.com).
2. Click the Hardware menu to see the list of supported hardware.
You can search for a board by name or vendor, or filter the list by vendor or core. You can also display boards with example projects with the Only include boards with example projects checkbox.
3. Search for and select the FRDM-K32L3A6 board in the list.
You can either download or import example projects into Keil Studio from the Projects tab and get access to board details from the Features and Documentation tabs.
4. Find the Blinky example in the Projects tab and click the Open in Keil Studio button for the example project.
5. Log into Keil Studio with your Arm or Mbed account if you are not already logged in.
6. Keil Studio opens. Confirm the project name in the Import Project dialog box. Keil Studio sets the newly imported project as the active project by default.

Figure 5-1: Import project



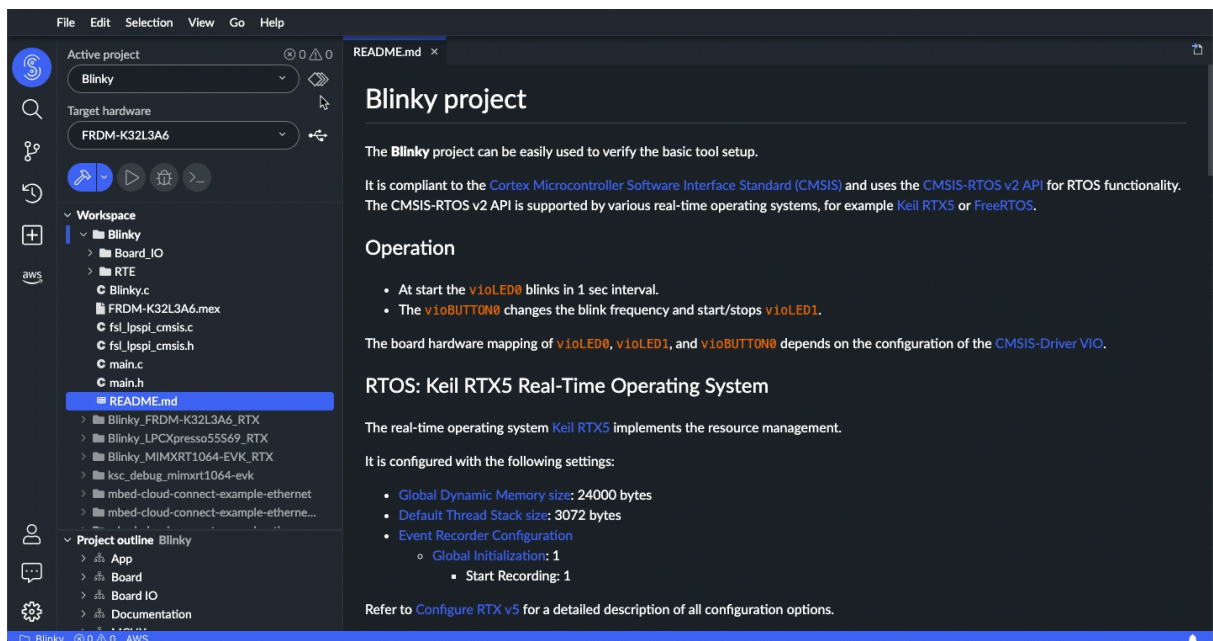
7. Click Add project.
The project loads to your workspace and is the active project. The `README.md` file of the project displays. Review the file to learn more about project settings and board requirements.

Figure 5-2: Open project showing README.md



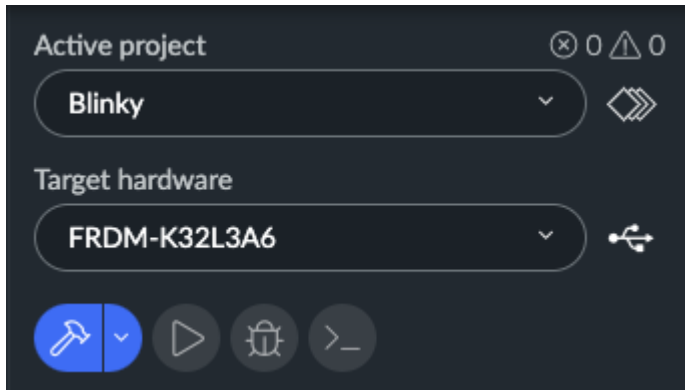
- Inspect the software components that are used in the project and access their documentation:

Figure 5-3: Manage software components



- The Target hardware is automatically set to the build target of the project:

Figure 5-4: Target hardware

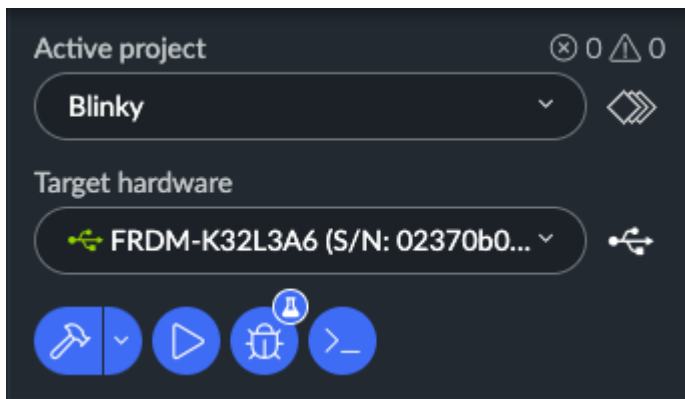


A build target tells Keil Studio how to build the project so that it matches your hardware. To


build the project, click the Build project button, . Build project builds Blinky and stops.

10. Connect your board to your computer. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target:

Figure 5-5: Target connected



The green USB icon to the left of the target hardware name shows that the target is connected.

The serial number also displays in the tooltip. Now, the Run project  button is enabled. Click the Run project button to build Blinky and flash it to your board.





5.2 Get started with an Mbed OS Blinky example

This tutorial explains how to start a project in Keil Studio using an Mbed OS Blinky example. Blinky is a simple application that blinks the LED on your development board. Learn how to create a new project, and then build and flash Blinky to your board.

Before you begin

Go to studio.keil.arm.com and log into Keil Studio using your Arm or Mbed account.

Procedure

1. Create a new project, select File > New... > Mbed Project.
The New Project dialog box opens.
2. Click the Example project drop-down list and select `mbed-os-example-blinky` under Mbed OS 6.
3. Confirm the project name.
4. Check the default options:
 - Keil Studio sets the newly created project as the active project. Build and run commands only apply to the active project. Clear the checkbox if you do not want to make the new project active.
 - Keil Studio initializes the project as a Git repository. Clear the checkbox if you do not want to turn your project into a Git repository. See [Configure a project for source control and collaboration](#) for more details.
5. Click Add project.
The project loads to your workspace and is the active project. The `README.md` file of the project displays. Review the file to learn more about project settings and board requirements.
6. The Target hardware drop-down list shows the build target set for the project. A build target tells Keil Studio how to build Mbed OS so that it matches your hardware. To select a build target, you can either:
 - Use the Target hardware drop-down list. Your target hardware name most likely matches the board name.
 - Connect your board to your computer. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target.
7.  To build the project, click the Build project button, . Build project builds Blinky and stops.
8.  If you have a board connected, click Run project . Run project builds Blinky and flashes it to your board.
You might have to restart your board for Blinky to run.

For Mbed projects only: Select the Use Daplink option in File > Settings > Open Preferences > Run category to build and flash projects to your board. When this option is not selected, the Run project button is not available.

5.3 Export and import a CMSIS project between Keil Studio and Keil MDK

While Keil Studio cannot read the Keil MDK `.uvprojx` format, you can switch from Keil Studio to Keil MDK using the CMSIS `.cprj` project file format.

This tutorial explains how to export a CMSIS project in `.cprj` format from Keil Studio and then import the project in Keil MDK. It also explains how to export a project from Keil MDK and import it in Keil Studio.

Before you begin

- In Keil Studio, create a CMSIS project from the examples provided and set it as the active project (right-click the project and select Set Active Project).
- If you do not have Keil MDK already installed, you can download and install the MDK-Community edition for free from [keil.arm.com](https://www.keil.arm.com). You need an Arm account (or Mbed account) to access the page and get a license key.



Keil MDK is supported on Windows 64-bit variants only.

5.3.1 Export a CMSIS project from Keil Studio and import it in Keil MDK

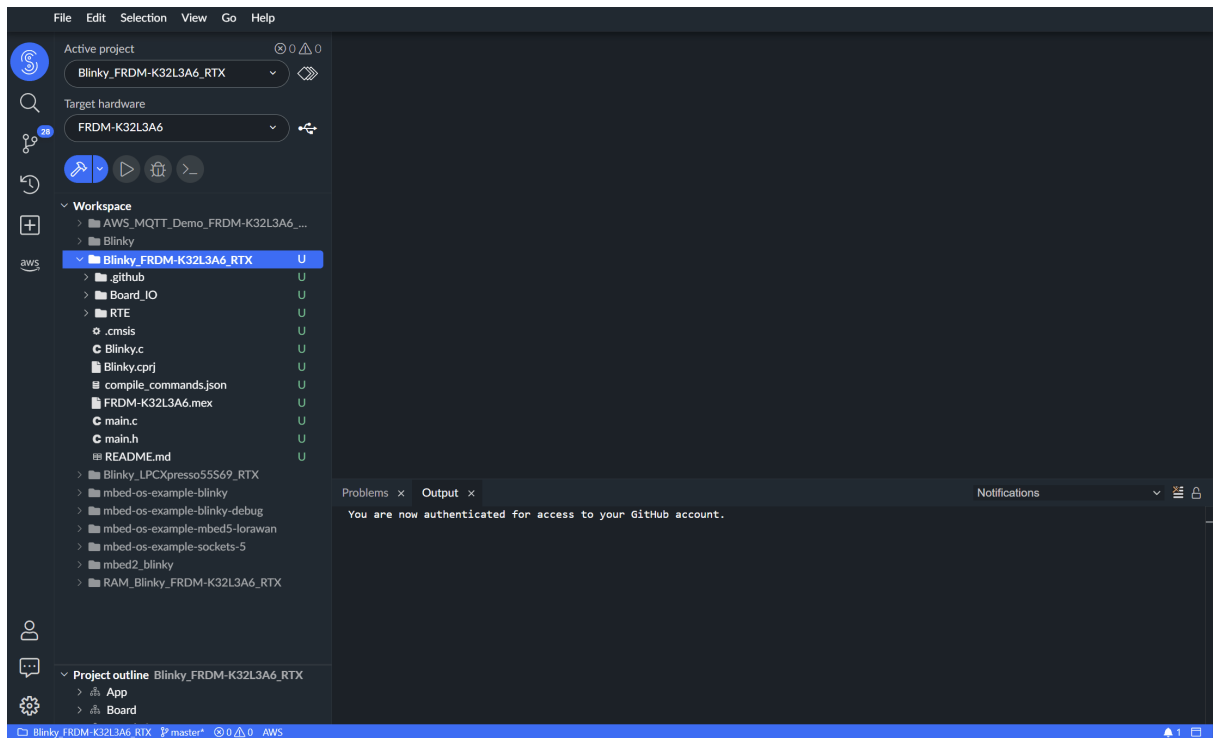
Keil Studio uses the `.cprj` project file format natively. You can use the download options available in Keil Studio to download your project as a `.tar` file and then import the project in Keil MDK.

Procedure

1. In Keil Studio: Go to the File menu and select the Download Current Selection or the Download Active Project option.
Your project is exported as `<Project_Name>.tar` and contains a `<Project_Name>.cprj` project file.

Note: The Download Current Selection option is also available from the Explorer view in the right-click menu.

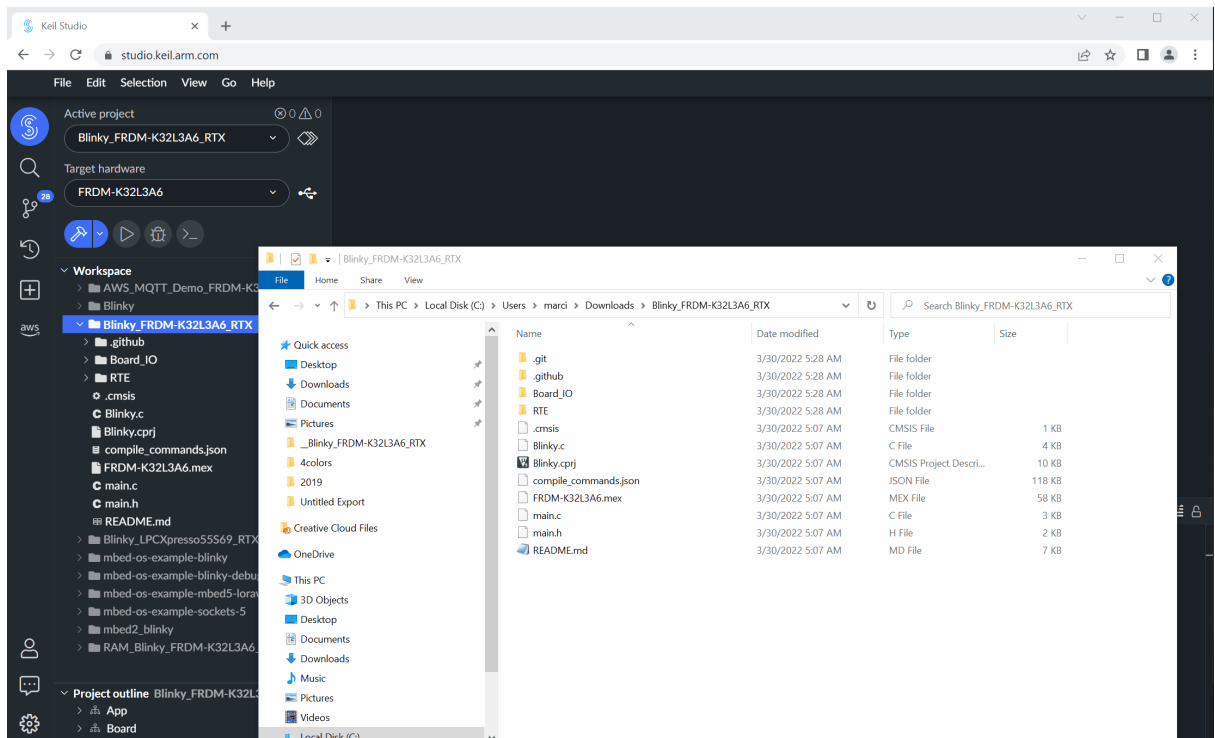
2. Open the `<Project_Name>.tar` file with an appropriate application on your PC.

Figure 5-6: Download your project as a .tar file

3. Double-click the `<Project_Name>.cprj` project file to import the project in Keil MDK. The Keil μ Vision IDE opens.

Note: If Keil μ Vision is already open, you can go to Project > Import > Import Project from CPRJ Format to import the project.

4. In Keil μ Vision: If there are missing packs, you are prompted to install them. Click Yes to install the packs with the Pack Installer.
Keil MDK creates native MDK files (`<Project_Name>.uvprojx` and `<Project_Name>.uvoptx`) in the project folder and automatically copies specific files from software components in the RTE folder. It also creates folders for the output (Objects) and debug configuration files (DebugConfig).

Figure 5-7: Import the project in Keil MDK

Check the [Getting started with MDK](#) to start working with the project imported.

5.3.2 Export a CMSIS project from Keil MDK and import it in Keil Studio

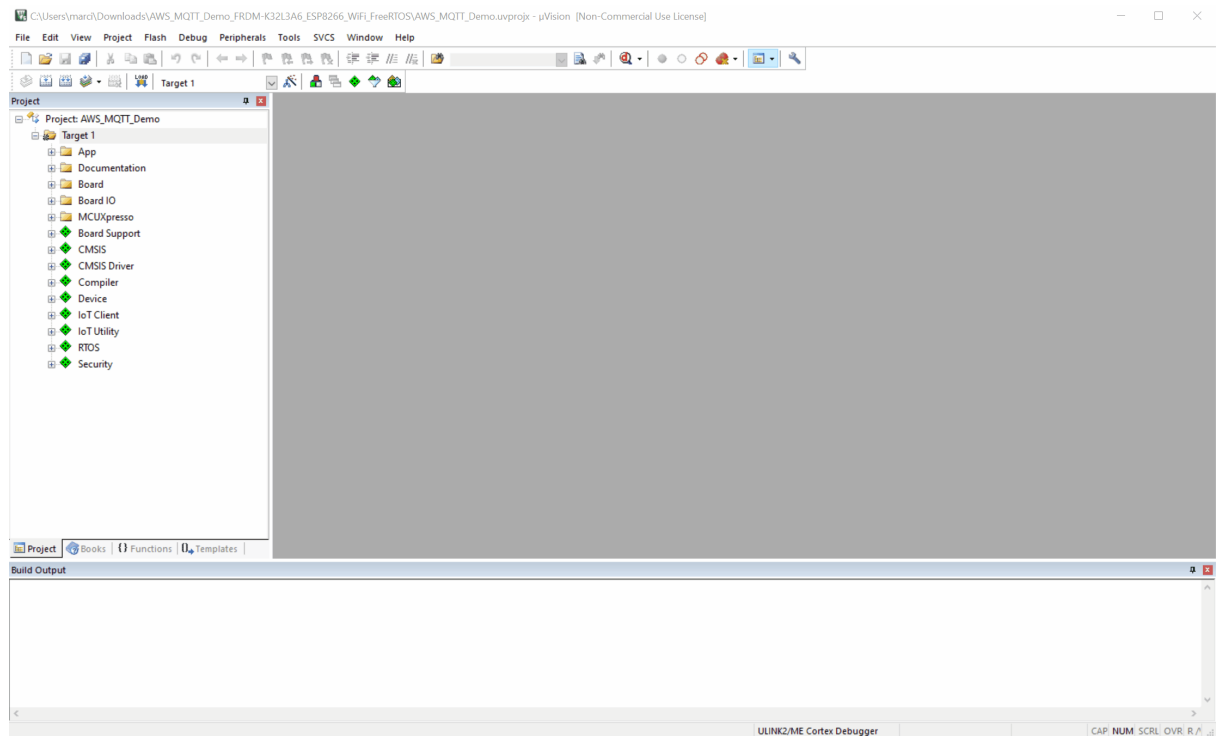
You can use the export option available in Keil MDK to export a project as a `.cprj` file and then import the project in Keil Studio.

Procedure

1. In Keil μ Vision: Open a project and go to Project > Export > Save Project to CPRJ Format to export the project.

A `<Project_Name>.<Target_Name>.cprj` file is automatically saved in the root folder of the project.

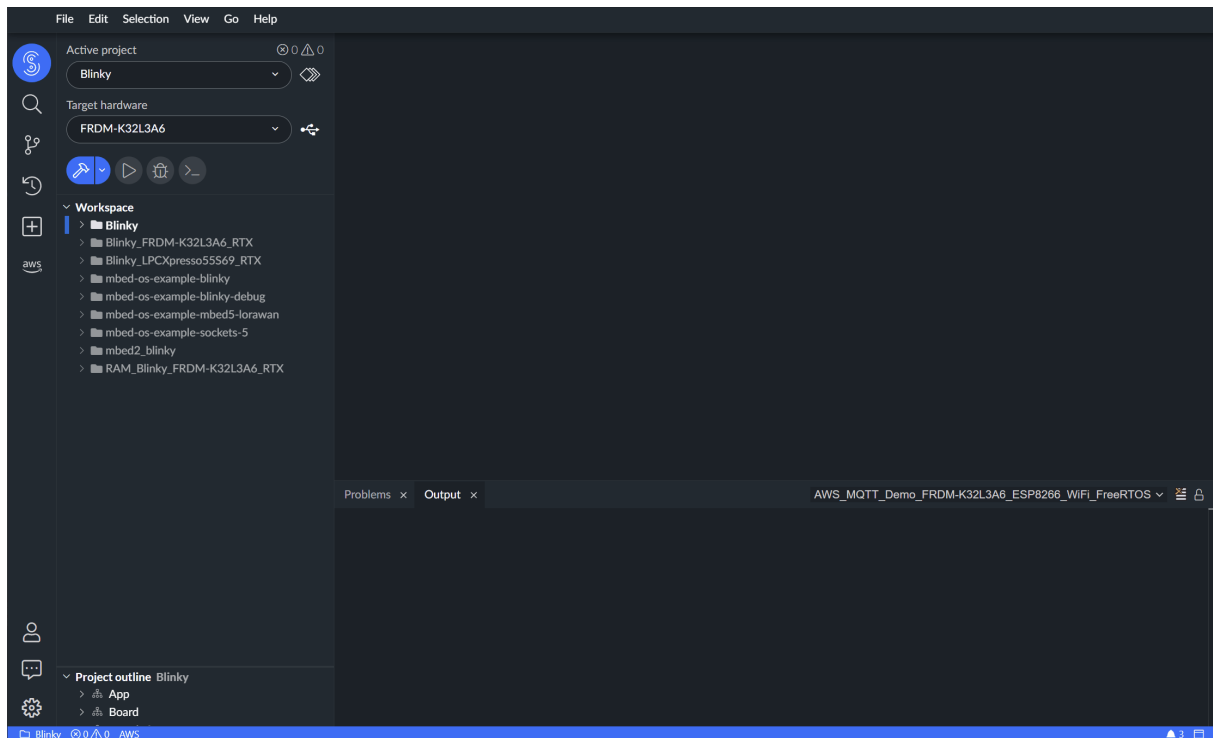
Figure 5-8: Export the project in .cprj format



2. In Keil Studio: Drag and drop the project folder from your file system to the Explorer view in Keil Studio.
3. Once the project is imported, right-click the project and select Set Active Project.
- 4.



Build the project by clicking the Build project button .

Figure 5-9: Import and build the project in Keil Studio

5.4 Work with Git source control

This tutorial explains the basics of Git source control in Keil Studio using the CMSIS Blinky example available for the NXP FRDM-K32L3A6 board. Note that the details provided are relevant for Mbed projects too. Start by configuring your project for source control to be able to publish and share your changes. Then, learn how to:

- Create a working branch.
- Review code changes in the Source Control view.
- Stage, commit, and push your changes.
- Merge your changes into the main branch.

See the sections below for more details.

5.4.1 Configure your project for source control

There are various ways of working with source control in Keil Studio: you can fork an existing Git project and publish your changes to your GitHub account, you can also start from a local project

and use an existing (empty) remote repository to publish changes, or you can publish your changes to a new repository from Keil Studio.

Before you begin

In this tutorial, you will learn how to publish your changes to a new GitHub repository.

- You need a GitHub account for this tutorial. Create an account on github.com or, if you already have an account, sign in. Once this is done, check that your Arm account and your GitHub account are connected as described in [Set credentials for GitHub](#).
- Check that you have created a `Blinky_FRDM-K32L3A6_RTX` project from the examples provided and set it as the active project (right-click the project and select Set Active Project).

Procedure

1. Go to the Source Control view.

- If you did not previously initialize the project as a Git repository when creating the project, an “Active project is not under version control” message displays. Click the Publish Project button or click the ... at the top of the Source Control view and select Publish Project.
- If the project is already initialized for Git, click the ... at the top of the Source Control view and select Publish Project.

The Publish dialog box opens.

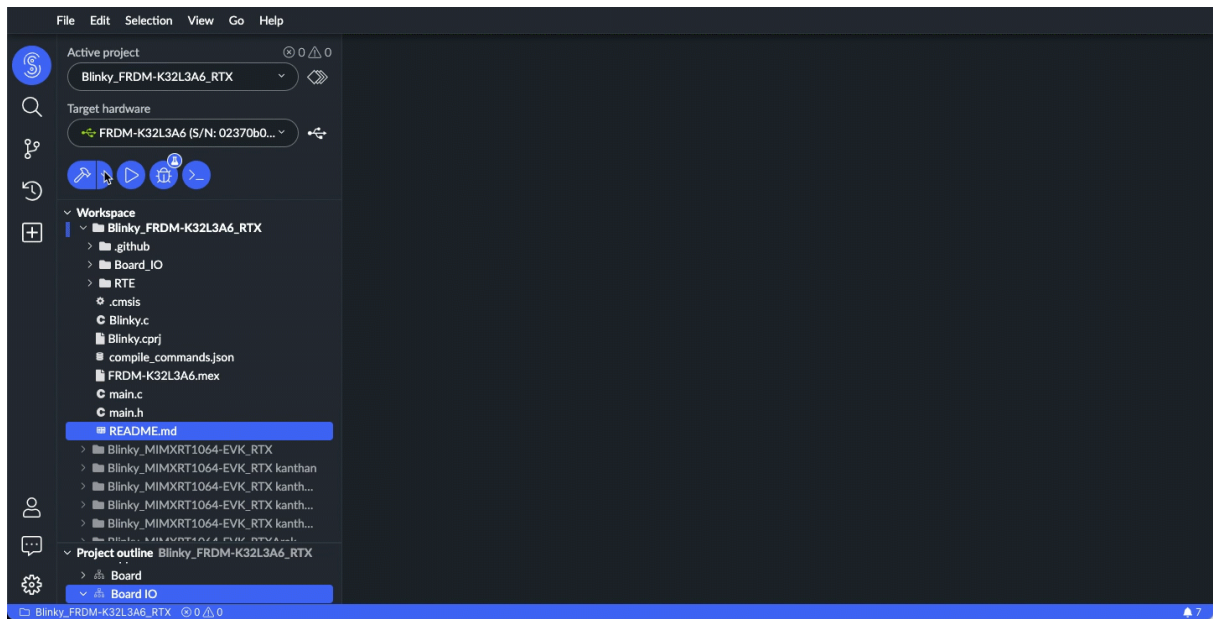
2. Select the Publish to a new GitHub repository option and enter a repository name and description.

3. By default Keil Studio creates a private repository. If you want to create a public repository, clear the Private repository checkbox.

4. Click Publish.

The Source Control view shows untracked changes (with a U status) in the Changes list. These are changes that have not been staged yet. When first setting a repository, you are on the master branch by default.

5. Go to your GitHub account and check that the repository has been created as expected. The repository does not contain any commits yet.

Figure 5-10: Configure your project for source control

5.4.2 Use source control and publish your changes

You are all set! You can now update your project and publish the changes.

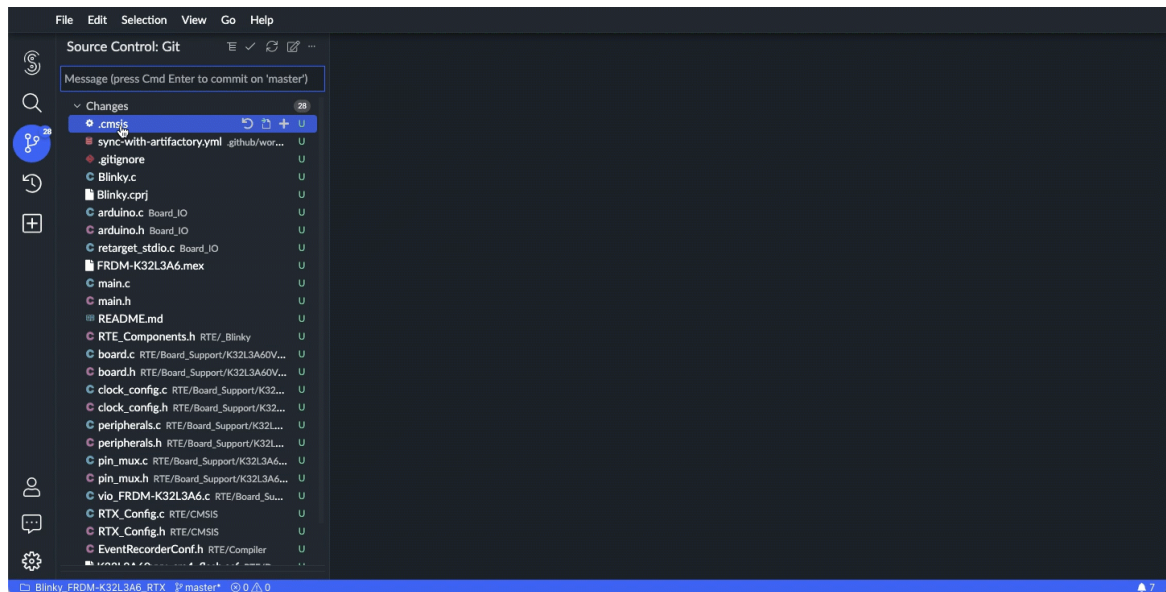
Before you begin

When working with other people on the same project, it is good practice to create a working branch for the feature you are currently working on, make changes on that working branch, then merge your work into the main branch (master).

Procedure

1. In the Source Control view, create an initial commit on the master branch:
 - a. Select all the unstaged files, hover over one of the files and click + to stage all the files.
The files are listed under Staged changes.
 - b. In the Message box, enter a commit message and click Commit to create the initial commit.

The committed changes are visible at the bottom of the Source Control view and in the History view, and your local master branch is updated.

Figure 5-11: Create a first commit

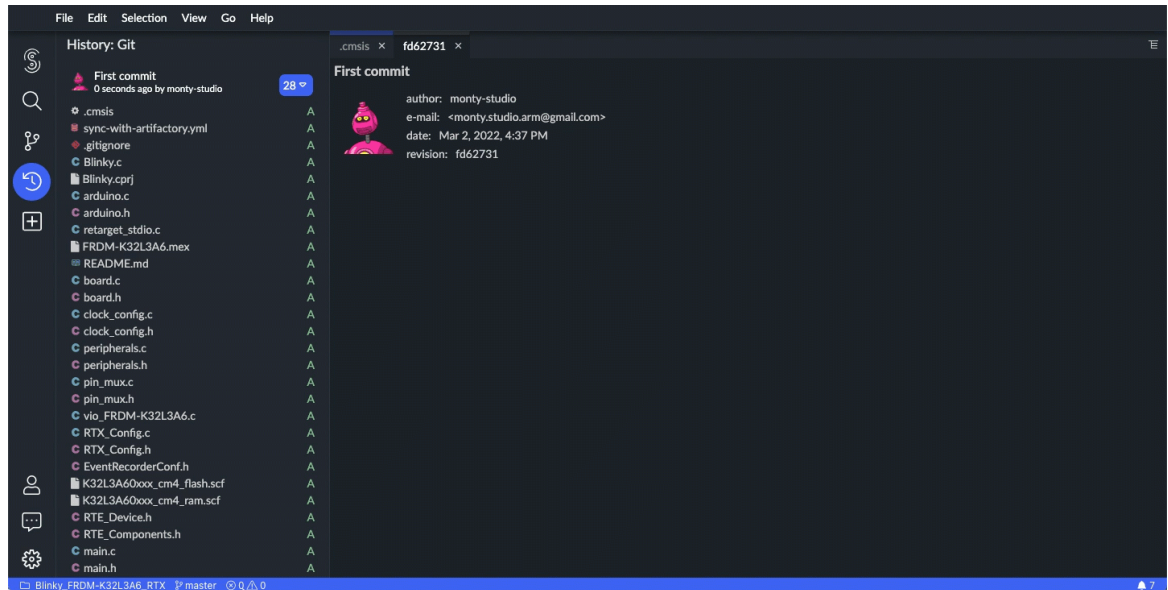
2. Create a working branch:

- a. Click the master branch in the status bar.

An input field pops up at the top of the window.

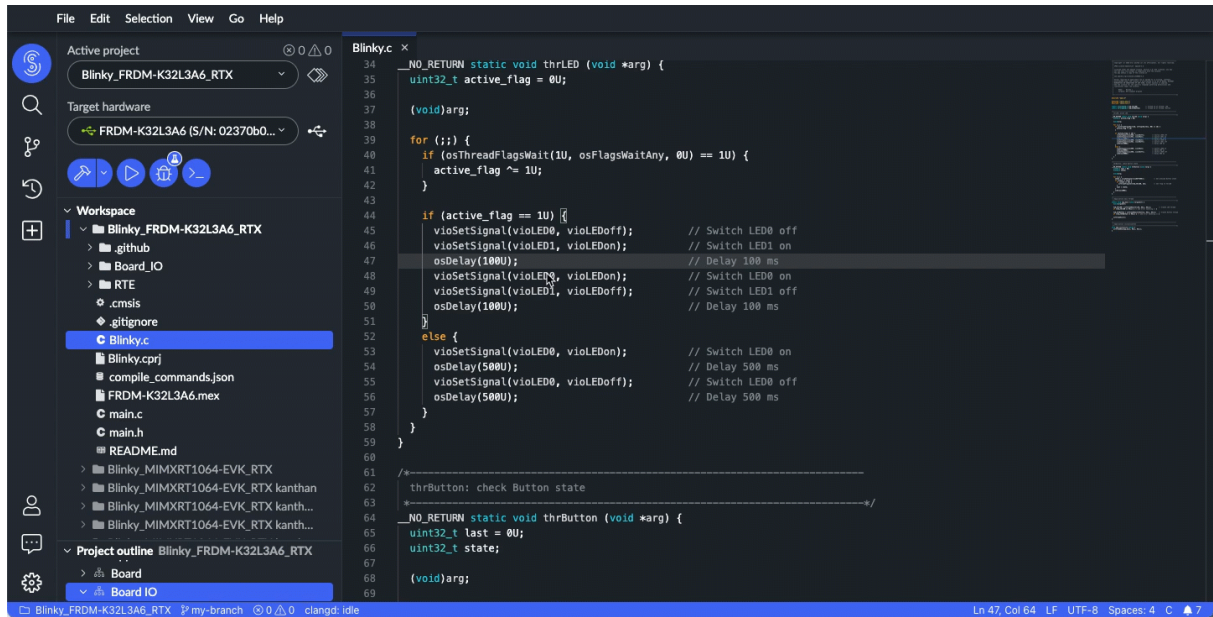
- b. Click Create new branch... and type the name of your branch in the field. For example my-branch.
- c. Press Enter.

The branch switches automatically to my-branch.

Figure 5-12: Create a working branch

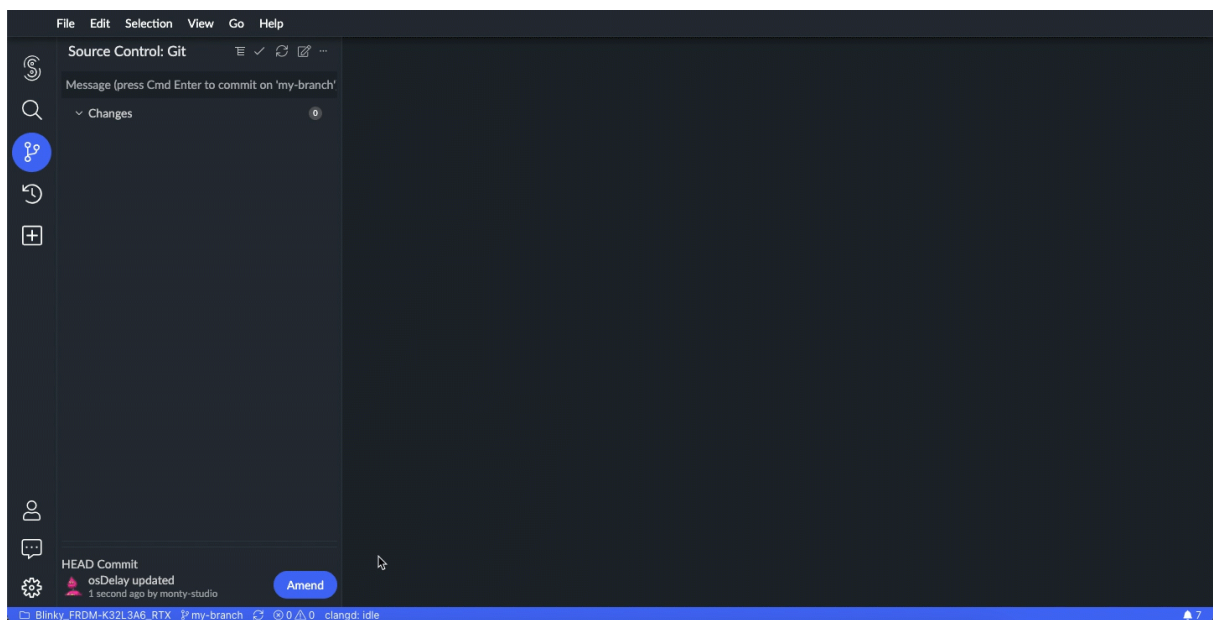
3. Go to the Explorer view, open the `Blinky_FRDM-K32L3A6_RTX` project folder, and look for the `Blinky.c` file.
4. Open that file and in the `thrLED: blink LED` block of code, change the `osDelay` values in the second `if` statement.
5. Now go to the Source Control view and click the `Blinky.c` file to check your changes. Note that you can also make changes from the side-by-side comparison window.
6. Hover over the `Blinky.c` file and click `+` to stage your changes. The file is listed under Staged changes.
7. In the Message box, enter a commit message and click Commit to commit the changes. The committed changes are visible at the bottom of the Source Control view and in the History view, and `my-branch` is updated.
8. To push your local commit to the remote repository, click `... > Push` from the Source Control view.

Figure 5-13: Stage, commit, and push your changes



9. Finally, to merge your changes into the main branch (master):
 - a. Switch to the master branch by clicking `my-branch` in the status bar and selecting `master` in the popup drop-down list.
 - b. Click `... > Merge...` and select `my-branch` in the popup drop-down list to merge the changes done on `my-branch` into `master`.
10. Click `... > Push` to update the remote repository.

Figure 5-14: Merge your changes into master



5.5 Debug a Blinky example

This tutorial explains how to debug a project using the CMSIS Blinky example available for the NXP FRDM-K32L3A6 board. Note that the details provided are relevant for Mbed projects too. Learn how to start a debug session, set breakpoints, step through your code, inspect variables, and examine threads and call stacks.

Before you begin

- Import the [Blinky example for the FRDM-K32L3A6 board from keil.arm.com](#) (Projects tab > Open in Keil Studio button) and set it as the active project (right-click the project and select Set Active Project).
- Connect your board to your computer. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target.
- Check that the correct build target is set in the Target hardware drop-down list.

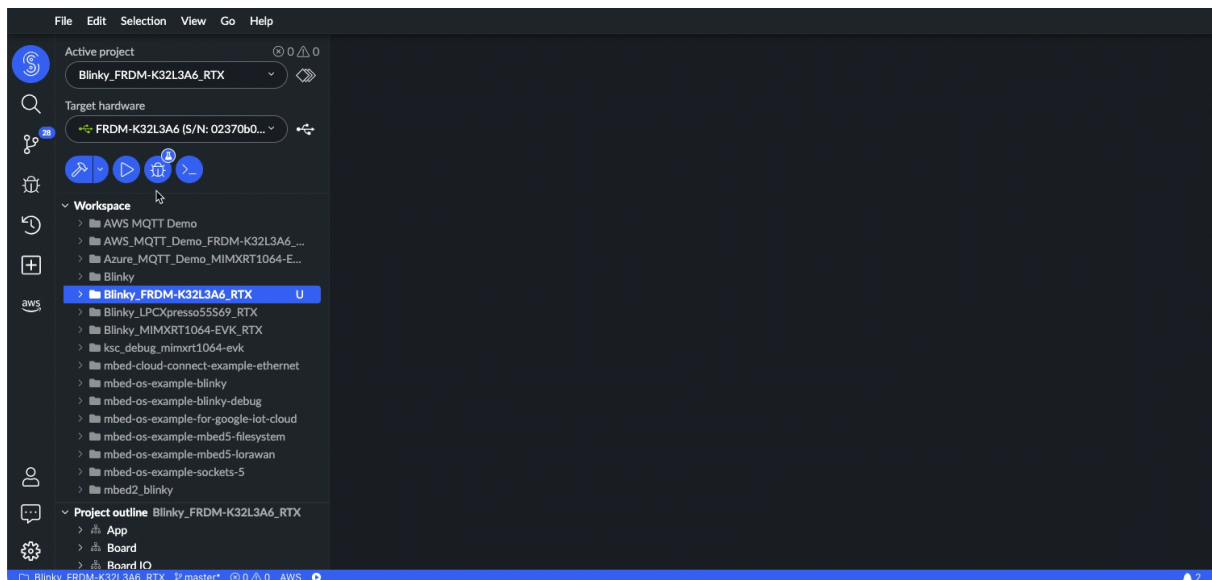
Procedure

1.

Click the Debug project  button to start a debug session.

Keil Studio automatically builds and flashes the project to your board. The debug session starts: Keil Studio switches to the Debug view and the status bar turns orange. The Debug Console view displays and shows debugging output. The execution stops at `main()`.

Figure 5-15: Start a debug session



2. While the debug session is still running: go to the Explorer view, open the `Blinky_FRDM-K32L3A6_RTX` project folder, and look for the `Blinky.c` file.
3. Open that file and in the `throwLED: blink LED` block of code, set a breakpoint on the `if (active_flag == 1U) {` statement by clicking the left margin. A red dot displays where you set the breakpoint.


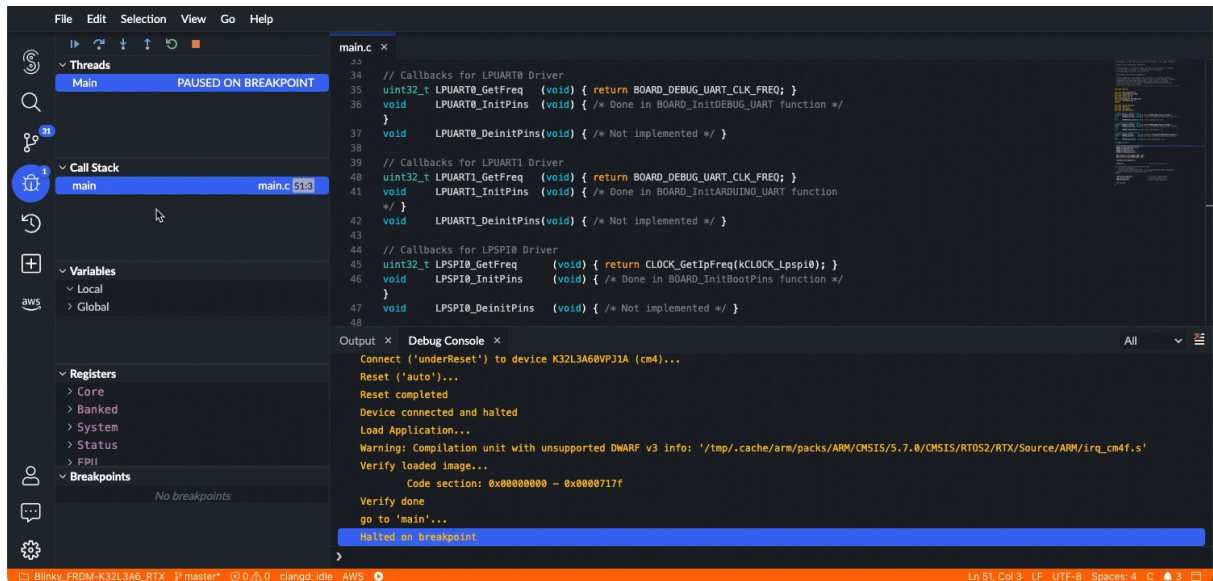
4. Return to the Debug view. You can see that your breakpoint has been added to the Breakpoints list on the left of the Debug view.
5.  Click the Continue button to start debugging. The debugger runs to the breakpoint you set and stops. A yellow arrow displays next to the statement on which the debugger paused. The statement highlights in yellow.

Figure 5-16: Set a breakpoint




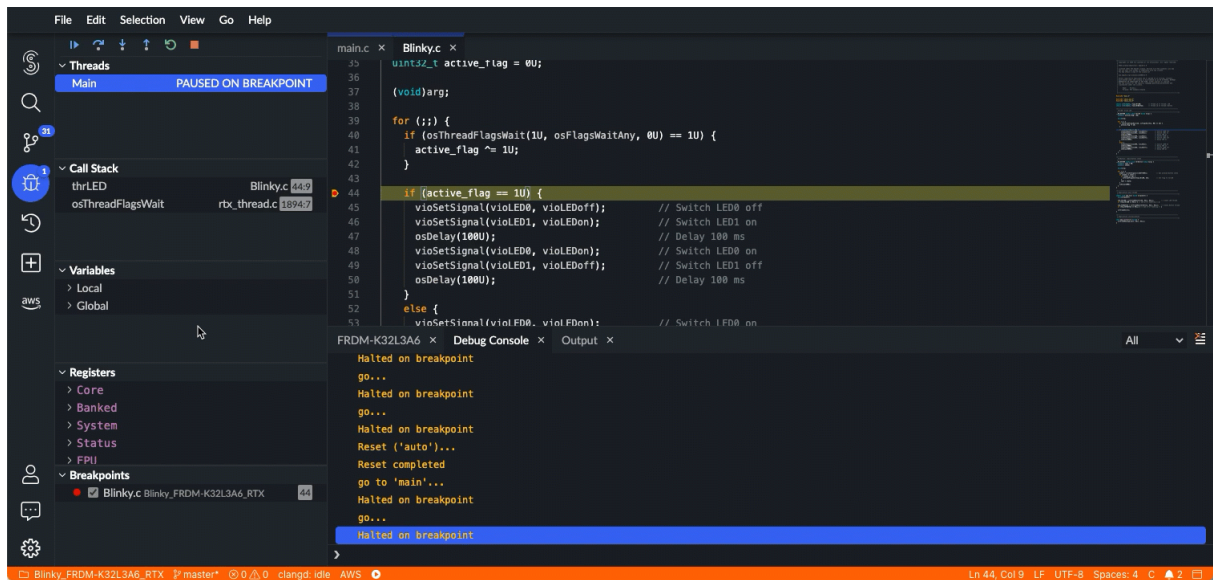
6. Go to the Variables list and check the local variables which are grouped under Local. The `active_flag` variable for `thrLED` is set at 0.
7. Locate, but do not press, the SW2 button on your board. See the [FRDM-K32L3A6 guide](#) to find the SW2 button. Prepare to press the SW2 button then:
 - a.  Click the Continue button.
 - b. Press the SW2 button on your board before the debugger hits the breakpoint set earlier.
8. Check the `active_flag` variable again. The `active_flag` variable is now set at 1.

Figure 5-17: Check variables







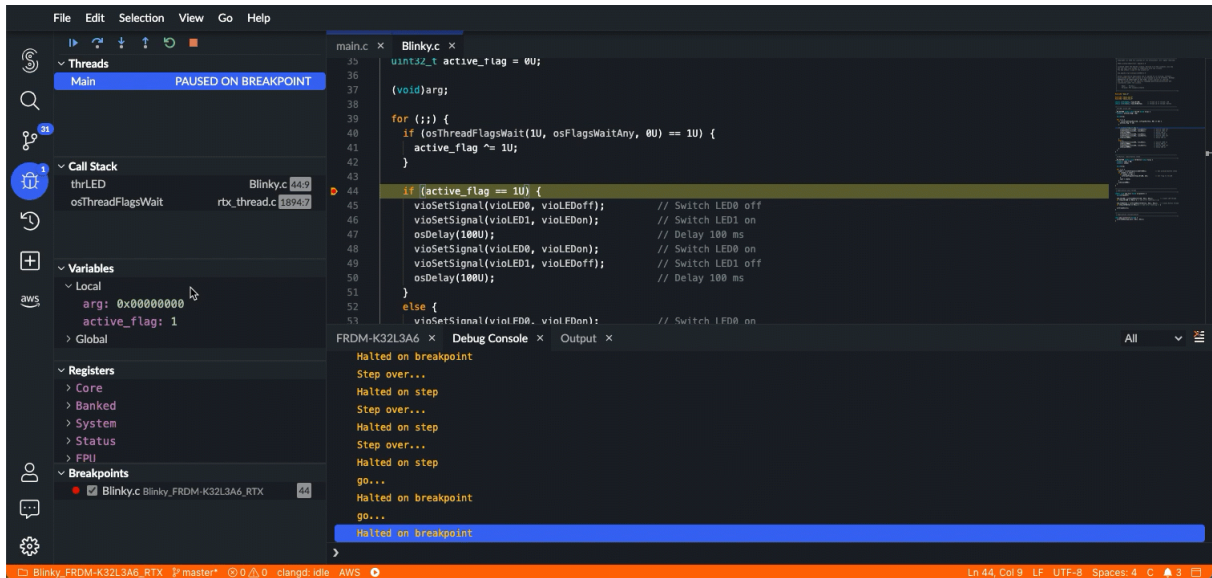
9. Click the Step over button  three times to go straight to `osDelay(100U);`.
10. Click the Step into button  to check this function. This opens the `rtx_delay.c` file where the function is called.
11. You can now examine what happens in the Threads and Call Stack lists. In the current version of Keil Studio, there is only one thread (Main). The call stack shows the execution flow of your code. In this example, you can see that `osDelay;` is called by `thrLED` in the `Blinky.c` file and `osstatus_t` in the `rtx_delay.c` file.
12. Click the Step into button  again several times to check the execution flow.
13. Click the Stop button  to stop the debugger and return to the editor.

Figure 5-18: Examine the call stack

5.6 Connect to AWS IoT and send MQTT messages

This tutorial explains how to connect your device to AWS IoT using the CMSIS AWS MQTT demo available for the STMicroelectronics B-L475E-IOT01A board. Learn how to manage your AWS IoT things, generate certificates, and configure connection settings in your project. Once everything is set up, you will be able to send MQTT messages to AWS IoT.

5.6.1 Manage your AWS IoT things and certificates

In this section, you will learn how to create a thing and a certificate (and associated RSA key pair). You will also create a policy and attach it to the certificate. Finally, you will activate the certificate and attach it to the thing.


Before you begin

- Install the AWS Toolkit extension and connect to AWS with your credentials. Follow the steps in [Install the AWS Toolkit extension](#) and [Connect to AWS from Keil Studio](#). Once connected to AWS, you should have access to the AWS Explorer from Keil Studio.
- You need a board that can connect to the internet. For this tutorial, we assume you are using the B-L475E-IOT01A board.
- To see the messages sent from your board, you will need an external terminal as the Serial Monitor view available in Keil Studio cannot be used to get serial output from STMicroelectronics boards.
- Import the [AWS MQTT Demo example for the B-L475E-IOT01A board from keil.arm.com](#) (Projects tab > Open in Keil Studio button) and set it as the active project (right-click the project and select Set Active Project).

5.6.1.1 Create a thing

Things are used to connect devices to AWS IoT. A thing is a representation of a specific device.

Procedure

1. Click the AWS icon  in the activity bar to open the AWS Toolkit.
2. Choose a region and open the IoT folder, then right-click the Things folder and select Create Thing....
A popup appears and asks you for a new thing name.
3. Type a name and press Enter.
A thing icon followed by the name you provided is added in the Things folder.

5.6.1.2 Add a certificate and RSA key pair

A certificate is needed to establish a secure connection between AWS IoT and your device. A X.509 certificate is a digital certificate that uses the international X.509 public key infrastructure (PKI) standard to verify that a public key belongs to the hostname/domain, organization, or individual contained within the certificate.

Procedure

1. Right-click the Certificates folder and select Create Certificate....
A dialog box opens.
2. Select the AWS MQTT Demo project folder and save the certificate there.
Keil Studio saves an X.509 certificate and RSA key pair in the project folder.

5.6.1.3 Create a policy and attach it to the certificate

Policies define how AWS IoT and your device can interact with each other.

Before you begin

The policy used below allows any resource to access AWS IoT. In production, you must use a more secure policy. See the AWS documentation on [Policies and permissions in IAM](#) for more details.

Procedure

1. Select File > New File.
2. Type a name for the file, for example, `myPolicy.txt` and click OK.
3. Open the file and copy and paste the following block in the file:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Receive",
        "iot:Subscribe"
      ]
    }
  ]
}
```

```
    ],  
    "Resource": [  
        "*" ]  
    }  
  ]  
}
```

4. Go back to the AWS Explorer and attach the policy to the certificate created earlier. The certificate is inactive for now and no policies are found.
5. Right-click the certificate and select Attach Policy....
6. Select the policy in the popup that appears. The policy is added under the certificate.

5.6.1.4 Activate the certificate and attach it to the thing

The final step is to activate the certificate and attach it to your thing.

Procedure

1. Right-click the certificate and select Activate....
2. Right-click the thing and select Attach Certificate....
3. Select the certificate in the popup that appears. The certificate is added under the thing.

5.6.2 Configure connection settings in your project

There are different elements to configure in the `iot_config.h` and `socket_startup.c` files contained in your project. These elements control how the example application connects and authenticates with AWS IoT. Follow the steps below to configure the endpoint, the thing, the certificate and private key, as well as the root certificate. You also must set up your Wi-Fi connection.

5.6.2.1 Configure the `iot_config.h` file and modify the certificate and private key files

The sections you must update in the `iot_config.h` file are presented below.

About this task

Sections to update:

- `IOT_DEMO_SERVER`: Endpoint (or remote host) you are going to connect to.
- `IOT_DEMO_IDENTIFIER`: Name of the thing you created earlier.
- `IOT_DEMO_CLIENT_CERT`: Certificate (or client certificate), that is the `<ID>-certificate.pem.crt` file in your project. You must modify the certificate as explained below and copy/paste the certificate in the `iot_config.h` file.
- `IOT_DEMO_PRIVATE_KEY`: This is the client private key, that is the `<ID>-private.pem.key` file in your project. You must follow the same steps as for the client certificate.

- `IOT_DEMO_ROOT_CA`: This is the AWS trusted server root certificate. See the certificate provided below. You must add the root certificate to the `iot_config.h` file.

Procedure

1. Go to RTE > IoT_Client in the project folder and open the `iot_config.h` file.
2. Start by configuring the endpoint. Go to the AWS Explorer, right-click the IoT folder and select Copy Endpoint....
3. Go back to the `iot_config.h` file and paste the endpoint between the quotation marks in the `IOT_DEMO_SERVER` section. An endpoint looks like this: `a3xyzzyx.iot.us-east-1.amazonaws.com`.
4. Type the name of the thing you created between the quotation marks in the `IOT_DEMO_IDENTIFIER` section.
5. Open the `<ID>-certificate.pem.crt` file stored in the project folder.
 - a. Change `-----BEGIN CERTIFICATE-----` to `"-----BEGIN CERTIFICATE-----\n\`.
 - b. Add a backslash `\` at the end of every line.
 - c. Add `\n` to the second to last line.
 - d. Change `-----END CERTIFICATE-----` to `-----END CERTIFICATE-----\n"`.
6. Copy the content of the `<ID>-certificate.pem.crt` file and paste it in the `iot_config.h` file next to `IOT_DEMO_CLIENT_CERT`. Replace the quotation marks already present by a backslash, press Enter, and paste the certificate.
7. Do the same for the `<ID>-private.pem.key` file.
 - a. Change `-----BEGIN RSA PRIVATE KEY-----` to `"-----BEGIN RSA PRIVATE KEY-----\n\`.
 - b. Add a backslash `\` at the end of every line.
 - c. Add `\n` to the second to last line.
 - d. Change `-----END RSA PRIVATE KEY-----` to `-----END RSA PRIVATE KEY-----\n"`.
8. Copy the content of the `<ID>-private.pem.key` file and paste it in the `iot_config.h` file in the `IOT_DEMO_PRIVATE_KEY` section. Replace the quotation marks already present by a backslash, press Enter, and paste the private key.
9. For the root certificate, replace the quotation marks by a backslash, press Enter, and then copy and paste the following in the `IOT_DEMO_ROOT_CA` section:

```
"-----BEGIN CERTIFICATE-----\n\
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF\
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6\
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL\
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvcjEzMDEwNAQEBQYwKj9wMDAwMTUy\
b3QgQ0EgMTCCASAwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj\
ca9HgFB0fW7Y14h29J1o91ghYP10hAEvrAIthtOgQ3p0sqTQNroBvo3bSMgHFzZM\
906II8c+6zf1tRn4SWiw3te5djdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw\
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6\
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kd1L\
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQM\
jgSubJrIqq0CAwEAANcMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC\
AYYwHQYDVDR0OBByEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBcWUA\
A4IBAQCj8jdaQZChGsV2USggNiMOruYou6r41K5IPDB/G/wkjUu0yKGX9rbxenDI\
U5PMCCjymCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxh1b11BjJt/msv0tadQ1wUs\
N+gDS63pYaAcBvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv\
o/ufQJVtMVT8QtPHRh8jrdkPSHca2XV4cdFyQzR1b1dZwgJcJmApzyMZFo6IQ6XU\
5MsI+yMRQ+hDKXJioaldXgJukK642M4UwtBV8ob2xJNdd2ZhwLnoQdeXeGADbkpy\
rqXRfboQnoZsG4q5WTP468SQvvG5\n\
```

```
-----END CERTIFICATE-----\n"
```

5.6.2.2 Set up the Wi-Fi connection in the `socket_startup.c` file

Configure your Wi-Fi connection.

Procedure

1. Open the `socket_startup.c` file.
2. Enter the SSID and password of your Wi-Fi network in the `SSID` and `PASSWORD` sections.
You are all set! You can now send MQTT messages to AWS IoT.


5.6.3 Send MQTT messages

Debug your project to start sending messages.

Before you begin

Before sending MQTT messages, set up an external terminal to be able to see the messages sent from the board. You can also open the AWS MQTT test client from the AWS Management Console to see the messages received. When you are ready, debug your project to start sending messages.

Procedure

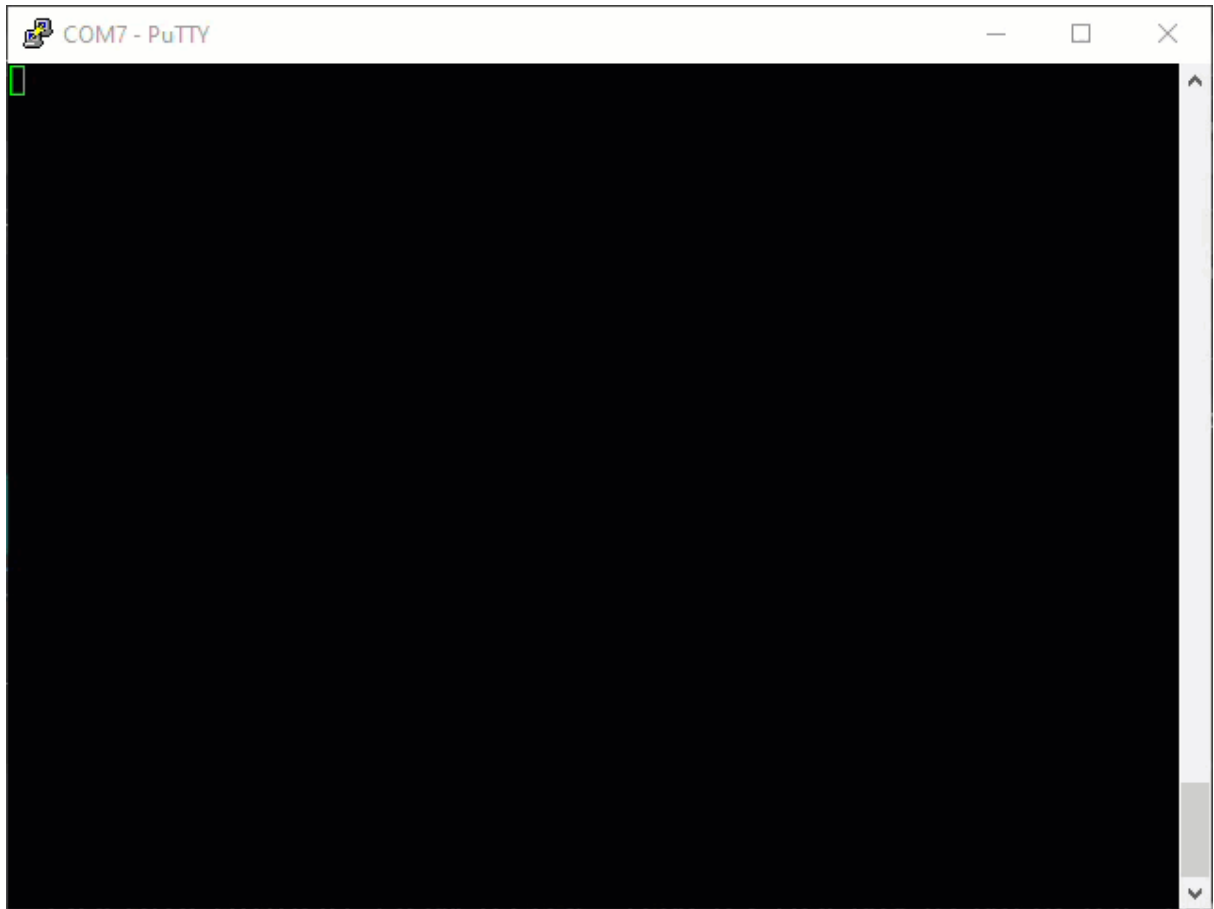
1. Connect your board to your computer. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target.
2. Click the debug button, . Keil Studio automatically builds and flashes the project to the board. The Debug view displays and the debug session starts.



If you are experiencing problems running the project on the board, set the Connect Mode preference to `underReset` to be able to flash a project to your board and do debugging. See [Advanced debugger settings](#) for more details on the different connect modes.

3. Check the output of the board in the terminal. When the connection is successful, the following message displays and MQTT messages start being sent:

```
AWS IoT Demo
Connecting to WiFi ...
WiFi network connection succeeded!
```

Figure 5-19: The connection is successful and MQTT messages are sent


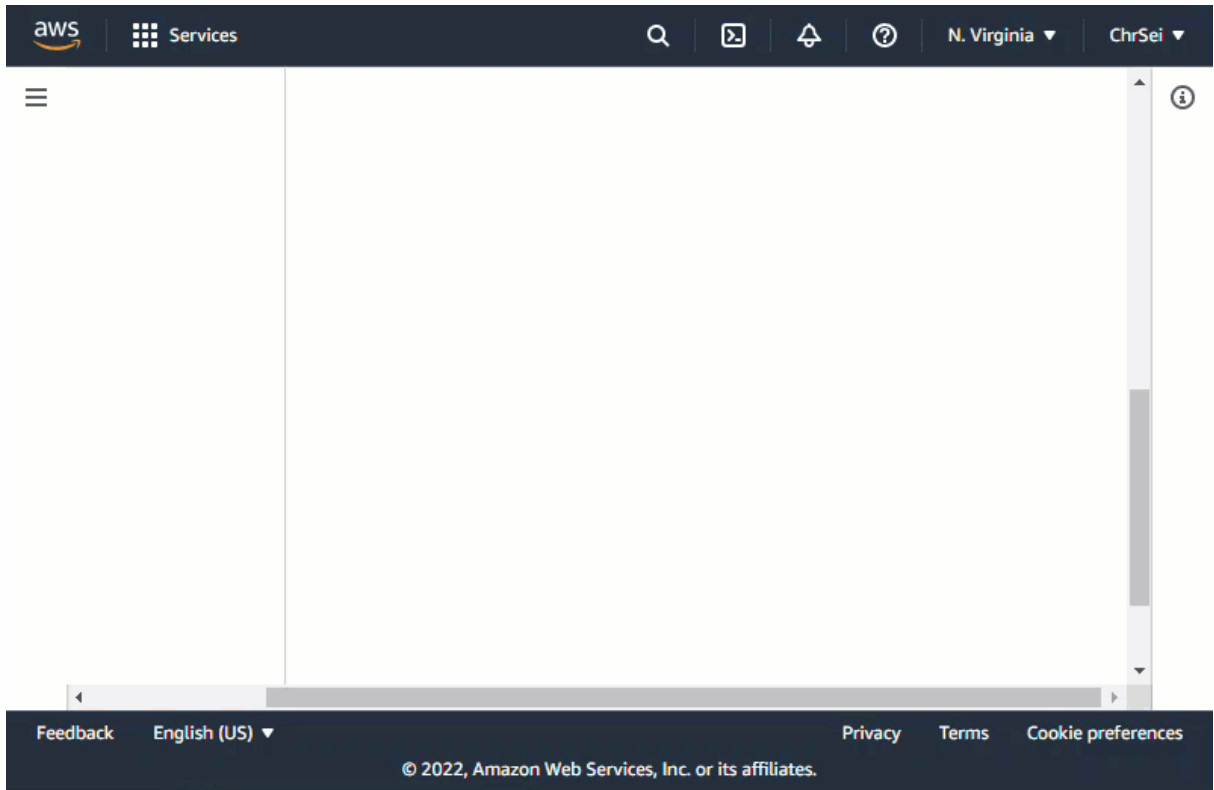
4. If the debugger is stopped, click the Continue button  to go through the code.
5. Check what happens from the AWS Management Console too. In IoT Core, go to Test > MQTT test client.
6. In the Subscribe to a topic tab, enter # in the Topic filter field and click Subscribe.

Figure 5-20: You can see the messages received by AWS IoT from the MQTT test client



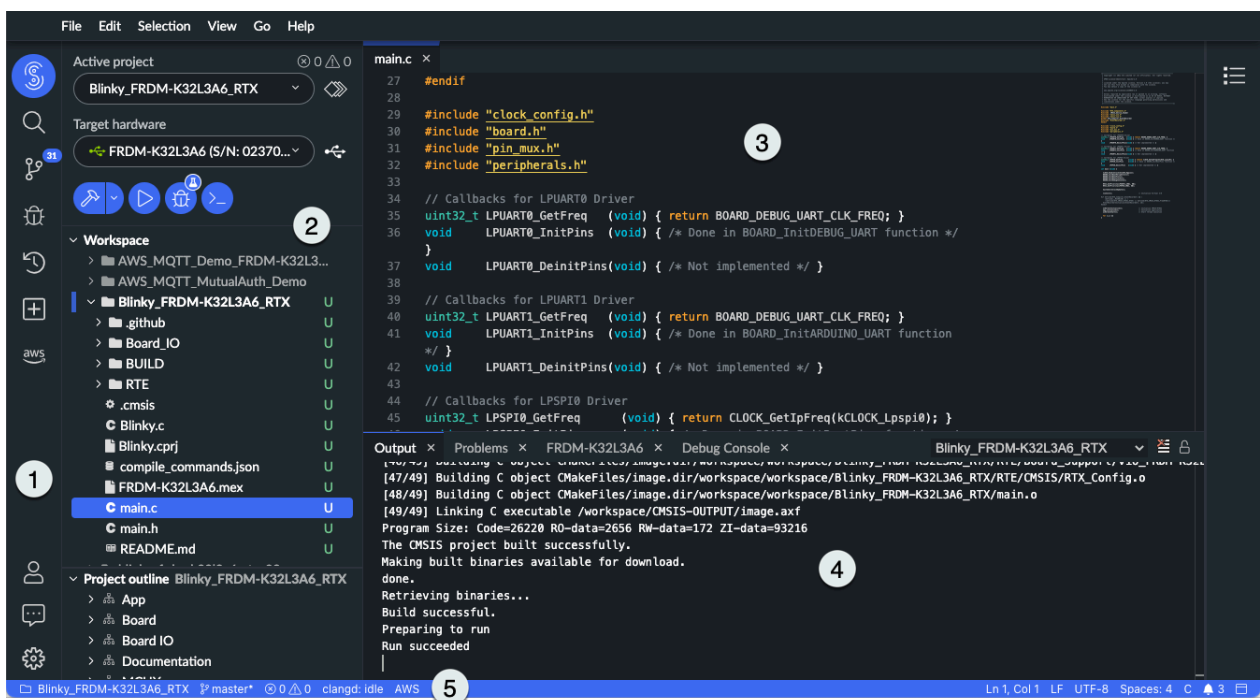
6. User interface

This chapter presents Keil Studio's user interface and the Command palette. It also gives you tips and tricks to help you use Keil Studio in an optimal way and customize the UI to your needs.

6.1 UI layout

Keil Studio's UI has an intuitive and customizable layout made of views.

Figure 6-1: UI layout



The UI is divided into different areas:

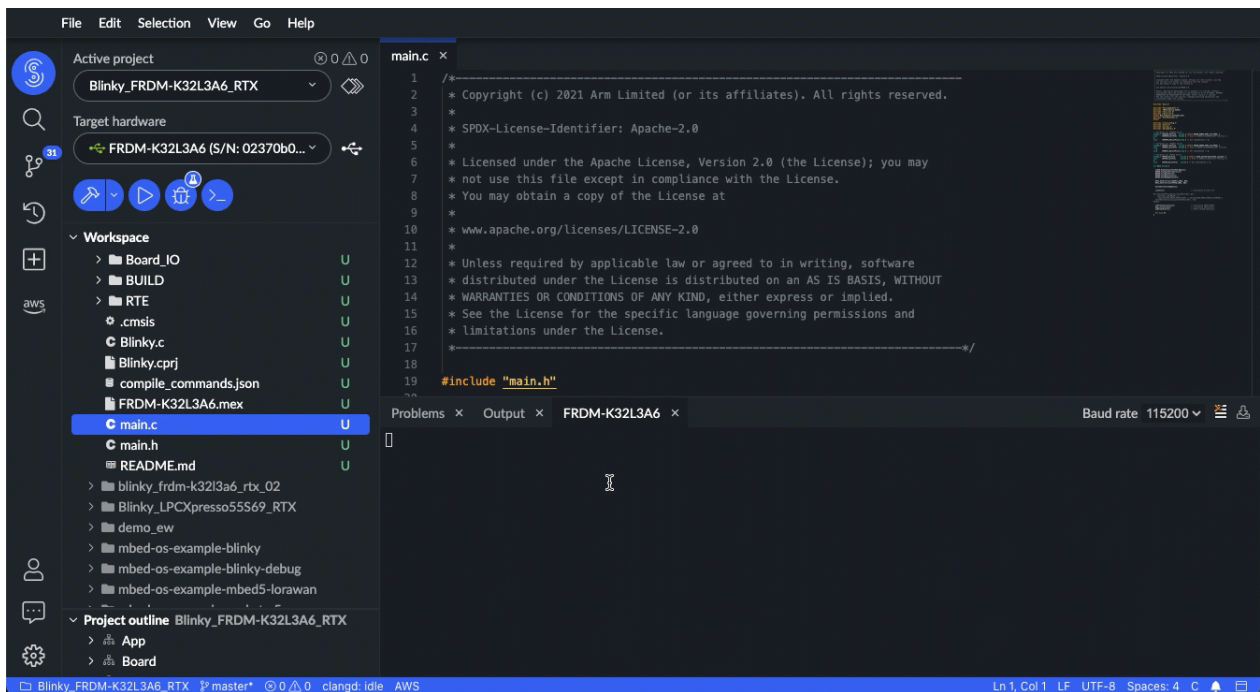
1. **Activity bar:** On the left-hand side, the activity bar allows you to switch between different views and displays context-specific indicators (for example, the number of Git outgoing changes). The activity bar also gives you access to the User Profile view, feedback and help options, and customizable settings.
2. **Side bar:** The side bar contains the Explorer view. The Explorer shows all the projects contained in your workspace organized by folders. The area just above the Workspace tree view shows the active project and build target (also called target hardware) currently selected for the project. The main buttons to build, build and flash, debug the active project and display the output of the board (Serial Monitor view) are also available from here. For CMSIS projects only, another area below the Workspace tree view shows the Project outline. The Project outline displays the software components and files contained in the active project organized in a logical way.

3. Editor: The editor on the right shows the content of the files you have opened from the Workspace tree view. It is the main area to edit your files. You can open as many editors as you like side by side, vertically or horizontally.
4. Panels: You can display different panels below the editor area for output or debug information, or errors and warnings.
5. Status bar: The status bar gives details about the active project and the files you are editing. If you are using source control, the branch you are working on displays there too.

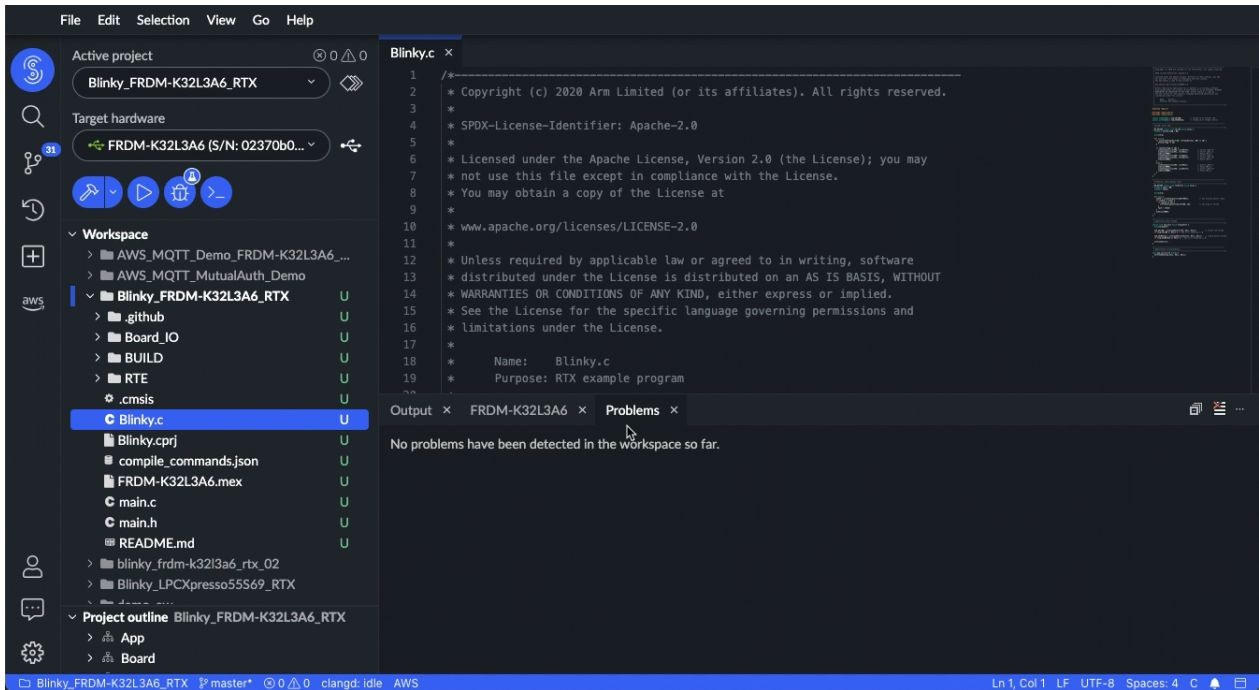
Most views are accessible from the View menu or from the activity bar. You can also search in View > Open View....

You can hide views easily. For example, try toggling the bottom panels with Ctrl+J (Windows) or Cmd+J (macOS) or the Explorer view with Shift+Ctrl+E (Windows) or Shift+Cmd+E (macOS).

Figure 6-2: Toggle views



You can also reorganize views the way you want by dragging and dropping them.

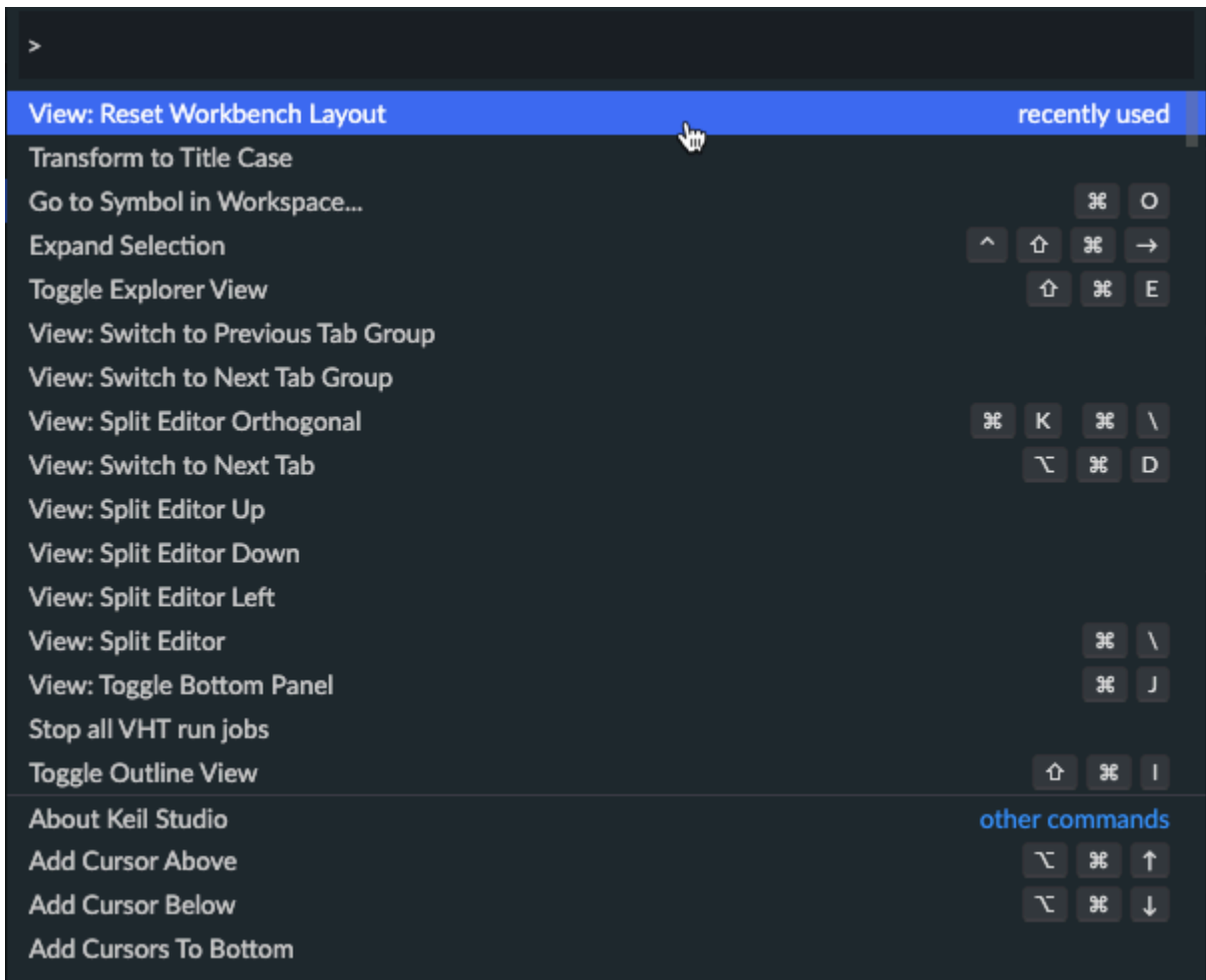
Figure 6-3: Drag and drop views to customize your workspace

To restore the default layout, go to **View > Find Command...** and look for the **View: Reset Workbench Layout** command. This command reloads Keil Studio with the default layout.

6.2 Command palette

Keil Studio is also accessible from the keyboard. The Command palette gives you access to all the functionality of Keil Studio, including keyboard shortcuts for the most common operations.

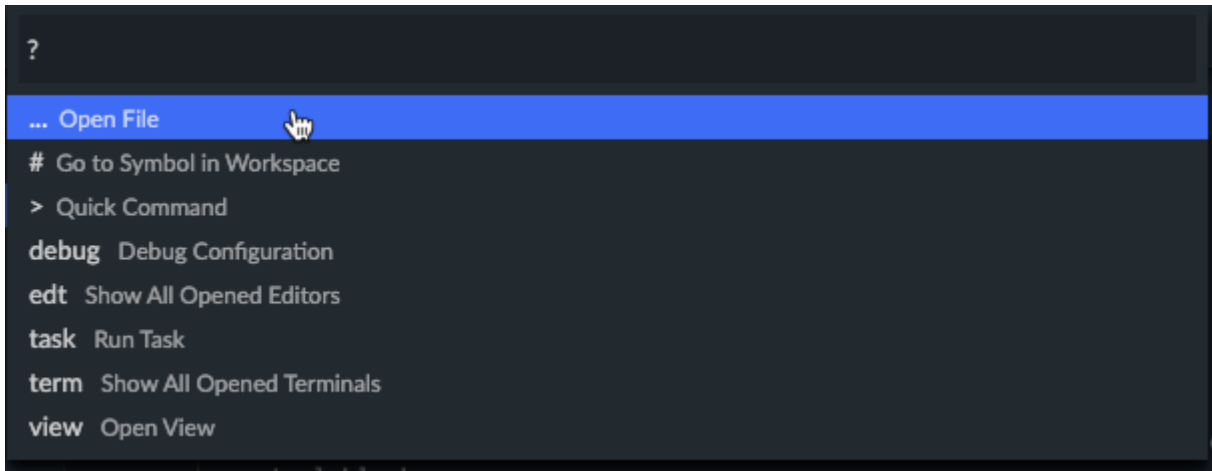
To bring up the Command palette, select **Find Command...** from the **View** menu or press **Ctrl+Shift+P** (Windows) or **Cmd+Shift+P** (macOS). Recently used commands appear at the top. To find what you are looking for, start typing the name of a command in the search field.

Figure 6-4: Recently used commands appear at the top

You can also open files or search for symbols from the Command palette.

- To open files: Press Ctrl+P (Windows) or Cmd+P (macOS). Recently opened files appear automatically, then file results are based on the string you enter in the search field.
- To search for symbols: Press Ctrl+P (Windows) or Cmd+P (macOS), then type # in the search field followed by the name of the symbol you are looking for.

If you type ?, a list of available commands you can execute from here displays:

Figure 6-5: Available commands

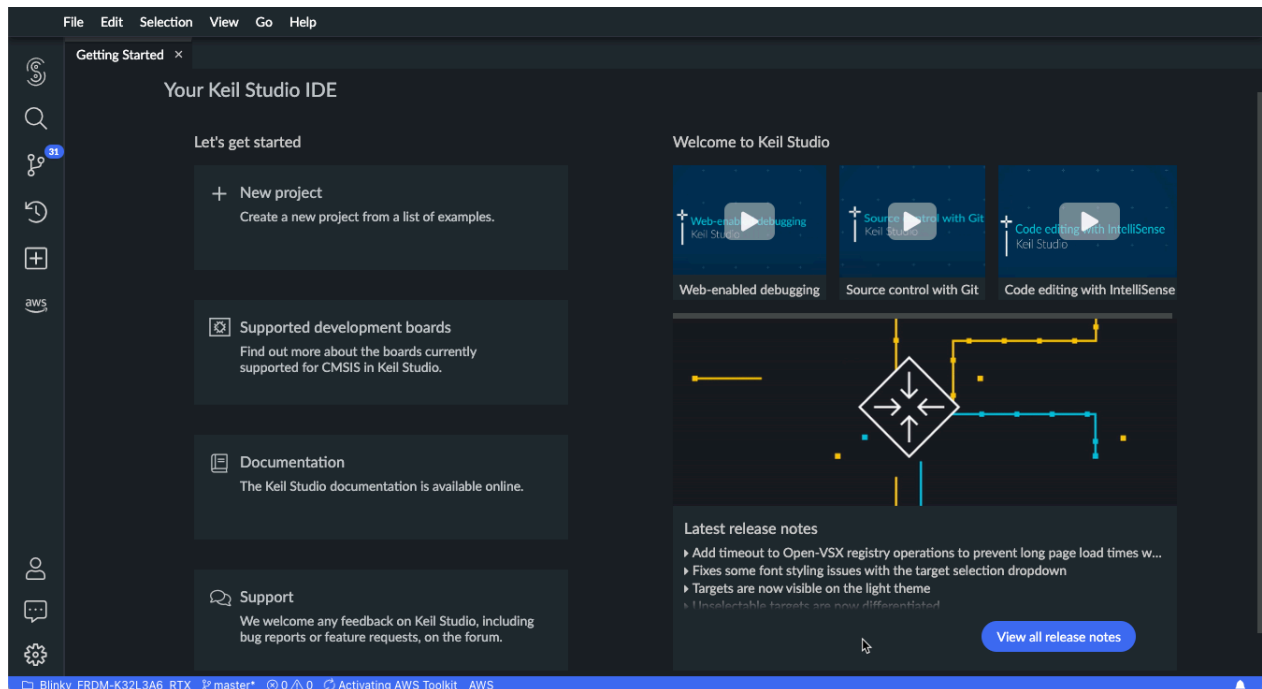
6.3 Tips and tricks

Here are some tips and tricks to help you work more efficiently with Keil Studio.

6.3.1 Getting Started page

If you are new to Keil Studio, start by checking the Getting Started page. It displays by default the first time you open Keil Studio. It is also accessible when you go to Help > Getting Started.

Figure 6-6: Getting Started page



This page gives you useful shortcuts, links to videos presenting Keil Studio's most popular features and a link to the release notes.

Do not forget to check the [tutorials](#) available in this guide as well, they provide step-by-step instructions to help you get started with Keil Studio.

6.3.2 Editor

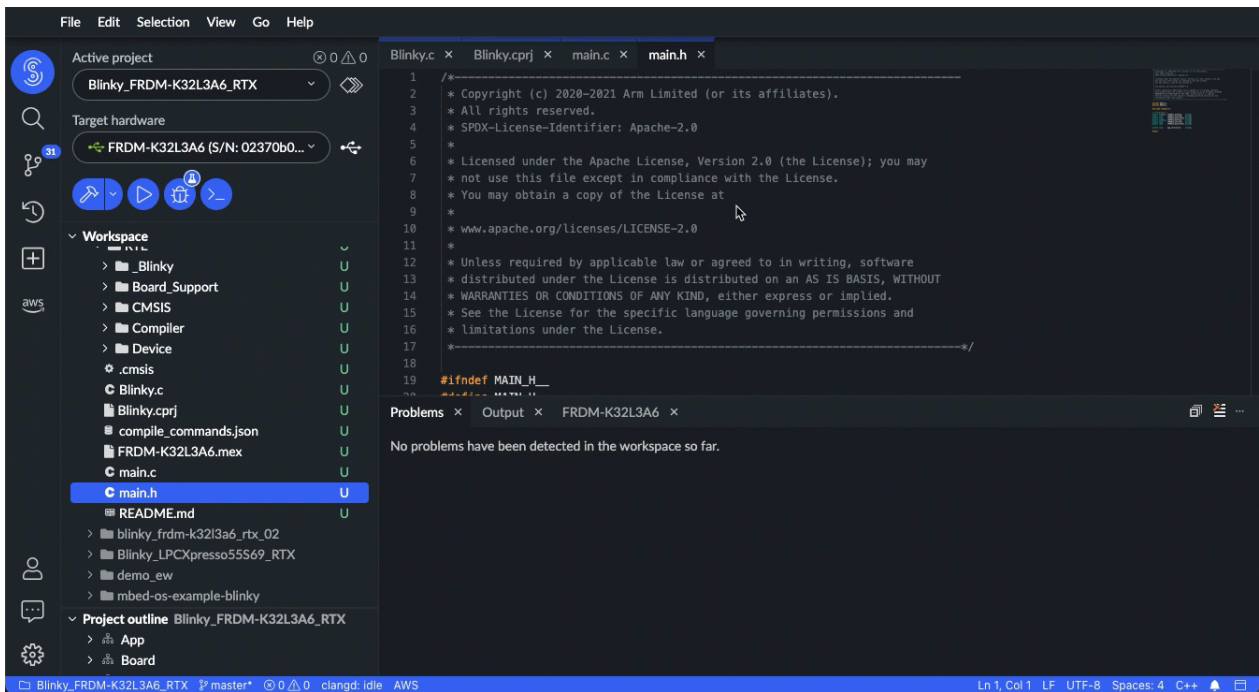
An essential part of Keil Studio is the editor where you edit your code. This section presents the editor and some useful editing tips.

6.3.2.1 Use side-by-side editing

You can open as many editors as you like side by side, vertically or horizontally.

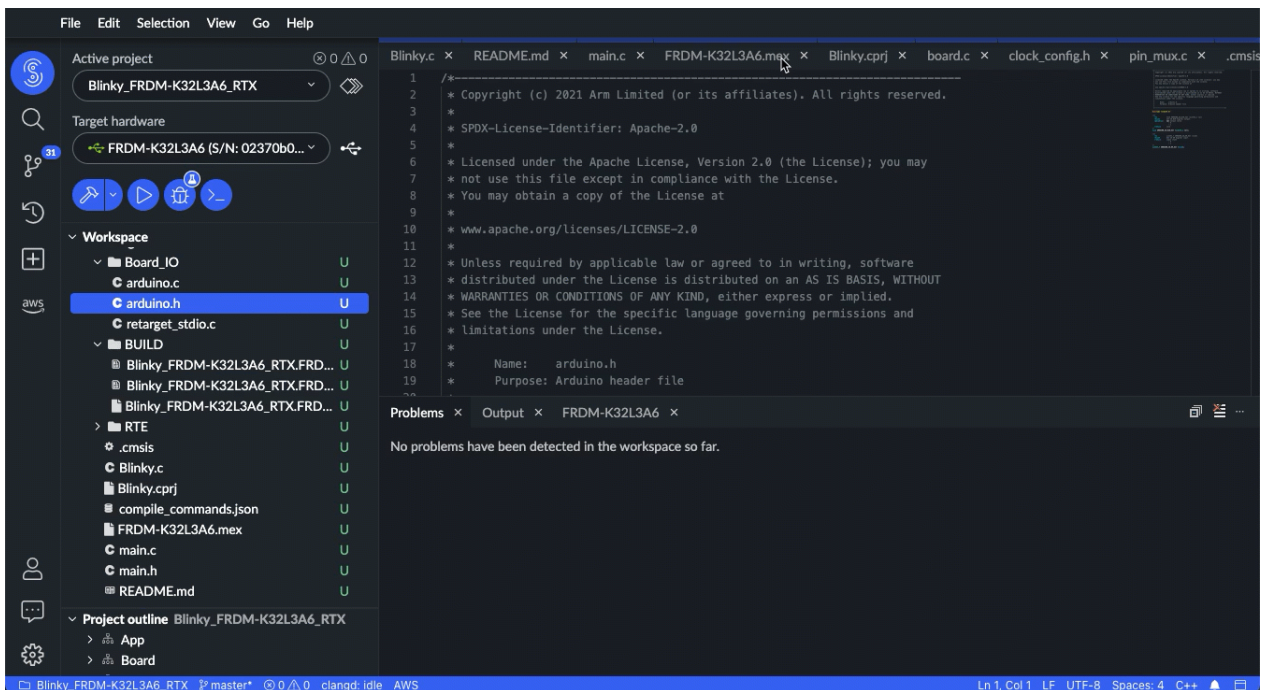
To split the active editor into two, press `Ctrl+\` (Windows) or `Cmd+\` (macOS). Whenever you open a new file, the editor that is active displays the content of that file. You can resize editors and reorder them easily. Drag and drop the title area of the editor you want to reposition or resize.

Figure 6-7: Split the active editor



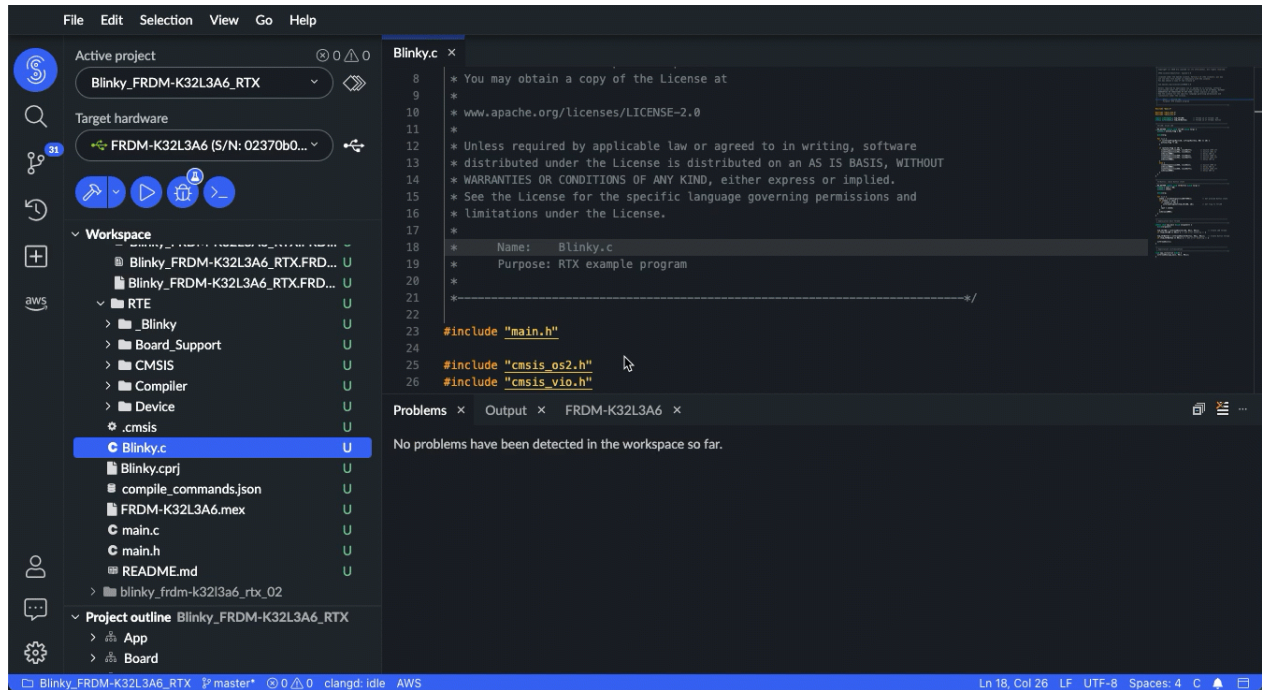
If there are many editors opened, use your mouse wheel to scroll through the tabs.

Figure 6-8: Scroll through the tabs



You can maximize the size of the editor region with the View > Collapse All Side Panels option. The default shortcuts are Alt+Shift+C (Windows) or Option+Shift+C (macOS).

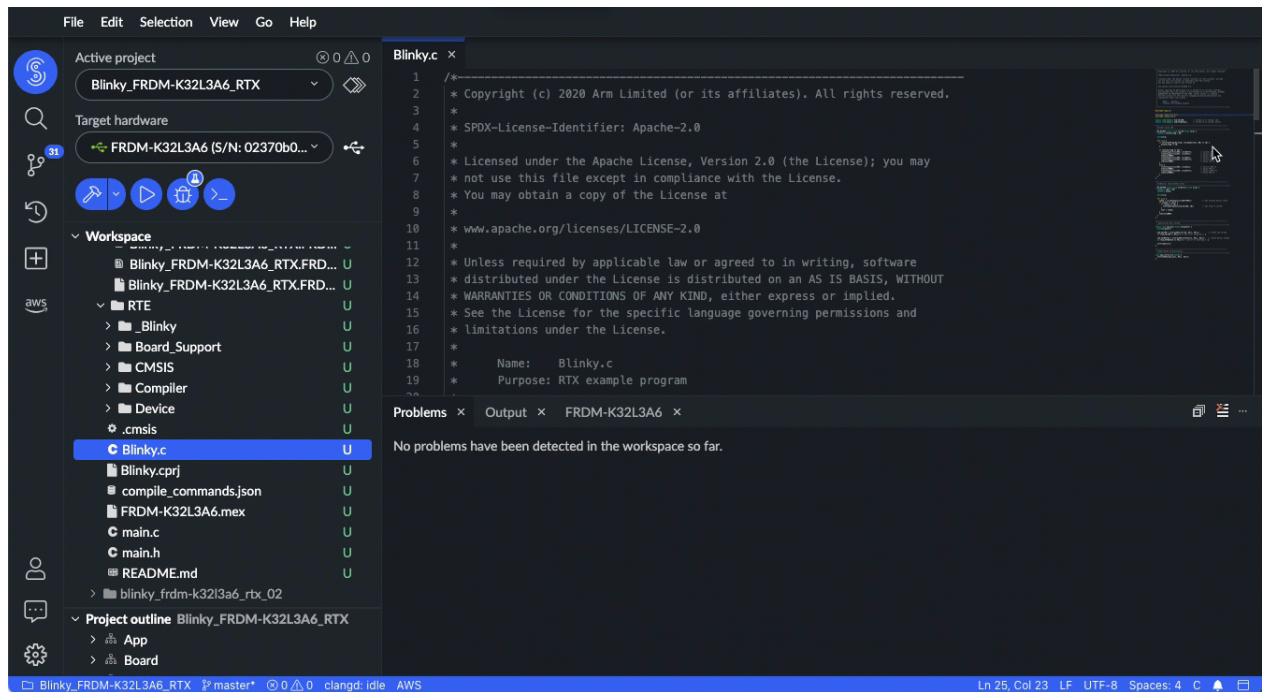
Figure 6-9: Maximize the editor region



6.3.2.2 Minimap

A minimap is available on the right side of the editor. You can click or drag the shaded area to jump to different sections of a file.

Figure 6-10: Minimap



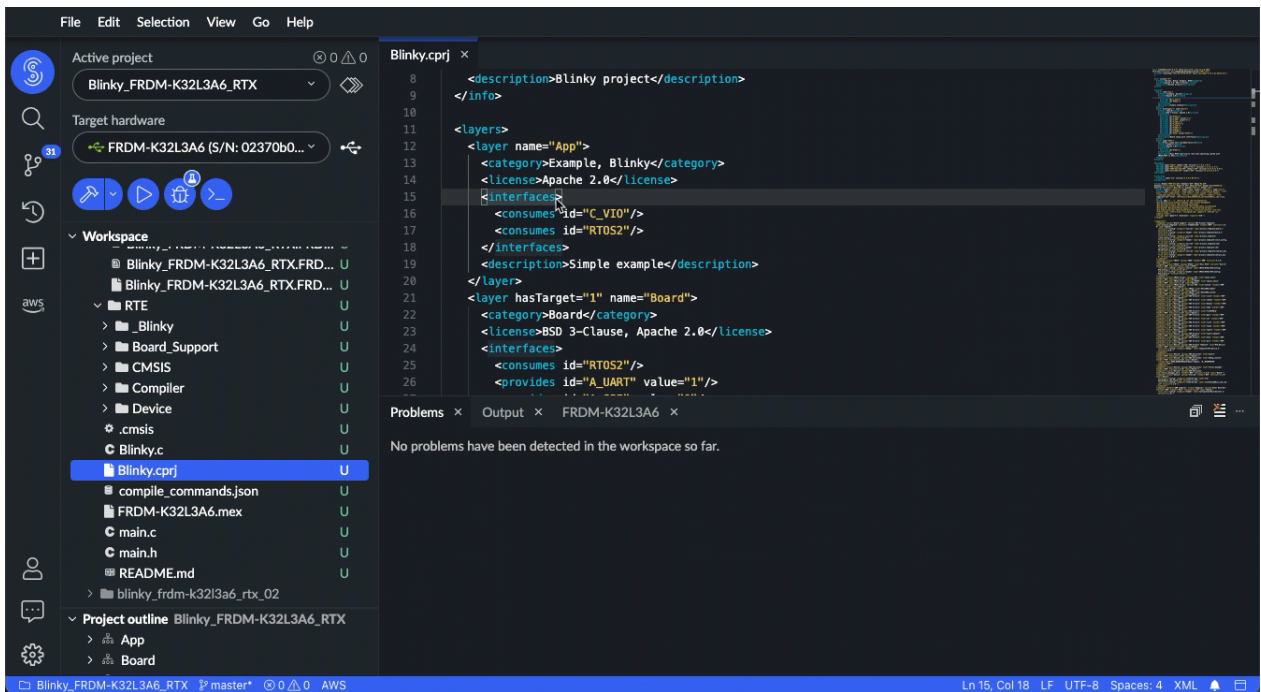
6.3.2.3 Editing tips

Here are some common features to make code editing faster.

Multi cursor selection

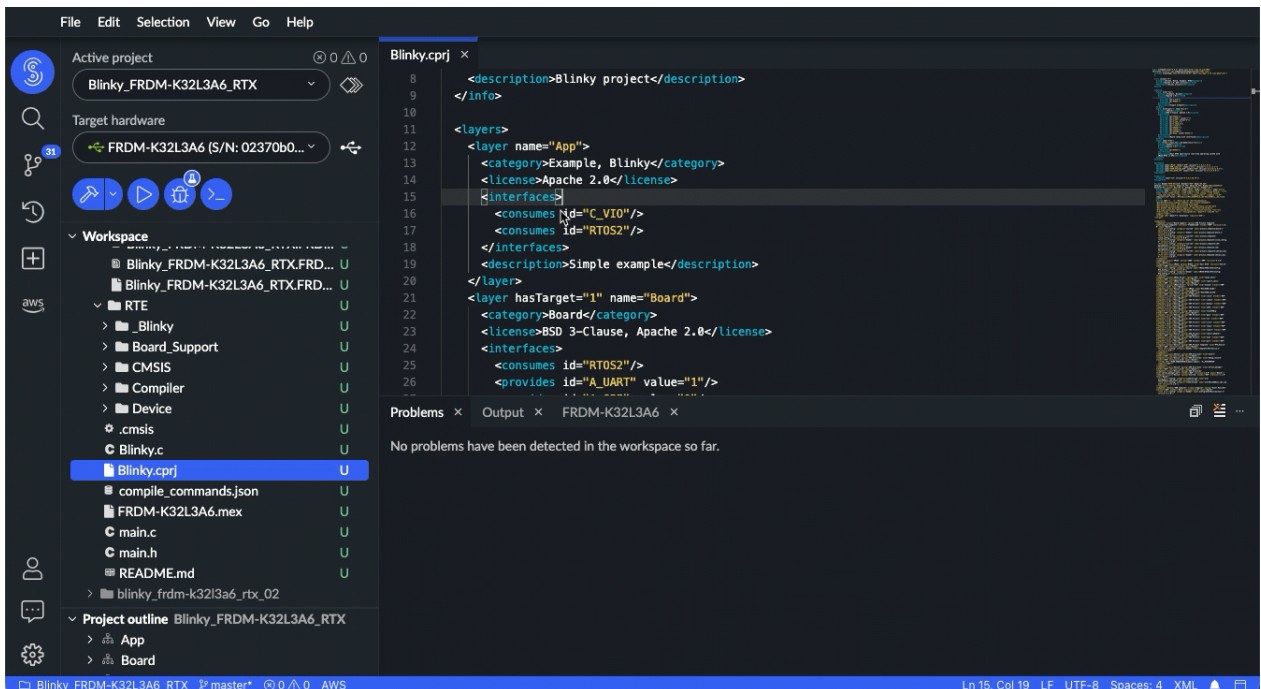
You can add cursors at different positions using your mouse and Alt+Click (Windows) or Option+Click (macOS).

Figure 6-11: Add cursors at arbitrary positions



To set cursors above or below the current position, use Alt+Ctrl+Up arrow (or Down arrow) (Windows) or Option+Cmd+Up arrow (or Down arrow) (macOS).

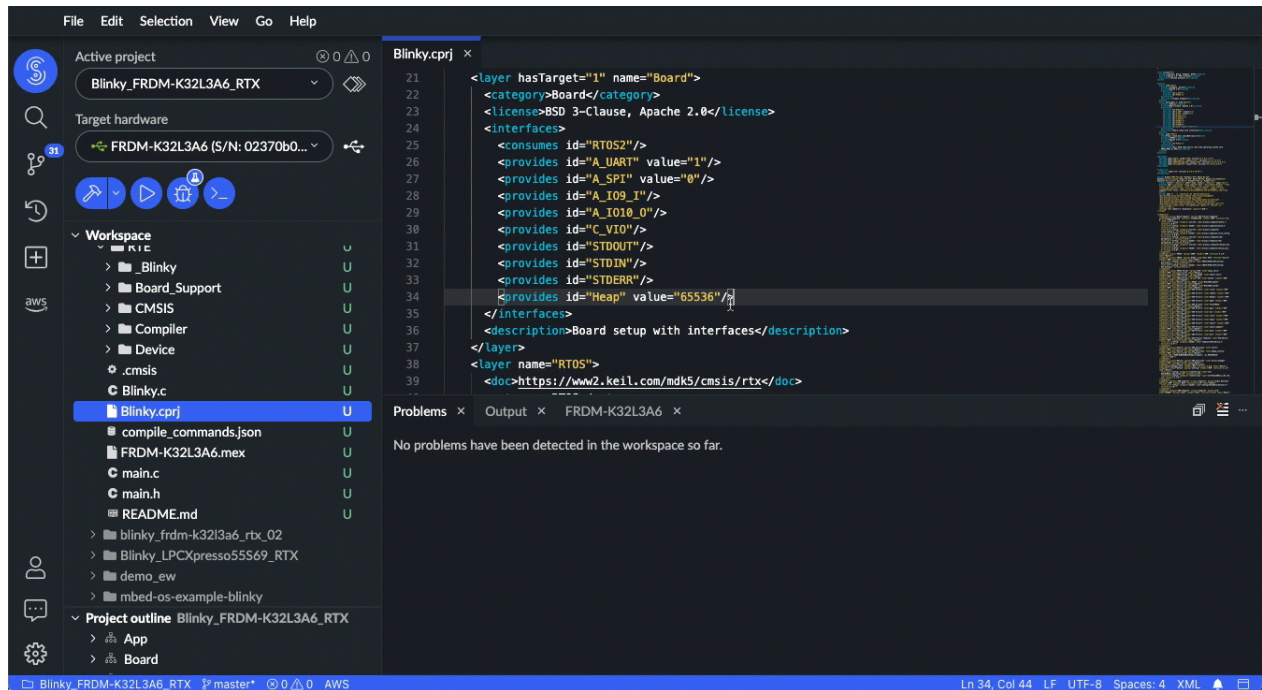
Figure 6-12: Add cursors above or below the current position



Column selection

You can select blocks of code by holding Shift+Alt (Windows) or Shift+Option (macOS) and dragging your mouse over what you want to select.

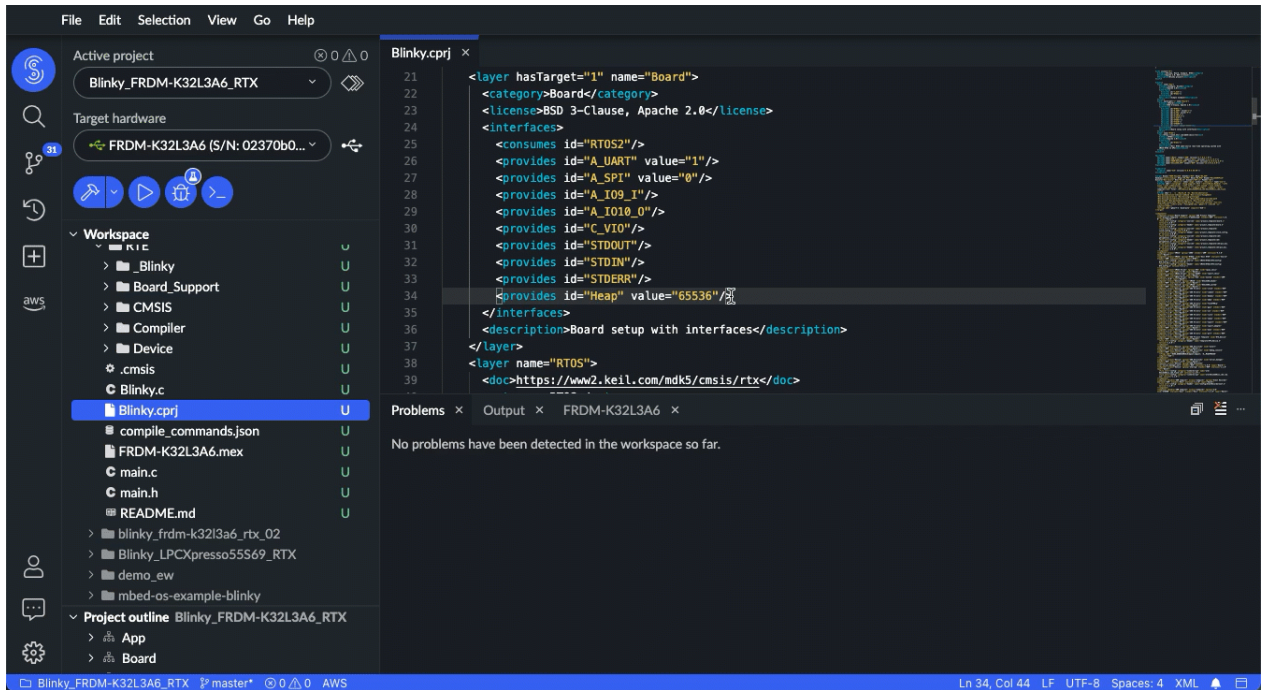
Figure 6-13: Select a column of code



Copy line up or down

To copy a line up or down, use Shift+Alt+Up arrow (or Down arrow) (Windows) or Shift+Option+Up arrow (or Down arrow) (macOS).

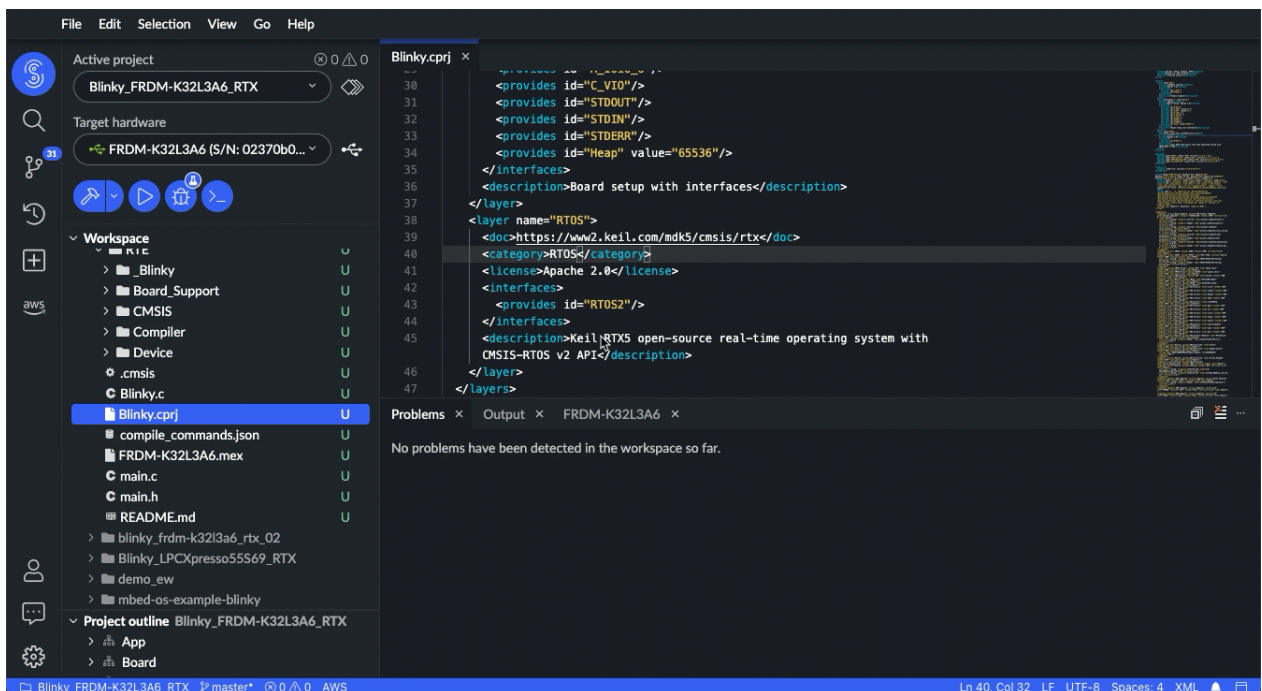
Figure 6-14: Copy line up or down



Move line up or down

To move a line up or down, use Alt+Up arrow (or Down arrow) (Windows) or Option+Up arrow (or Down arrow) (macOS).

Figure 6-15: Move line up or down



Navigate to beginning or end of file

To navigate to the beginning or the end of a file, use Ctrl+Up arrow (or Down arrow) (Windows) or Cmd+Up arrow (or Down arrow) (macOS).

6.4 Customize the UI

This section describes the preferences and shortcuts that you can update to tailor Keil Studio to your preferred setup.

6.4.1 Preferences

You can customize how Keil Studio operates to suit the way that you are used to working.

6.4.1.1 Change preferences


This section describes how to change preferences in Keil Studio.

Procedure

1. Go to File > Settings > Open Preferences.
There are two tabs: a User tab and a Workspace tab.

In the User tab, you can define preferences that apply to all workspaces. In the Workspace tab, you can define preferences for the workspace currently open only. Preferences set in the Workspace tab take precedence over preferences set in the User tab.

Note: In the current version of Keil Studio, there is only one workspace available per account, so you can use the User tab or the Workspace tab interchangeably.

The Open Preferences in JSON icon  opens a JSON file storing all the preferences you changed from their default values.

2. Select the User or the Workspace tab.
3. Click the preference that you want to change and modify the predefined values as needed.
Note: For some preferences, you can add a manual value in the JSON file.
4. For preferences that require a manual value, when you click the Edit in settings.json link, a preference ID is added in the JSON file. You must add the correct value for that preference ID. For example, if you click Edit in settings.json for the Files preference, the JSON file opens in a separate tab and shows the `files.associations` ID without a value. You can now associate new file extensions to a language:

```
{  
  "files.associations": {"*.myextension": "cpp"}  
}
```

In this example, your files with a `*.myextension` extension are recognized as C++ files.

Results

Once you have changed a preference, a blue line and a cogwheel icon appear to the left of the preference. You can click the cogwheel icon and select Reset Setting to reset the preference. You can also copy the preference ID and the value set for the preference in JSON format (Copy Setting as JSON option), or copy the preference ID only (Copy Setting ID option).



To look for a specific preference, type the preference name or ID in the Search Settings field.

Example 6-1: Common preferences example

In the following example, Keil Studio will:

- Only open files on double-click (instead of single-click).
- Use `\r\n` for the end of line character (instead of matching the operating system).
- Not auto indent.
- Not ignore white-space changes when comparing files.
- Auto close brackets only before a white space.

You can change the most common preferences without going to the Preferences list. For example, to switch from the default dark theme to a light theme, select File > Settings > Color Theme, or to indent with tabs, click the Spaces option on the information bar when the editor is open.

6.4.1.2 Set word wrap preferences

You can control word wrap in the editor through the Word Wrap preference. By default, this preference is off but if you set it on, text wraps at the width of the editor window.

Two other options are available for Word Wrap and work in combination with the value set for the Word Wrap Column preference. Word Wrap Column is the maximum number of characters that can be displayed on a single line.

- `wordWrapColumn`: When this option is selected, text wraps at the value set for Word Wrap Column.
- `bounded`: When this option is selected, text wraps at the width of the editor window or at the value set for Word Wrap Column, whichever is the smaller value of the two widths.

Once you have set these preferences, select View > Toggle Word Wrap to toggle between the different options for the Word Wrap preference.

6.4.2 Keyboard shortcuts

Keil Studio lets you perform most tasks from the keyboard with keyboard shortcuts (also called keybindings). You can modify existing keyboard shortcuts or add new ones to commands that do not yet have shortcuts associated with them.


6.4.2.1 Change keyboard shortcuts


This section describes how to change keyboard shortcuts in Keil Studio.

Procedure

1. Select File > Settings > Open Keyboard Shortcuts.
2. In the Search keybindings field, type the name of a command or keybinding. You can also do a search on the context and "when" clauses. Note that you can click the Clear Keybindings Search

Input icon  to clear the field, if needed.


3. Hover over the shortcut you want to modify and click the Edit Keybinding pen icon  to the left of the command. You can also double-click the shortcut to edit it. The Edit Keybinding for <command> dialog box opens.
4. Update the shortcut and click OK. To find out what keys you can use, see the Supported Keys section in [Using the JSON file](#).

To make further changes, you can click the Open Keyboard Shortcuts in JSON icon . The Open Keyboard Shortcuts in JSON icon opens the `keymaps.json` file which stores all the shortcuts that have been changed from their default values. Change the lines corresponding to the shortcut or shortcuts you want to modify. For more information, see [Use the JSON file](#).

6.4.2.2 Reset keyboard shortcuts

This section describes how to reset keyboard shortcuts in Keil Studio.

Procedure

1. Search for the keyboard shortcut you want to reset.
2. Hover over the shortcut and click the Reset Keybinding icon  to the left of the command. The Reset keybinding for <command> dialog box opens.
3. Click OK. Alternatively, you can reset a shortcut from the Edit Keybinding for <command> dialog box with the Reset button. You can also delete the lines corresponding to the shortcut you want to reset in the `keymaps.json` file. For more information, see the next section.

6.4.2.3 Use the JSON file

All the changes that you make to keyboard shortcuts are stored in a `keymaps.json` file. Once you have made changes to shortcuts, you can update or delete your changes directly from the JSON file.

Keyboard rules

When you open the `keymaps.json` file, you can see that two blocks are present for each shortcut that is modified. The current value of a shortcut is presented first, followed by the previous values for that shortcut. The blocks contain what are called *keyboard rules*.

Each rule consists of:

- A `"command"` containing the identifier of the command to execute.
- A `"keybinding"` that describes the pressed keys.
- An optional `"when"` clause that contains a boolean expression which evaluates depending on the current context. If there is no `"when"` clause, the keyboard shortcut is globally always available.

For example, if the keyboard shortcut for the Copy Line Down command was modified to use `shift+down` instead of `shift+alt+down` to trigger the command when using the editor, you would see:

```
{
  "command": "editor.action.copyLinesDownAction",
  "keybinding": "shift+down",
  "when": "editorTextFocus && !editorReadOnly"
},
{
  "command": "-editor.action.copyLinesDownAction",
  "keybinding": "shift+alt+down",
  "when": "editorTextFocus && !editorReadOnly"
}
```

To reset the keyboard shortcut, you can modify the `"keybinding"` element in the first block or delete both blocks.



Note

It is not currently possible to change the `"command"` identifiers or `"when"` clauses.

Supported keys

The following keys are supported:

To use `ctrl` on Windows or Linux and `cmd` on macOS, use `ctrlcmd`.

You can use `shift`, `ctrl`, `alt`, `meta`, `option (alt)`, `command (meta)`, `cmd (meta)` as modifiers.



If you define a custom shortcut with `cmd`, `command`, or `meta` in your `keymaps.json` file, the same `keymaps.json` file will not work on a Windows or Linux machine because these machines do not have equivalent keys.

You can also use the following strings for special keys: `backspace`, `tab`, `enter`, `return`, `capslock`, `esc`, `escape`, `space`, `pageup`, `pagedown`, `end`, `home`, `left`, `up`, `right`, `down`, `ins`, `del`, and `plus`.

Chords (two separate keypress actions) are supported and must be separated with a space.

7. Work with standalone CMSIS projects and CMSIS solutions

This chapter describes the differences between standalone CMSIS projects and CMSIS solutions and how to work with them in Keil Studio.

7.1 Standalone CMSIS projects

You can work with standalone CMSIS projects that contain a single `.cprj` project file. CPRJ is a generic CMSIS-aware project file format that allows IDEs and command-line build tools to share the same projects. See this [page](#) for more details on the CPRJ project file format.

7.2 CMSIS solutions

You can also work with solutions that can contain one or more projects. A solution is a container used to organize related projects that are part of a larger application and that can be built separately.

See [Project setup for related projects](#) for a solution example.

Solutions are defined in YAML format using `*.csolution.yaml` files. A `*.csolution.yaml` file defines the complete scope of an application and the build order of the projects the application contains. Individual projects are defined using `*.cproject.yaml` files. A `*.cproject.yaml` file defines the content of an independent build - it is the solution-based equivalent of a `.cprj` file.

The `*.csolution.yaml` and `*.cproject.yaml` files of a solution can be manually edited. Keil Studio integrates the Red Hat YAML Language Support extension and uses YAML schemas to make the editing of these files easier. See the [vscode-yaml repository](#) for more information on the extension.

The solution and project files are then translated to the CPRJ project file format.

See the [Overview](#) of the `csolution - CMSIS Project Manager` project and the [project examples](#) presented to understand how solutions and projects are structured. The `csolution - CMSIS Project Manager` project is currently under development and part of the [Open-CMSIS-Pack](#) open source project.

7.3 Create, import or clone a standalone CMSIS project

This section describes how to start working with standalone CMSIS projects in Keil Studio.

You can:

- Create a new project from an example shipped with Keil Studio.

- Import an example project from keil.arm.com.
- Clone an existing project from a Git hosting service (such as GitHub).

Arm recommends that you use the example projects provided. The examples use predefined `.cprj` templates. See this [page](#) for more details on the CPRJ project file format.



If you do not already have an Arm account, [create one](#) and log into Keil Studio with it. Alternatively, you can use your Mbed account.

7.3.1 Create a project from a CMSIS example project

You can create a CMSIS project from an example.

Procedure

1. Select File > New... > CMSIS Project.
The New Project dialog box opens.
2. Click the Example project drop-down list and select an example from the list.
Each example has a `readme.md` file that explains the details of that example.
3. In the Project name field, edit the name of the new project or keep the name provided by default.
4. Check the default options:
 - Keil Studio sets the newly created project as the active project. Build and run commands only apply to the active project. Clear the checkbox if you do not want to make the new project active.
 - Keil Studio initializes the project as a Git repository. Clear the checkbox if you do not want to turn your project into a Git repository. See [Configure a project for source control and collaboration](#) for more details.
5. Click Add project. Keil Studio creates the project in your workspace.

7.3.2 Import a CMSIS example project from the Keil Studio website

You can import a CMSIS project from keil.arm.com.

Procedure

1. Go to keil.arm.com.
2. Click the Hardware menu to see the list of supported hardware.
You can search for a board by name or vendor, or filter the list by vendor or core. You can also display boards with example projects with the Only include boards with example projects checkbox.
3. Once you have found an example project, click the Open in Keil Studio button for the project you want to use.

4. Log into Keil Studio if you are not already logged in.
5. Keil Studio opens. Confirm the project name in the Import Project dialog box. Keil Studio sets the newly imported project as the active project by default.
6. Click Add project. The project loads to your workspace and is the active project.

7.3.3 Clone a CMSIS project

You can clone existing CMSIS projects from a Git hosting service (such as GitHub).

Before you begin

For projects on a Git hosting service, you can clone public repositories without setting up your Git credentials. If you want to clone a private repository, [set up Git credentials](#).

For more information about what cloning a repository means, see the [git-clone chapter](#) on git-scm.com.

Procedure

1. Open the File menu and select Clone....
The Clone dialog box displays.
2. Paste the full HTTPS or SSH URL of the relevant web page and (optionally) edit the project name.
This can be a page on a Git hosting service.

Keil Studio sets the newly cloned project as the active project by default. Build and run commands only apply to the active project. Clear the checkbox if you do not want to make the new project active.

3. Click Add project.
You can push changes back to the remote repository from Keil Studio. For more information, see [Source control](#).

7.3.4 Further resources about CMSIS

You are now ready to work on your project.

You can learn more about [CMSIS](#).

When you are ready, [build and run your project](#).

7.4 Manage a standalone CMSIS project and its software components

This section describes how to manage standalone CMSIS projects and their software components.

7.4.1 Manage Software Components view and Project outline

The Manage Software Components view in Keil Studio allows you to see which software components are included in a project.

You can select the software components to include in your project and build the project using different combinations of pack and component versions, and different versions of a toolchain.

The Project outline is accessible from the Explorer view and gives you an overview of what is included in your project (software components and files). You can easily manage your project files from the Project outline and organize them by groups to simplify the project maintenance.


7.4.2 Open the Manage Software Components view

This topic describes how to open the Manage Software Components view.

Before you begin

The Manage Software Components view shows all the software components in the active CMSIS project. From the Manage Software Components view you can see all the component details (called attributes in the [CMSIS-Pack specification](#)).

Procedure

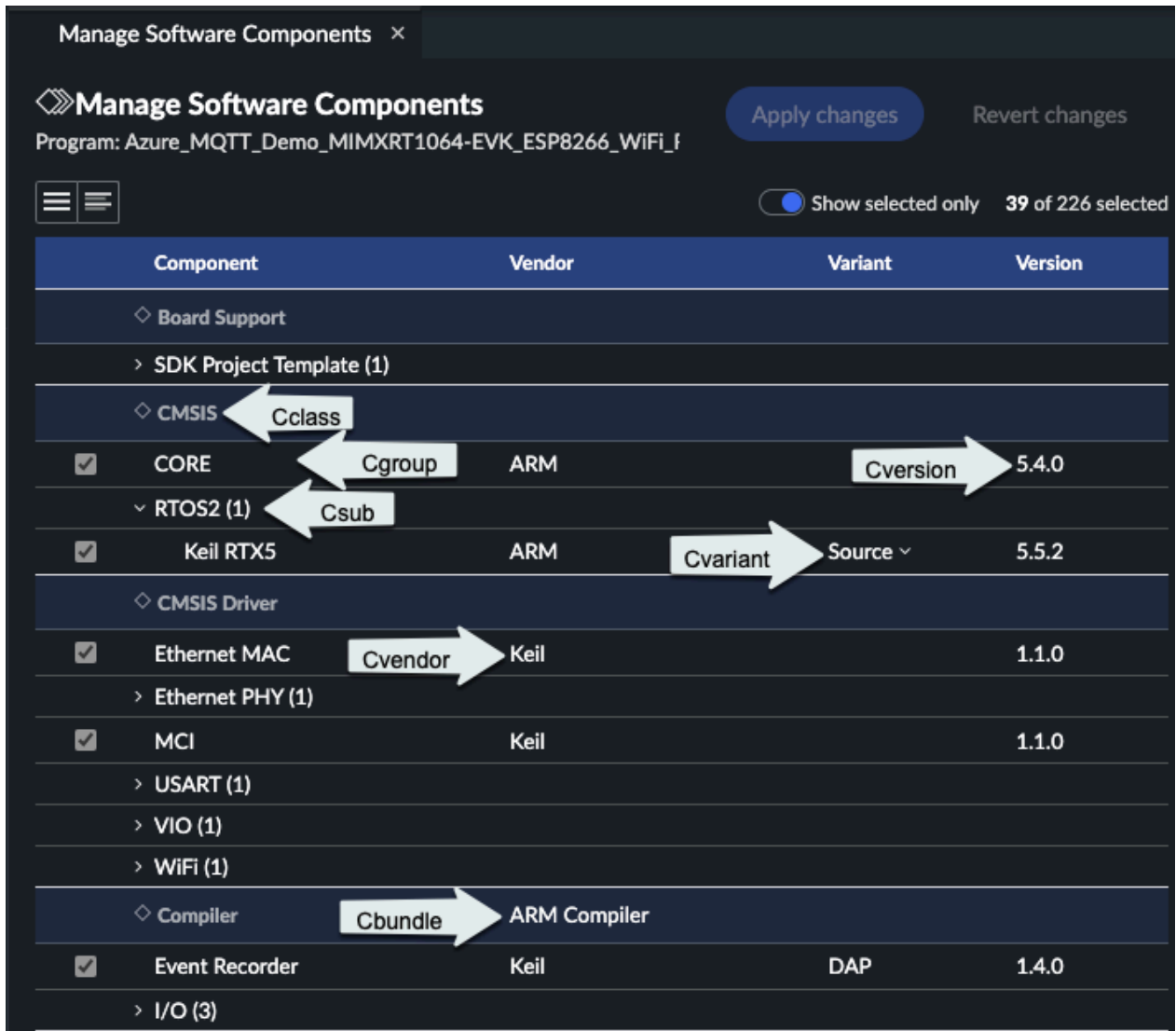
1. Set the project you want to work on as the active project.
2. Click the Open Software Components view icon  from the Explorer.

Results

The Manage Software Components view opens.

The Show selected only toggle button is on by default. Only the components included in the active project display. If you switch off the toggle, you can see all the components that are available for use.

Figure 7-1: The ‘Manage Software Components’ view showing the components included in the active project



The first column in the Manage Software Components view shows how many instances of the component are deployed in the project. If the component has a selected checkbox, one instance is deployed. If the checkbox is not selected, it is not deployed in the project. If a number input displays, multiple instances of this component can be deployed to a project.

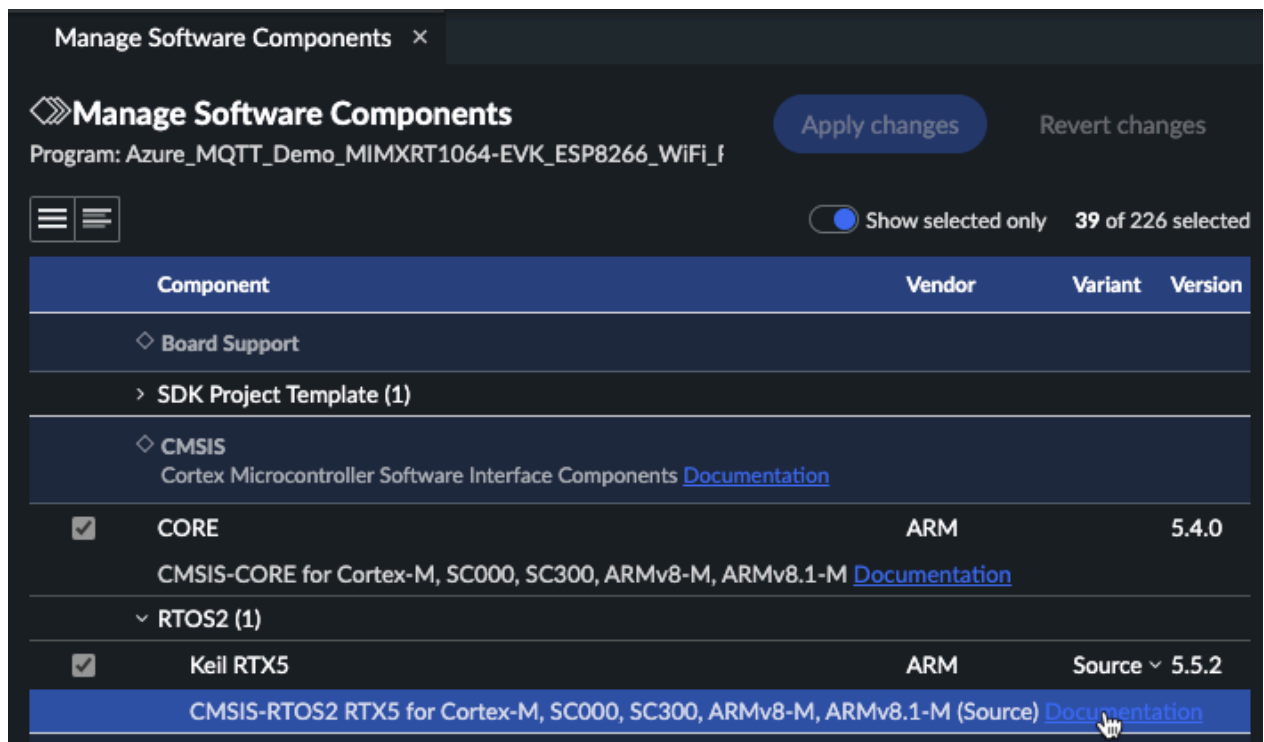
The CMSIS-Pack specification states that each software component should have the following attributes:

- Component class (Cclass): A top-level component name. For example: CMSIS. For a full list of component classes, see [Software Component Cclasses](#).
- Component group (Cgroup): A component group name. For example: CORE for the CMSIS component class.
- Component version (Cversion): The version number of the software component.

Optionally, a software component might have these additional attributes:

- Component subgroup (Csub): A component subgroup that is used when multiple compatible implementations of a component are available. For example: Keil RTX5 implements CMSIS > RTOS2.
- Component variant (Cvariant): A variant of the software component is typically used when the same implementation has multiple top-level configurations, like Source for RTOS2 > Keil RTX5.
- Component vendor (Cvendor): The supplier of the software component. For example: Keil.
- Bundle (Cbundle): Allows you to combine multiple software components into a software bundle. Bundles have a different set of components available. All the components in a bundle are compatible with each other but not with the components of another bundle. For example: ARM Compiler for the Compiler component class.

Figure 7-2: Documentation links in the 'Manage Software Components' view



Documentation links are available for most component classes, component groups and subgroups when you click their names.

With the Search field, you can do a search on the component groups and subgroups.

7.4.3 Modify the software components in your project

This topic describes how to select software components.

Before you begin

For standalone CMSIS projects, you can only add components from packs that are already selected for a given project.

Procedure

1. Switch the Show selected only toggle button off to see all the components available.
2. Select or clear components as needed using the checkboxes. If a number input displays instead of a checkbox, it means multiple instances of this component can be deployed to a project. For some components, you can also select a vendor, variant or version.

The Revert changes button allows you to revert all your unsaved changes at once.

3. Click Apply changes.

7.4.4 Manage your project files from the Project outline

The Project outline allows you to see what is in your project (software components and files). You can change how your project is structured using groups. Groups are useful to structure code files into logical blocks.

You can for example sort files according to:

- Their function (for example, you can separate CPU-related files from system calls and other source files).
- Separate groups for different teams.
- Memory they will execute in.
- Compiler and linker settings (for example, optimization level).

File groups have no influence on the build process for your project (unless you specify command line options for the groups in the `.cprj` file of the project).

When managing groups, you can:

- Create a group.
- Add a file from the Workspace to a group.
- Delete a group. This action deletes the group and removes the files contained in the group from the Project outline (the files are still available from the Workspace).
- Move a group inside another group.
- Rename a group.

With files, you can:

- Add a file from the Workspace to the Project outline.
- Move a file to a different group.
- Remove a file from a group (the file is still available from the Workspace).



Changes you make to the Project outline are reflected in the `.cpzj` file.

Procedure

1. Hover over the Project outline and click the ... button, then select New Group.
2. Enter the name of the group in the popup and press Enter.
3. Right-click the group and select Add File to Group....

A dialog box opens.

4. Select the file you want to add to the group and click Add.
5. Select a file category in the popup.

For some file types, the file category is automatically set (for example, *.c files are in the `sourceC` category by default, so there is no file category selection). The file categories available are described in the [Build](#) section of the CMSIS documentation.

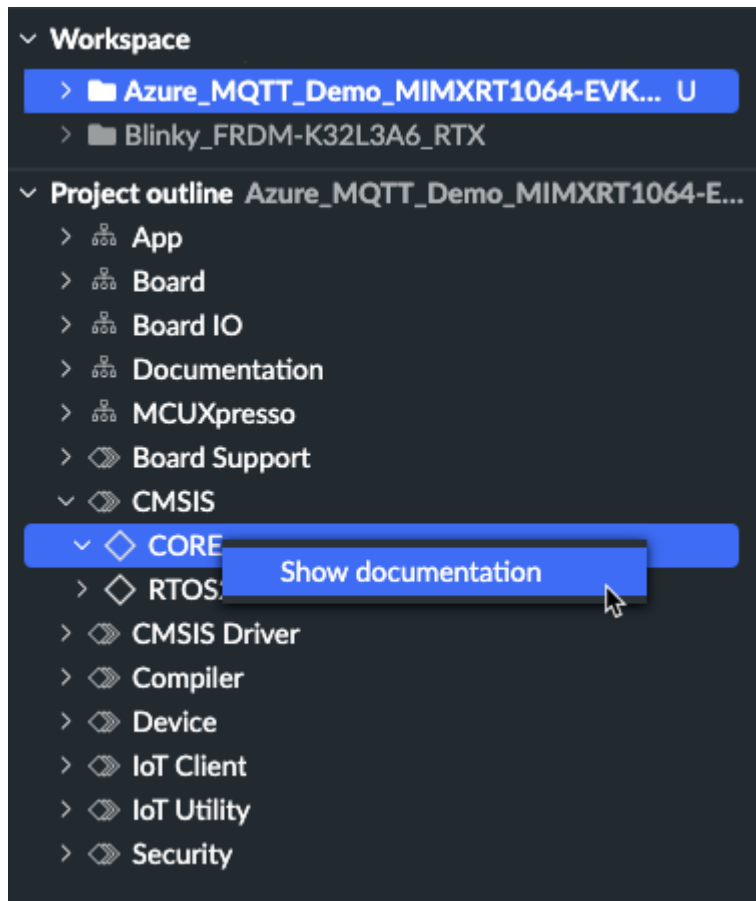
Note that you can also add a file to a group from the Workspace. Right-click a file in the Workspace and select Add to Project Outline, then select a destination group and a file category.

6. If you need to move a group inside another group, right-click the group and select Move to Group....
7. Select a destination group in the popup.
8. Similarly, if you need to move a file to a different group, right-click the file and select Move to Group....
9. Select a destination group in the popup.

Results

You can easily create groups, add files to groups, and reorganize groups and files the way you want to.

Note that you can also open the documentation available for component groups and subgroups from the Project outline. Open a class, then right-click a group (for example: CORE for the CMSIS component class) and select Show documentation.

Figure 7-3: The 'Project outline'

7.5 Manage a CMSIS solution and its software components

This section describes how to manage CMSIS solutions and their software components.

7.5.1 Manage Software Components view

The Manage Software Components view in Keil Studio allows you to see which software components are included in a solution.

You can select the software components to include in your solution and build the solution using different combinations of pack and component versions, and different versions of a toolchain.


7.5.2 Open the Manage Software Components view

This topic describes how to open the Manage Software Components view.

Before you begin

The Manage Software Components view shows all the software components in the active CMSIS solution. From the Manage Software Components view you can see all the component details (called attributes in the [CMSIS-Pack specification](#)).

Procedure

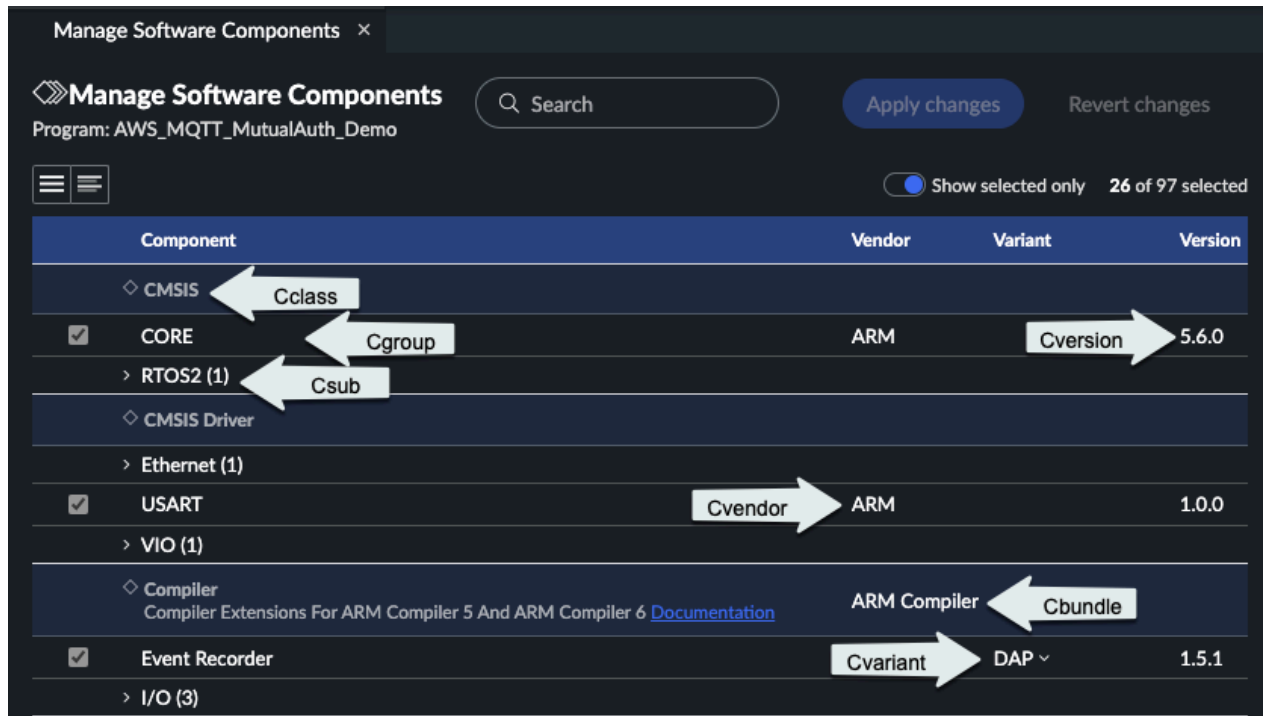
1. Set the solution you want to work on as the active project.
2. Click the Open software components view icon  from the Explorer.

Results

The Manage Software Components view opens.

The Show selected only toggle button is on by default. Only the components included in the active project display. If you switch off the toggle, you can see all the components that are available for use.

Figure 7-4: The ‘Manage Software Components’ view showing the components included in the active project



The first column in the Manage Software Components view shows how many instances of the component are deployed in the project. If the component has a selected checkbox, one instance

is deployed. If the checkbox is not selected, it is not deployed in the project. If a number input displays, multiple instances of this component can be deployed to a project.

The CMSIS-Pack specification states that each software component should have the following attributes:

- Component class (Cclass): A top-level component name. For example: CMSIS. For a full list of component classes, see [Software Component Cclasses](#).
- Component group (Cgroup): A component group name. For example: CORE for the CMSIS component class.
- Component version (Cversion): The version number of the software component.

Optionally, a software component might have these additional attributes:

- Component subgroup (Csub): A component subgroup that is used when multiple compatible implementations of a component are available. For example: Keil RTX5 implements CMSIS > RTOS2.
- Component variant (Cvariant): A variant of the software component is typically used when the same implementation has multiple top-level configurations, like Source for RTOS2 > Keil RTX5.
- Component vendor (Cvendor): The supplier of the software component. For example: Keil.
- Bundle (Cbundle): Allows you to combine multiple software components into a software bundle. Bundles have a different set of components available. All the components in a bundle are compatible with each other but not with the components of another bundle. For example: ARM Compiler for the Compiler component class.

Figure 7-5: Documentation links in the 'Manage Software Components' view

The screenshot displays the 'Manage Software Components' interface. At the top, there is a search bar and buttons for 'Apply changes' and 'Revert changes'. Below the search bar, a table lists components with columns for Component, Vendor, Variant, and Version. The 'CORE' component is expanded to show 'RTOS2 (1)' with a sub-entry for 'FreeRTOS' which includes a 'Documentation' link. Other components like 'CMSIS Driver', 'Ethernet (1)', 'USART', and 'VIO (1)' are also visible.

Component	Vendor	Variant	Version
◇ CMSIS Cortex Microcontroller Software Interface Components Documentation			
☑ CORE	ARM		5.6.0
▽ RTOS2 (1)			
☑ FreeRTOS CMSIS-RTOS2 implementation for Cortex-M based on FreeRTOS Documentation	ARM	Cortex-M	10.4.6
◇ CMSIS Driver			
> Ethernet (1)			
☑ USART	ARM		1.0.0
> VIO (1)			

Documentation links are available for most component classes, component groups and subgroups when you click their names.

With the Search field, you can do a search on the component groups and subgroups.

7.5.3 Modify the software components in your project

This topic describes how to select software components.

Before you begin

For solutions, you can add components from all the packs available (so it is not limited to the packs that are already selected for a given project).

Procedure

1. Switch the Show selected only toggle button off to see all the components available.
2. Select or clear components as needed using the checkboxes. If a number input displays instead of a checkbox, it means multiple instances of this component can be deployed to a project. For some components, you can also select a vendor, variant or version.

The Revert changes button allows you to revert all your unsaved changes at once.

3. Click Apply changes.

8. Work with Mbed projects

Keil Studio supports Mbed OS 5.12 and newer, and Mbed OS 6 and newer. Keil Studio also provides limited support for Mbed 2: you can build and run Mbed 2 projects, but you cannot do debugging. Arm recommends that you upgrade to Mbed OS 5 or 6 to benefit from all the features. See [Upgrade from Mbed 2 to Mbed OS 5 or 6](#) for more details.

8.1 Create, import or clone an Mbed project

This section describes how to start working with Mbed projects in Keil Studio.

You can:

- Create a new project from an example shipped with Keil Studio (examples are available for Mbed OS 5, Mbed OS 6, and Mbed 2).
- Create a new project as an empty Mbed project (with Mbed OS 5 and Mbed OS 6).
- Create a blank Mbed project (start from an empty folder with a manually selected version of Mbed OS).

You can also:

- Import a project from your local file system.
- Import projects from your Mbed Online Compiler workspace.
- Clone an existing project from os.mbed.com/code (as a URL) or a Git hosting service (such as GitHub).




If you do not already have an Arm account, [create one](#) and log into Keil Studio with it. Alternatively, you can use your Mbed account.

8.1.1 Create a project from an Mbed example project or an empty Mbed project

You can create an Mbed project from an example or an empty project.

Procedure

1. Create a new project, either:
 - In the Explorer view, click the Active project drop-down list then click the New project button  .
 - Select File > New... > Mbed Project.

The New Project dialog box opens.

2. Click the Example project drop-down list and select an example from the list. Each example has a readme.md file that explains the details of that example.

To create an Mbed project with no content (other than Mbed OS), select empty Mbed OS project under Mbed OS 6 or Mbed OS 5.

3. In the Project name field, edit the name of the new project or keep the name provided by default.
4. Check the default options:
 - Keil Studio sets the newly created project as the active project. Build and run commands only apply to the active project. Clear the checkbox if you do not want to make the new project active.
 - Keil Studio initializes the project as a Git repository. Clear the checkbox if you do not want to turn your project into a Git repository. See [Configure a project for source control and collaboration](#) for more details.
5. Click Add project. Keil Studio creates the project in your workspace.

8.1.2 Create a blank Mbed project

You can create a blank project that has no Mbed OS and no standard files, such as `main.cpp`. You must manually add a version of Mbed OS to a blank project.

Procedure

1. To create an empty folder that is not associated with an existing project, right-click an empty area in the file list and select New folder.
An empty folder is created.
2. You must add a copy of Mbed OS to make your folder a project.
 - a. Right-click the folder name and select Add Mbed Library...
 - b. In the URL field, enter `git@github.com:ARMmbed/mbed-os.git` and click Next.
 - c. From the drop-down list, select the branch or tag of Mbed OS you want to use. You can use any supported version, including patch versions.

Note that release tags are listed on the [Mbed OS releases page](#).

- d. Click Finish. The selected branch or tag is added to the folder, and you can now use it as a project.

8.1.3 Import an Mbed OS example from your file system

You might already have Mbed projects on your file system. For example, you might have downloaded an Mbed project from os.mbed.com/code or another online location.

Procedure

- To import a project from your file system, drag and drop your project folder to the Explorer view.


8.1.4 Import Mbed projects from your Mbed Online Compiler workspace

You can import Mbed projects stored in your Mbed Online Compiler workspace from Keil Studio. You can either import all your projects at once, or import them one by one. The import process creates copies of your projects.

Procedure

1. If your Keil Studio workspace is empty, click the Import projects from Mbed Online Compiler button from the Explorer view.
Alternatively, if you already have projects in your workspace, go to File > Import from Mbed Online Compiler....

The Copy Programs from Mbed Online Compiler dialog box opens.

2. Select the first checkbox to import copies of all your projects, or select one or more projects to copy individually.
3. Click the Copy programs button.
A progress bar indicates the progress of the import. A message and status icon display for each project. If a copy fails, hover over the  icon to get more details on the import.
4. Click the Finish button and check your imported projects in the Explorer view.
If you have imported Mbed 2 projects, Arm recommends that you upgrade to Mbed OS 5 or 6 to benefit from all the features. See [Upgrade from Mbed 2 to Mbed OS 5 or 6](#) for more details.

8.1.5 Clone an Mbed project

You can clone existing Mbed projects from os.mbed.com/code (as a URL) or a Git hosting service (such as GitHub).

Before you begin

For projects on a Git hosting service, you can clone public repositories without setting up your Git credentials. If you want to clone a private repository, [set up Git credentials](#). For Mbed projects on Mercurial, Mercurial uses your Arm account or Mbed account credentials, so you have access to the same Mercurial repositories as on os.mbed.com.

For more information about what cloning a repository means, see the [git-clone chapter](#) on git-scm.com.

Procedure

1. Open the File menu and select Clone....
The Clone dialog box displays.
2. Paste the full HTTPS or SSH URL of the relevant web page and (optionally) edit the project name.
This can be a page on a Git hosting service or a page listed on os.mbed.com/code.

Keil Studio sets the newly cloned project as the active project by default. Build and run commands only apply to the active project. Clear the checkbox if you do not want to make the new project active.

3. Click Add project. Keil Studio clones the project with the version of Mbed OS it was originally created with.
You can push changes back to the remote repository from Keil Studio. For more information, see [Source control](#).

8.1.6 Next steps

After importing or cloning an Mbed project, you must check whether the project has all the required libraries and default configuration files. If you import or clone an Mbed project that has an unsupported version of Mbed OS, you must add a supported version of the `mbed-os` library to be able to use the project in Keil Studio.

See [Manage an Mbed project and its libraries](#) for more information.

Imported or cloned Mbed projects have an `.mbed` file storing their settings, such as a default build target (target hardware) and toolchain. The build target you select in the UI overrides the build target set in the `.mbed` file.

8.1.7 Further resources about Mbed OS

You are now ready to work on your project.

You can find out more about [Mbed OS APIs and programming on the Mbed OS documentation site](#).

When you are ready, [build and run your project](#).

8.2 Manage an Mbed project and its libraries

This section describes how to manage Mbed projects and their libraries.

When you create a new project, or import or clone an existing project, you also import the libraries on which it is dependent, including Mbed OS which is delivered as `mbed-os`.

- When you create an example Mbed project, including an empty example, Keil Studio adds the version of Mbed OS that you selected when creating the project.

- When you import or clone an existing project, Keil Studio adds the version of Mbed OS stated in the `mbed-os.lib` file of the imported project.

Your project might require extra libraries, for example libraries that support hardware you have added to your development board. If you must add extra libraries, see [Import an Mbed library](#).



The Mbed OS bare metal profile is not a separate library; the Mbed OS bare metal profile is the full Mbed OS library, which is stripped down at build time. For more information, see the [Mbed OS bare metal profile](#) page.



Keil Studio provides limited support for Mbed 2. You cannot modify the Mbed 2 library of a project (`mbed.lib` file) from the Mbed Libraries view. You can manually change the version of Mbed in the `mbed.lib` file. See [Change the Mbed version for your project](#). However, Arm recommends that you upgrade to Mbed OS 5 or 6 to benefit from all the features. See [Upgrade from Mbed 2 to Mbed OS 5 or 6](#) for more details.

8.2.1 Mbed OS 5 or 6 project

This section describes how to manage the Mbed libraries in your Mbed OS 5 or 6 project.

8.2.1.1 Open the Mbed Libraries view

The Mbed Libraries view lists all the libraries in an active project (indicated with the cogwheel icon on the file list). From the Mbed Libraries view, you can see library details, such as the library version and where it was imported from. You can also import, remove, update a library, or check out a specific library version.

Procedure

1. Right-click the project name in the Explorer view and select Set Active Project.
2. Select Mbed Libraries from the View menu.
The Mbed Libraries view opens in the bottom panel. When a library depends on another library, the Mbed Libraries view presents the dependency indented under the dependent library.

8.2.1.2 Import an Mbed library

This section describes how to import Mbed libraries to an existing project.

Before you begin

The project that you want to add a new Mbed library to must be set to the active project. To make the project active, right-click the project name in the Explorer view and select Set Active Project.

About this task


You can import Mbed libraries to an existing project from os.mbed.com or from a Git hosting service. Each library that you add to a project can be identified by a `.lib` file.



The Keil Studio Explorer view hides `.lib` files by default. Go to File > Settings > Open Preferences and search for the Exclude preference. Open the `settings.json` file and set `"files.exclude"` to `false` for `"**/*.lib"` files.

Procedure

1. Either:

- Open the Mbed Libraries view, select View > Mbed libraries. In the Mbed Libraries view, click the Add new library button .
- In the Explorer view, right-click the project in which you want to add your new Mbed library and select Add Mbed Library....

The Add Mbed Library dialog box opens.

2. Choose whether you want to import the library from os.mbed.com or from a Git hosting service and paste the full remote URL.
3. Modify the library name as required, and click Next.
4. Specify the release, branch or tag you want to import.

Notes:

- You must be connected to GitHub to see the list of Mbed OS releases available. See [Set credentials for GitHub](#).
- When a branch is checked out, the library checks out to the tip of the selected branch.
- When a tag is checked out, the library is put in a detached head state.

5. Click Finish.

You can view the progress of the import on the Background Tasks progress indicator, which appears in the bottom-right corner of the screen. When the import is complete, the library appears in the Mbed Libraries view and in the Explorer view.


8.2.1.3 Remove an Mbed library

This section describes how to remove an Mbed library from your active project in Keil Studio.

Before you begin

- Ensure that your project contains one or more Mbed libraries. To see how to import an Mbed library, see [Import an Mbed library](#).
- The project containing the Mbed library (or libraries) that you want to remove must be set to the active project. To make the project active, right-click the project name in the Explorer view and select Set Active Project.

Procedure

1. Click the Remove library button , which appears on the right when you hover the mouse pointer over the name of the library in the Mbed Libraries view. The Remove library dialog box displays.
2. Click Remove.

8.2.1.4 Update and change Mbed libraries

This section describes how to update and change Mbed libraries in Keil Studio.

When you import a library to Keil Studio either manually, or as a dependency for an imported project, you clone the library to your workspace.

Libraries are associated with specific projects in your workspace. If you have multiple projects that use a particular library, you have multiple copies of the same library; one per project. Each copy of the library can be at a different commit, if you update a library for one project, it does not update other copies of that library.

In Keil Studio, you can easily update all Mbed libraries at once, update a single library to the latest version, or update a single library to a specific version (as described in the following topics).

8.2.1.4.1 Update all Mbed libraries

This section describes how to update all of the top-level Mbed libraries in the active project.


Before you begin

- Ensure that your project already contains one or more Mbed libraries. To see how to import an Mbed library, see [Import an Mbed library](#).
- The project containing the Mbed libraries that you want to update must be set to the active project. To make the project active, right-click the project name in the Explorer view and select Set Active Project.

About this task

When there are updates to libraries in an active project, the Update all button in the top right-hand corner of the Mbed Libraries view indicates the number of updates available. Click the Update all button to view details about the available updates.

Procedure

1. Open the Mbed Libraries view, select View > Mbed libraries.
2. In the top right-hand corner of the Mbed Libraries view, click the Update all top-level libraries button . The Update all libraries dialog box opens with details about all the libraries for which updates are available.
3. Click Update all.

Note: Updating all top-level libraries does not automatically update all library dependencies.


8.2.1.4.2 Update an Mbed library to the latest version

This section describes how to update an Mbed library to the latest version.

Before you begin

- Ensure that your project already contains one or more Mbed libraries. To see how to import an Mbed library, see [Import an Mbed library](#).
- The project containing the Mbed library that you want to update must be set to the active project. To make the project active, right-click the project name in the Explorer view and select Set Active Project.

Procedure

1. Open the Mbed Libraries view (select View > Mbed Libraries).
2. To the right of the name of the library, click the Update to latest button .

8.2.1.4.3 Update an Mbed library to a specific version

This section describes how to update to an Mbed library to a specific version.

Before you begin

- Ensure that your project already contains one or more Mbed libraries. To see how to import an Mbed library, see [Import an Mbed library](#).
- The project containing the Mbed library that you want to update must be set to the active project. To make the project active, right-click the project name in the Explorer view and select Set Active Project.

Procedure

1. Either:
 - Open the Mbed Libraries view, select View > Mbed Libraries. Next to the library name is the Update to latest button and a chevron. Click the chevron, and select Change Library Version.
 - In the Explorer view, right-click the Mbed OS library that you want to update, and select Change Library Version.

A dialog opens with the title Update <selected-library> to Version.

2. In the dialog, click the drop down menu and select the version of the library to check out. The list is divided into RELEASES, BRANCHES and TAGS.
Notes:

- You must be connected to GitHub to see the list of Mbed OS releases available. See [Set credentials for GitHub](#).
- When a branch is checked out, the library checks out to the tip of the selected branch.

- When a tag is checked out, the library is put in a detached head state.
3. Click Update library.
The selected version of the library starts to check out. You can monitor the progress of the checkout in the Mbed Libraries view.

When the checkout is complete the `.lib` file is updated.

8.2.1.5 Fix library problems

This section describes how to fix library problems in Keil Studio.


About this task

When there are problems with any of the libraries in an active project, the Fix all problems button in the top right-hand corner of the Mbed Libraries view indicates the number of fixes required. For example, if you do not have the source of a library downloaded, or if you check out a new commit of a project that refers to a library commit you do not have checked out.

Click the Fix all problems button to view details about the required fixes.

You can fix all the libraries in the active project in one click, or you can fix each library individually.

Procedure

1. Click the Fix all problems button  in the top right-hand corner of the Mbed Libraries view. The Fix all problems dialog box opens with details about all the required fixes.
2. Click Fix all.
To fix a specific library, click the Fix problems button to the right of the library name in the Mbed Libraries view.

8.2.1.6 Source-control library updates

`.lib` files describe the libraries in your project. To share library changes with collaborators, commit `.lib` file changes to the source control repository of your project. Do not add the library directory and its contents to your project repository.



The `.lib` file only identifies the commit that you are using; the file does not include branch information. In Keil Studio, you can check out a library to the tip of a specific branch, but this information is not included if you open your project repository in another application, or if you push changes to a remote repository.

8.2.2 Mbed 2 project

This section describes how to manually change the version of Mbed for your Mbed 2 project. Also describes how to upgrade to Mbed OS 5 or 6 (recommended approach).

8.2.2.1 Change the Mbed version for your project

Any Mbed 2 project is shipped with an `mbed.b1d` file which contains a link to the Mbed 2 library version used for the project. You cannot modify the Mbed 2 library version from the Mbed Libraries view. Instead, you must manually update the version of Mbed in the `mbed.b1d` file.

Procedure

1. Set the project as the active project.
2. Open the `mbed.b1d` file from the Explorer view. The file contains a link to the Mbed 2 library.
3. Delete the link.
4. Go to os.mbed.com and find the Mbed 2 version you need.
5. Copy the URL and paste it in the `mbed.b1d` file.
6. Select Find Command... from the View menu or press Ctrl+Shift+P (Windows) or Cmd+Shift+P (macOS).
7. Look for the Refresh Mbed 2 SDK command and select it.
Keil Studio synchronizes the `mbed.b1d` file and the Mbed 2 source files contained in the `mbed` folder in your project.

8.2.2.2 Upgrade from Mbed 2 to Mbed OS 5 or 6

Arm recommends that you upgrade your Mbed 2 projects to Mbed OS 5 or 6 to benefit from all the features.

About this task

The main steps are as follows:

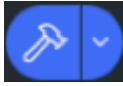
1. Mbed 2 projects contain an `mbed.b1d` file or an `mbed-rtos.lib` file or both. Mbed OS 5 projects or later have an `mbed-os.lib` file. Once you have created or imported an Mbed 2 project, delete the `mbed.b1d` and `mbed-rtos.lib` files from your project and add the correct `mbed-os.lib` file.
2. Check for compilation errors and try to correct your code to fix errors.

See the upgrade example below. In this example, we use the Mbed 2 Blinky example project that is available in Keil Studio.

:::Note Optionally, you can enable the bare metal profile for your project to use Mbed without an RTOS. See [Using the bare metal profile](#) for more information. :::

Procedure

1. Select File > New... > Mbed Project.
The New Project dialog box opens.
2. Click the Example project drop-down list, then look for the Mbed 2 `mbed2-example-blinky` example and select it.
3. Click Add project. Keil Studio creates the project in your workspace.
4. Open the project from the Explorer view.

5. Right-click the `mbed.b1d` file and select Delete.
Note: If you cannot see the `mbed.b1d` file, check that files with this extension are not hidden by default. Go to File > Settings > Open Preferences and search for the Exclude preference. Open the `settings.json` file and check the visibility for `"/**/*.*.b1d"` files. The `"files.exclude"` value must be set to `false`.
6. Right-click the name of your project and select Add Mbed Library to add the `mbed-os.lib` file. The Add Mbed Library dialog box opens.
7. In the URL field, paste the following URL: <https://github.com/ARMmbed/mbed-os> (this is the GitHub URL where you can find the Mbed OS source code).
8. Keep `mbed-os` in the Library Name field and click Next.
9. Select the latest Mbed OS version and click Finish.
Keil Studio imports the `mbed-os.lib` library.
10. Check that a build target (or target hardware) is selected in the Target hardware drop-down list.
11. 
Click the Build project (hammer) button to build the project.
Two errors display in the Output view: `use of undeclared identifier 'wait'`.
12. In the code, replace `wait` by `wait_ns` and `(0.2)` by `(1000)`. `wait` has been deprecated in later versions of Mbed OS.
13. Build the project again. The project now builds successfully.

9. Build and run a project

This chapter describes how to build and run a project with Keil Studio.

9.1 Select a build target

This section describes how to select a build target (or target hardware) for your project in Keil Studio. A build target tells Keil Studio how to build the RTOS so that it matches your hardware.

Procedure

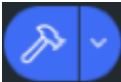
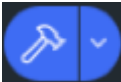
- To select a build target, either:
 - Connect your board to your computer. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target hardware.
 - Use the Target hardware drop-down list. Your target hardware name most likely matches the board name.

If you have an unsupported board, see the [Custom targets page](#).

9.2 Build and flash a project to the board

This section describes how to build and flash a project.

Procedure

1. Ensure the project you want to build and flash is set to active.
To make the project active, right-click the project name in the Explorer view and select Set Active Project.
2. Select one of these options:
 - Build project: Build the project, without flashing it to the board.
 - : Build project (hammer) : Checks the `BUILD` subdirectory (where the previous build is stored) and optimizes build time by using as much as it can from the last build. For example, if two consecutive builds are for the same target hardware, the build rebuilds only the files that actively changed between the two builds, or from the point at which you stopped the previous build attempt.
 - Clean build (click the drop-down next to the hammer): Ignore all previous builds. A clean build is slower than reusing a previous build, but guarantees that your build has no old artifacts.

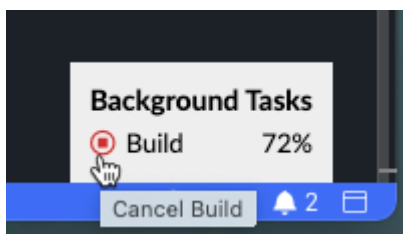
- Run project : Build the project and flash it to the board.

Note that there is only one default build profile.

For Mbed projects only: Select the Use Daplink option in File > Settings > Open Preferences > Run category to build and flash projects to your board. When this option is not selected, the Run project button is not available.

You can stop a build at any time with the Cancel Build button in the Background Tasks progress indicator.


Figure 9-1: Cancel a build



9.3 Output view

The output of a build displays in the Output view. This topic describes how you can copy, search the contents of the Output view, look for errors and warnings, and clear the Output view.

To open the Output view, select View > Output.

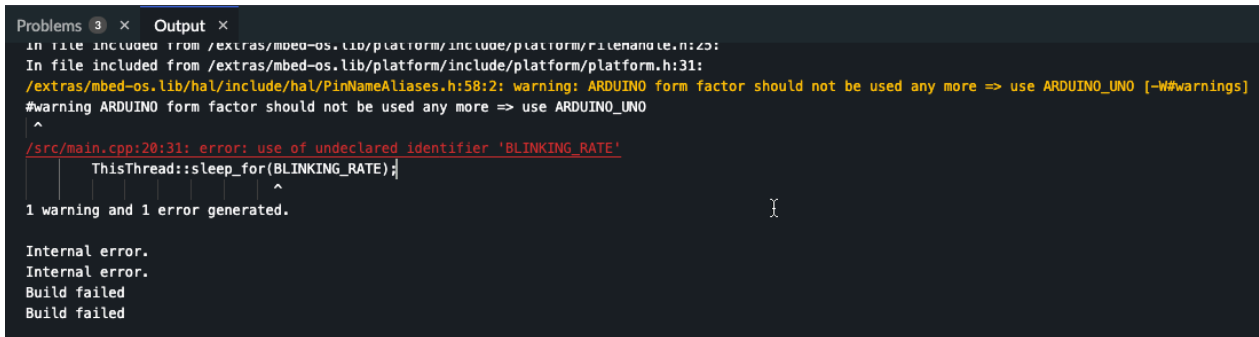
To turn auto scrolling on, click the Turn Auto Scrolling On  icon. To turn auto scrolling off, click the Turn Auto Scrolling Off icon. When the auto scrolling is off, you can read the output or copy content from the output while the build is running and new output content is logged.

To copy the output, right-click in the Output view and select Copy or Copy All:

- Copy: Copies the line where your cursor is.
- Copy All: Copies the whole output.


To search the content of the output, right-click in the Output view and select Find. When the Output view is in focus, you can also go to the Edit menu and select Find.

Errors display in red and warnings display in yellow. If a link is available, use Cmd + mouse click to follow the error or warning to the relevant part of the code:

Figure 9-2: The error text is a link to the relevant line of code

```
Problems 3 x Output x
In file included from /extras/mbed-os.lib/platform/include/platform/Platform.h:25:
In file included from /extras/mbed-os.lib/platform/include/platform/platform.h:31:
/extras/mbed-os.lib/hal/include/hal/PinNameAliases.h:58:2: warning: ARDUINO form factor should not be used any more => use ARDUINO_UNO [-W#warnings]
#warning ARDUINO form factor should not be used any more => use ARDUINO_UNO
^
/src/main.cpp:20:31: error: use of undeclared identifier 'BLINKING_RATE'
    ThisThread::sleep_for(BLINKING_RATE);
                          ^
1 warning and 1 error generated.

Internal error.
Internal error.
Build failed
Build failed
```

To clear the build output, click the Clear Output icon  or right-click and select Clear Output.

9.4 Configuring compile-time customizations

This section describes how to configure compile-time customizations in Keil studio.

For Mbed projects only: The Arm Mbed OS configuration system, a part of the Arm Mbed OS build tools that underpin Keil Studio, customizes compile-time configuration parameters. Each library might define various configuration parameters in its `mbed_lib.json`. The project-level `mbed_app.json` might override the values of these configuration parameters. At compile-time, the configuration system gathers and interprets the configurations defined in all `mbed_lib.json` files and the `mbed_app.json` file, and creates a single header file, `mbed_config.h`. In `mbed_config.h`, the defined configuration parameters are converted into C preprocessor macros.

Some examples of configuration parameters:

- The sampling period for a data acquisition application.
- The default stack size for a newly created OS thread.
- The receive buffer size of a serial communication library.
- The flash and RAM memory size of an Mbed target.
- Bare metal profile and small C-library use.

To use the configuration system, [see the Mbed OS documentation about the configuration system](#).

10. IntelliSense

In this chapter, you will learn how to quickly navigate your code, how to use the IntelliSense code editing features to update and improve your code, and how to enable and use linting.

10.1 Code editing

Keil Studio provides C and C++ language support using the clangd language server and the Language Server Protocol (LSP). Keil Studio includes various navigation features and IntelliSense code editing features to help you code faster and more efficiently.

The IntelliSense code editing features are available for standalone files or for multiple files, for all your projects.



You must set a build target (target hardware) to get all IntelliSense features.

10.1.1 Navigate your code

This section describes how to quickly navigate your code, and how to use the highlighting, in-tool pop up information, and focused-searching functionality in Keil Studio.

Quick file navigation

This section describes how to quickly navigate your files using keyboard shortcuts in Keil Studio.

To follow a link included in your code or find a file where a code element has been defined, hold Cmd (macOS) or Ctrl (Windows and Linux) and click the link or the code element. The file opens in a new tab and is directly accessible from the Explorer view.

Bracket and quote highlighting

By default, the editor highlights the matching brackets or quotes when your cursor is over them. When you type the left bracket or quote, the editor automatically inserts the matching right bracket or quote.

Source hover

Additional information about the code elements you are hovering over in your code displays in a popup.

Figure 10-1: Additional information available when you hover over the code

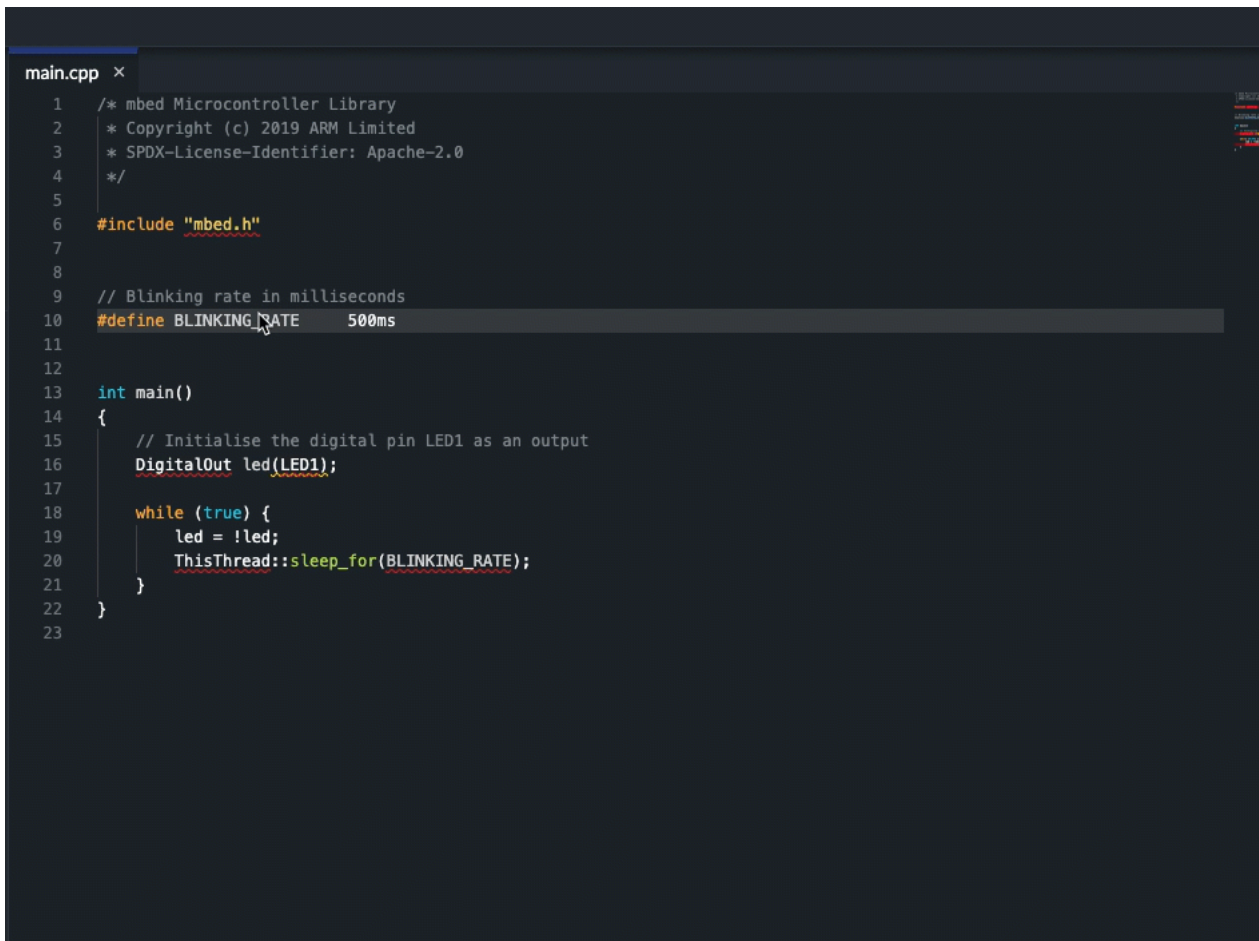
```
5
6 #include "PinNames.h"
7 #include "mbed.h"
8
9
10 // Blinking rate in milliseconds
11 #define BLINKING_RATE 500ms
12
13
14 int main()
15 {
16     // Initialise the digital pin LED1 as an output
17     DigitalOut led(LED1);
18
19     while (true) {
20         led = !led;
21         ThisThread::sleep_for(BLINKING_RATE);
22     }
23 }
24
```

Go to Declaration and Peek Declaration

To navigate to a declaration, use the Go to Declaration and Peek Declaration options.

Hover over a code element, right-click, and select one of these options:

- Go to Declaration: Opens the header file where the declaration is defined in a separate tab.
- Peek > Peek Declaration: Takes you to the declaration but the declaration is presented inline.

Figure 10-2: Declaration presented inline with the Peek Declaration option

```
main.cpp x
1  /* mbed Microcontroller Library
2   * Copyright (c) 2019 ARM Limited
3   * SPDX-License-Identifier: Apache-2.0
4   */
5
6  #include "mbed.h"
7
8
9  // Blinking rate in milliseconds
10 #define BLINKING_RATE 500ms
11
12
13 int main()
14 {
15     // Initialise the digital pin LED1 as an output
16     DigitalOut led(LED1);
17
18     while (true) {
19         led = !led;
20         ThisThread::sleep_for(BLINKING_RATE);
21     }
22 }
23
```

Go to Definition and Peek Definition

You can navigate to the definition of a code element with the Go to Definition and Peek Definition options.

Hover over a code element, right-click, and select one of these options:

- Go to Definition: Opens the file where the element is defined in a separate tab.
- Peek > Peek Definition: Takes you to the definition of an element but the definition is presented inline.

Figure 10-3: Definition presented inline with the Peek Definition option

```
main.cpp x
1  /* mbed Microcontroller Library
2     * Copyright (c) 2019 ARM Limited
3     * SPDX-License-Identifier: Apache-2.0
4     */
5
6  #include "mbed.h"
7
8
9  // Blinking rate in milliseconds
10 #define BLINKING_RATE    500ms
11
12
13 int main()
14 {
15     // Initialise the digital pin LED1 as an output
16     DigitalOut led(LED1);
17
18     while (true) {
19         led = !led;
20         ThisThread::sleep_for(BLINKING_RATE);
21     }
22 }
23
```

Go to Implementations and Peek Implementations

You can navigate to the implementations of an abstract class with the Go to Implementations and Peek Implementations options.

The Go to Implementations and Peek Implementations options show the implementations of an abstract class inline (or, if there is only one implementation, the Go to Implementations option takes you directly to it).

Figure 10-4: Implementations of an abstract class presented inline with the Peek Implementations option

```

iotc_event_dispatcher_test.cc x
1  /* Copyright 2018–2020 Google LLC
2  *
3  * This is part of the Google Cloud IoT Device SDK for Embedded C.
4  * It is licensed under the BSD 3-Clause license; you may not use this file
5  * except in compliance with the License.
6  *
7  * You may obtain a copy of the License at:
8  * https://opensource.org/licenses/BSD-3-Clause
9  *
10 * Unless required by applicable law or agreed to in writing, software
11 * distributed under the License is distributed on an "AS IS" BASIS,
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 */
16 #include "gmock.h"
17 #include "gtest.h"
18
19 #include "iotc_event_dispatcher_api.h"
20 #include "iotc_event_handle.h"
21
22 namespace iotctest {
23 namespace {
24
25 using ::testing::Eq;
26 using ::testing::Return;
27
28 class FreeFunctions {
29 public:
30     virtual ~FreeFunctions() = default;
31     virtual iotc_state_t H0() const = 0;
32     virtual iotc_state_t H1(iotc_event_handle_arg1_t a1) const = 0;
33     virtual iotc_state_t H2(iotc_event_handle_arg1_t a1,
34                             iotc_event_handle_arg2_t a2) const = 0;
35     virtual iotc_state_t H3(iotc_event_handle_arg1_t a1,
36                             iotc_event_handle_arg2_t a2,
37                             iotc_event_handle_arg3_t a3) const = 0;
38     virtual iotc_state_t H4(iotc_event_handle_arg1_t a1,
39                             iotc_event_handle_arg2_t a2,
40                             iotc_event_handle_arg3_t a3,
41                             iotc_event_handle_arg4_t a4) const = 0;
42     virtual iotc_state_t H5(iotc_event_handle_arg1_t a1,
43                             iotc_event_handle_arg2_t a2,
44                             iotc_event_handle_arg3_t a3,
45                             iotc_event_handle_arg4_t a4,
46                             iotc_event_handle_arg5_t a5) const = 0;
47     virtual iotc_state_t H6(iotc_event_handle_arg1_t a1,
48                             iotc_event_handle_arg2_t a2,
49                             iotc_event_handle_arg3_t a3,
50                             iotc_event_handle_arg4_t a4,
51                             iotc_event_handle_arg5_t a5,
52                             iotc_event_handle_arg6_t a6) const = 0;
53 };
54

```

Go to References and Peek References

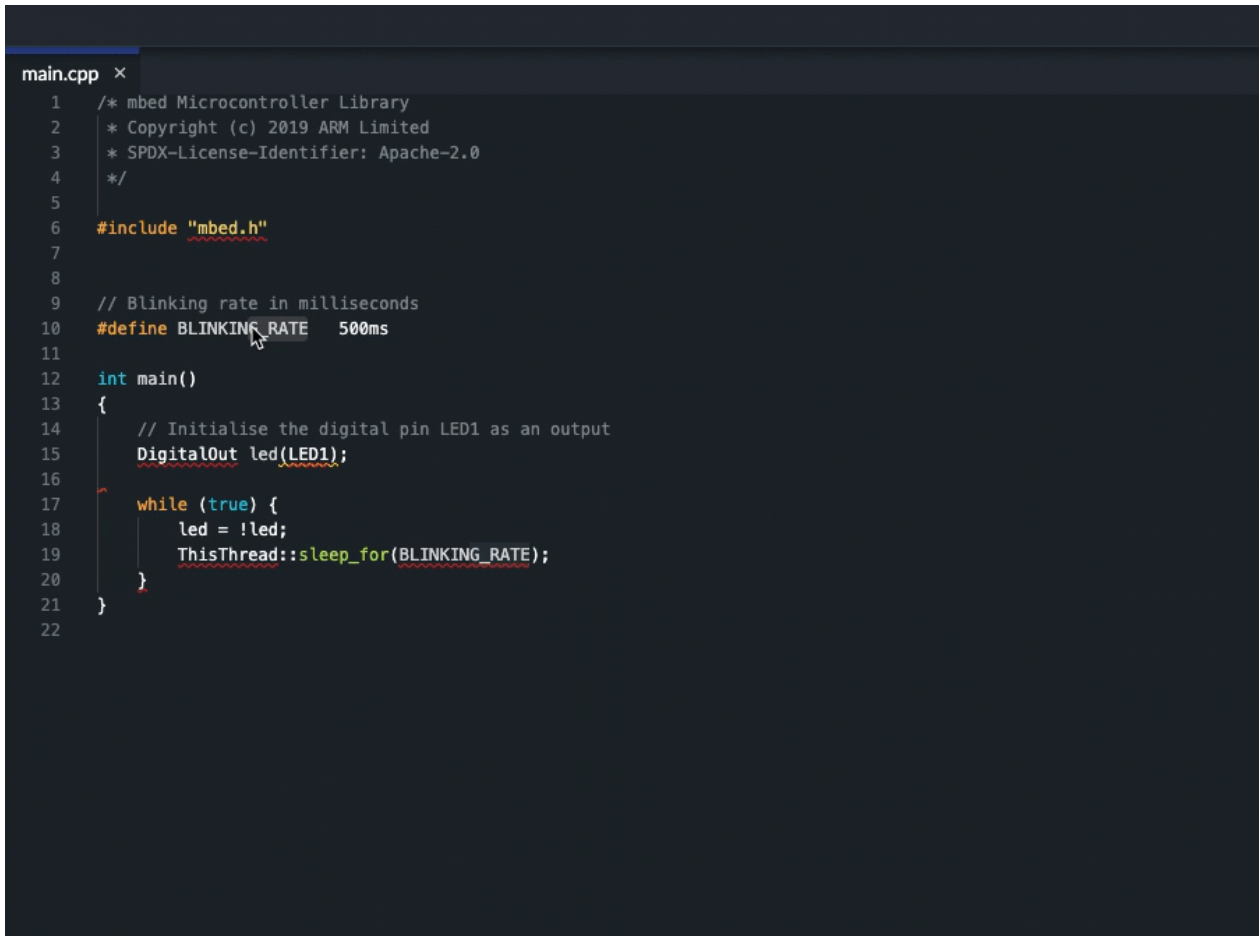
You can find the references to a code element throughout all the files of an active project with the Go to References and Peek References options.

The Go to References and Peek References options show the code element references inline (or, if there is only one reference, the Go to References option takes you directly to the element).

To navigate between references and make edits in the inline editor:

1. Double-click a reference to open the file or files where the code element appears.
2. Hover over the code element, right-click, and select one of the options.

Figure 10-5: Peek References option. You can edit directly in the inline editor



```
main.cpp x
1  /* mbed Microcontroller Library
2  * Copyright (c) 2019 ARM Limited
3  * SPDX-License-Identifier: Apache-2.0
4  */
5
6  #include "mbed.h"
7
8
9  // Blinking rate in milliseconds
10 #define BLINKING_RATE 500ms
11
12 int main()
13 {
14     // Initialise the digital pin LED1 as an output
15     DigitalOut led(LED1);
16
17     while (true) {
18         led = !led;
19         ThisThread::sleep_for(BLINKING_RATE);
20     }
21 }
22
```

Go to Symbol

This section describes how to use the Go to Symbol... option in Keil Studio to perform a symbol-focused search of your code.

To use the Go to Symbol... option:

1. Right-click anywhere in the editor and select Go to Symbol....

The search bar displays the available symbols in the file you are currently working on.

2. Select a symbol to go to that symbol.


Figure 10-6: Go to Symbol option

```
main.cpp x
104     if (ret != osOK) {
105         tr_error("Failed to start thread to process MQTT");
106         goto disconnect;
107     }
108
109     // Run a demo depending on the configuration in mbed_app.json:
110     // * demo_mqtt.cpp if aws-client.shadow is unset or false
111     // * demo_shadow.cpp if aws-client.shadow is true
112     demo();
113
114     disconnect:
115     if (client.isConnected()) {
116         connection_mutex.lock();
117         ret = client.disconnect();
118         connection_mutex.unlock();
119         if (ret != MBED_SUCCESS) {
120             tr_error("AWS::disconnect() failed");
121         }
122     }
123
124     // The process thread should finish on disconnection
125     ret = process_thread.join();
126     if (ret != osOK) {
127         tr_error("Failed to join thread");
128     }
129
130     ret = network->disconnect();
131     if (ret != MBED_SUCCESS) {
132         tr_error("NetworkInterface::disconnect() failed");
133     }
134
135     end:
136     tr_info("End of the demo");
137 }
```

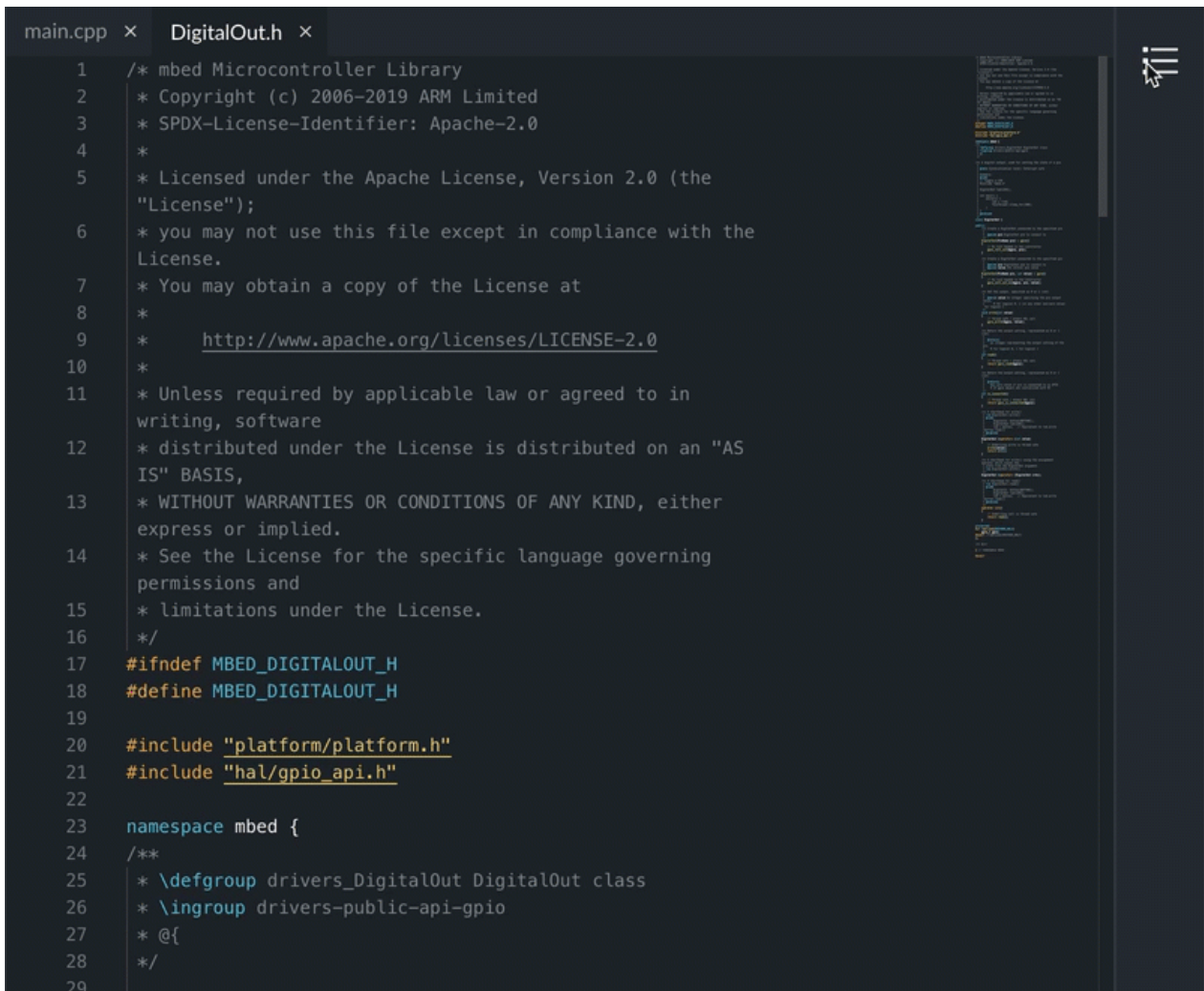
Open the Outline view

The Outline view gives you a tree view of the classes, variables, and functions in the current file.

To use the Outline view:

1. From the View menu, select Outline or click the Outline button  in the top right-hand corner of the screen.
2. To navigate through the file, click the elements in the Outline view.

In the Outline view, orange icons represent classes, white icons represent variables, and purple icons represent functions.

Figure 10-7: IntelliSense Outline view

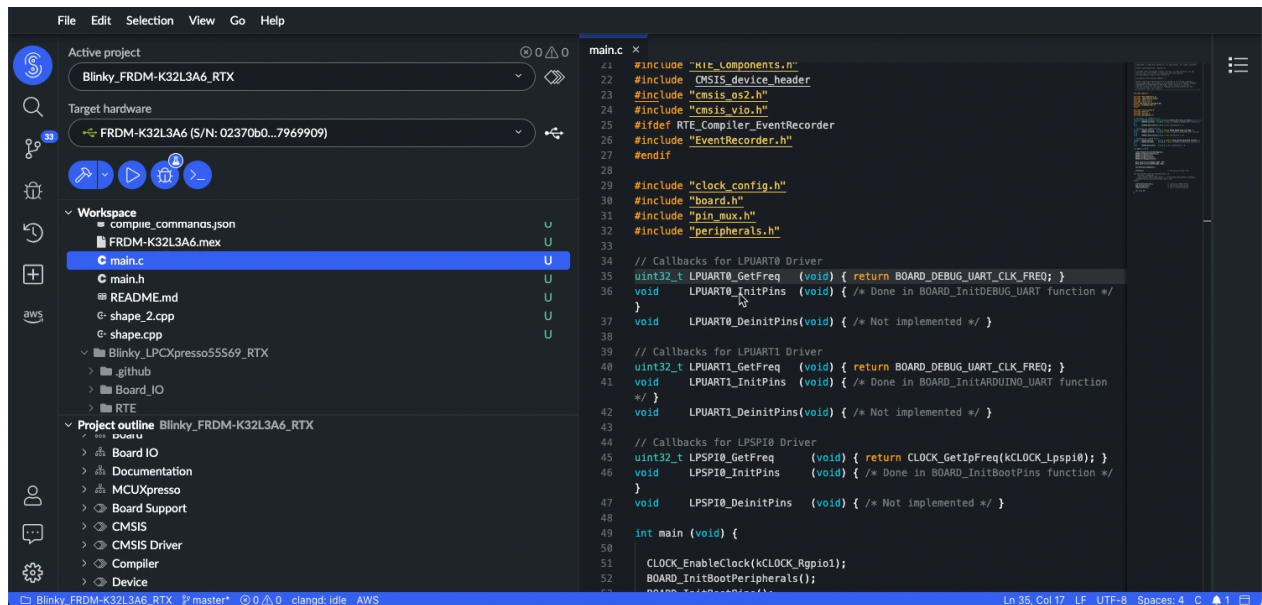
```
main.cpp × DigitalOut.h ×
1  /* mbed Microcontroller Library
2     * Copyright (c) 2006-2019 ARM Limited
3     * SPDX-License-Identifier: Apache-2.0
4     *
5     * Licensed under the Apache License, Version 2.0 (the
6     * "License");
7     * you may not use this file except in compliance with the
8     * License.
9     * You may obtain a copy of the License at
10    *
11    * http://www.apache.org/licenses/LICENSE-2.0
12    *
13    * Unless required by applicable law or agreed to in
14    * writing, software
15    * distributed under the License is distributed on an "AS
16    * IS" BASIS,
17    * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
18    * express or implied.
19    * See the License for the specific language governing
20    * permissions and
21    * limitations under the License.
22    */
23    #ifndef MBED_DIGITALOUT_H
24    #define MBED_DIGITALOUT_H
25
26    #include "platform/platform.h"
27    #include "hal/gpio_api.h"
28
29    namespace mbed {
30    /**
31     * \defgroup drivers_DigitalOut DigitalOut class
32     * \ingroup drivers-public-api-gpio
33     * @{
34     */
35    }
```

Show AST

You can open an abstract syntax tree with the Show AST option in the right-click menu.

The abstract syntax tree shows how the clangd language server parses the code.

Figure 10-8: Show AST option



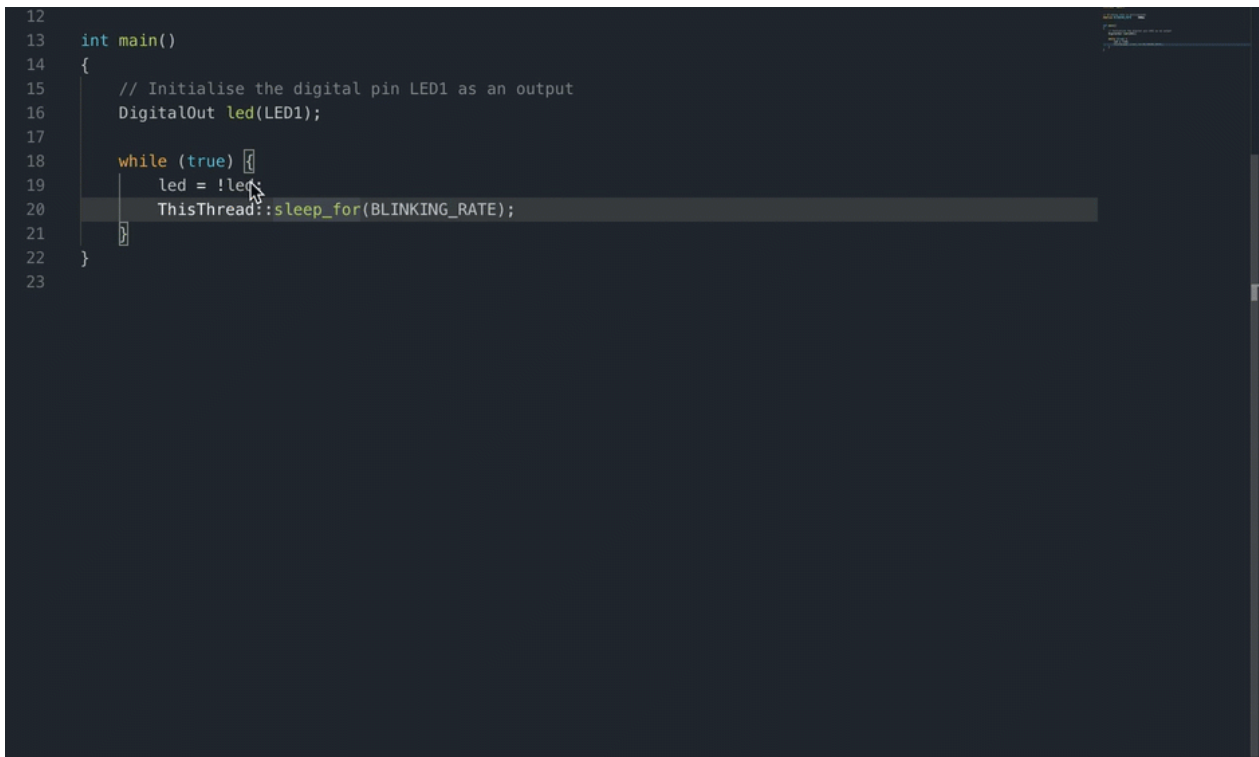
10.1.2 Edit and refactor your code

This section describes the auto-completion, renaming, format-correcting, and refactoring functionality that is available in Keil Studio, to help you edit your code.

Auto-completion and signature help

When you start typing a keyword, type, function, variable name, or other project element that Keil Studio recognizes, the editor offers to complete what you are typing. A drop-down list with possible options opens where you can select what you need. The list also shows an icon that represents the code entity of each option (for example namespace, function, class or variable), and signature help for parameters of functions. Clicking the “i” icon gives you more details on each option.

You can trigger auto-complete and help for existing code by pressing Ctrl+Space.

Figure 10-9: Auto-completion and i iconA screenshot of an IDE showing a C++ code file. The code is as follows:

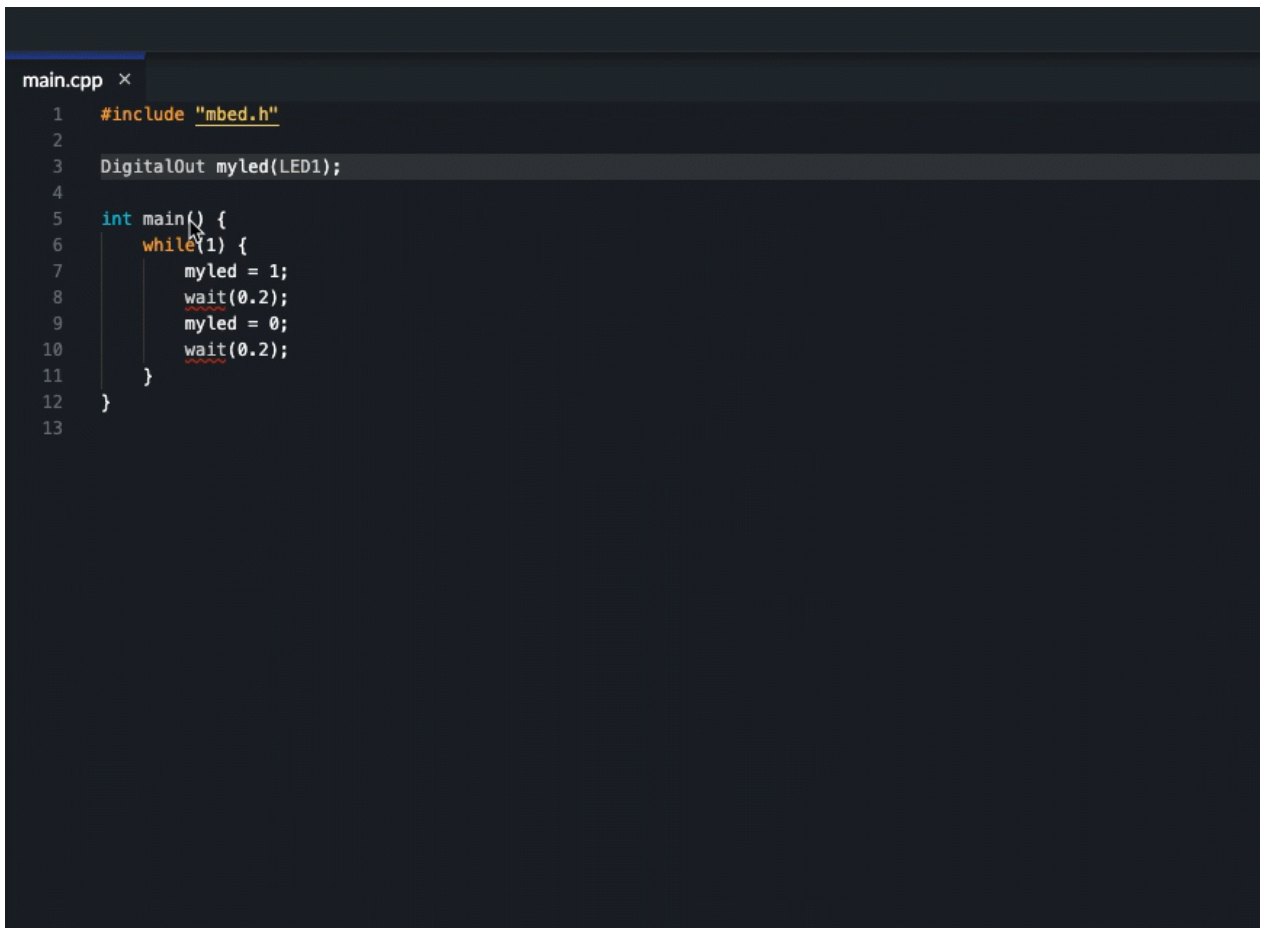
```
12
13 int main()
14 {
15     // Initialise the digital pin LED1 as an output
16     DigitalOut led(LED1);
17
18     while (true) {
19         led = !led;
20         ThisThread::sleep_for(BLINKING_RATE);
21     }
22 }
23
```

The cursor is positioned at the end of the line `led = !led;` on line 19. A dropdown menu is visible, showing the text `ThisThread::sleep_for(BLINKING_RATE);` on line 20. A mouse cursor is hovering over the `led` variable in the code, and a small 'i' icon is visible next to it, indicating an IntelliSense action.

Rename an element

If you want to rename an element in your code such as a class, a function, or a variable, use the Rename symbol option. This option renames all the occurrences of a symbol in the file you are currently working on.

To find and replace a string everywhere in the file you are currently working on, use the Change All Occurrences option.

Figure 10-10: Renaming a symbol

```
main.cpp ×
1  #include "mbed.h"
2
3  DigitalOut myled(LED1);
4
5  int main() {
6      while(1) {
7          myled = 1;
8          wait(0.2);
9          myled = 0;
10         wait(0.2);
11     }
12 }
13
```

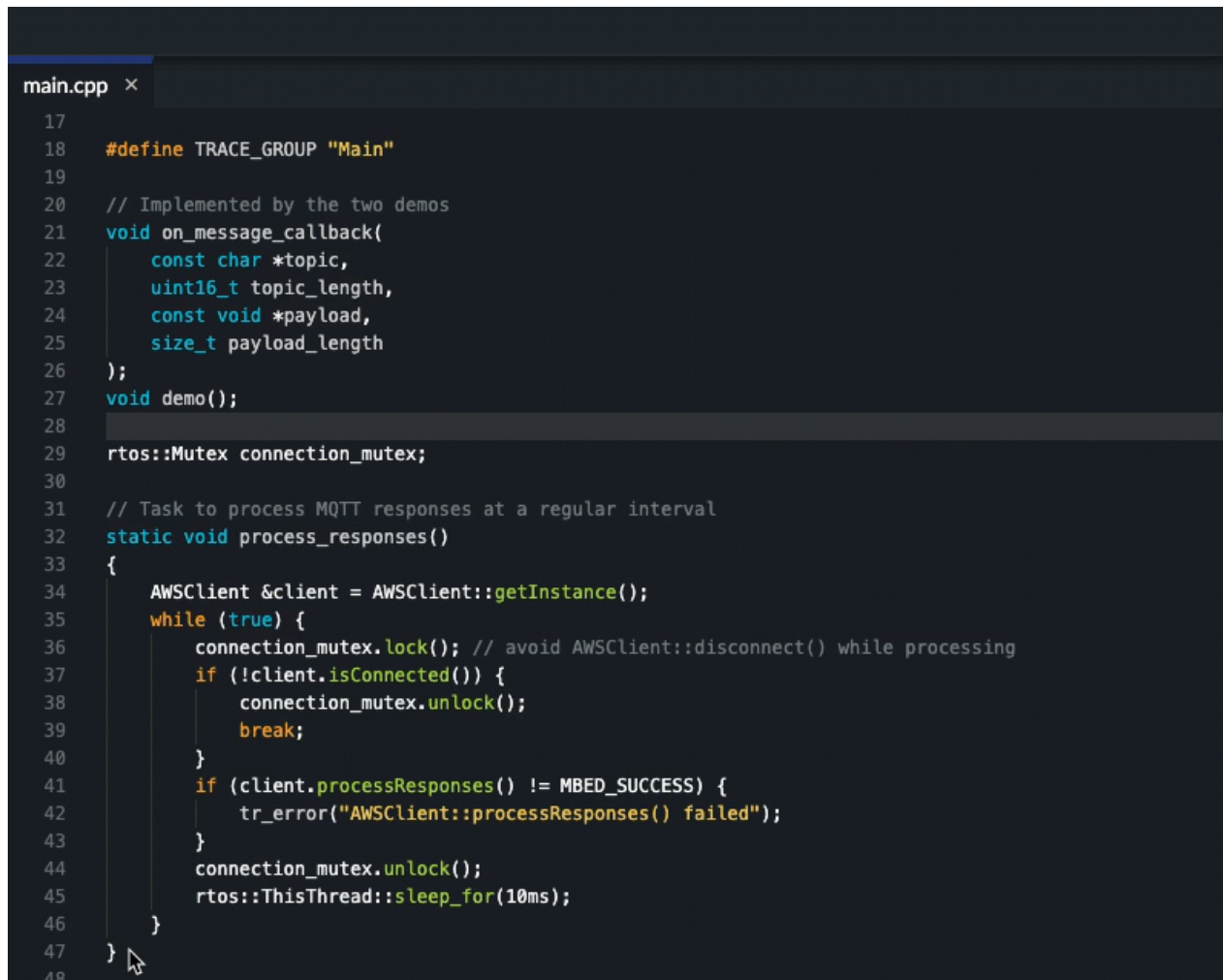
Code formatting

This topic describes how to change the indentation to format your code correctly.

The Format Document and Format Selection options allow you to indent your code correctly.

If you select:

- Format Document: the indentation corrects in the whole file you are currently working on.
- Format Selection: only the line, or lines, of code you selected indent correctly.

Figure 10-11: Changing the indentation with Format Selection

```
main.cpp x
17
18 #define TRACE_GROUP "Main"
19
20 // Implemented by the two demos
21 void on_message_callback(
22     const char *topic,
23     uint16_t topic_length,
24     const void *payload,
25     size_t payload_length
26 );
27 void demo();
28
29 rtos::Mutex connection_mutex;
30
31 // Task to process MQTT responses at a regular interval
32 static void process_responses()
33 {
34     AWSClient &client = AWSClient::getInstance();
35     while (true) {
36         connection_mutex.lock(); // avoid AWSClient::disconnect() while processing
37         if (!client.isConnected()) {
38             connection_mutex.unlock();
39             break;
40         }
41         if (client.processResponses() != MBED_SUCCESS) {
42             tr_error("AWSClient::processResponses() failed");
43         }
44         connection_mutex.unlock();
45         rtos::ThisThread::sleep_for(10ms);
46     }
47 }
48
```

To change the indentation:

1. Click the Spaces or Tab Size option on the right-hand side of the status bar.

A drop-down list opens at the top of the editor.

2. Select Indent Using Spaces or Indent Using Tabs in the drop-down list, then select the number of spaces and tabs to use for the indentation.

Fix-its, Peek Problem, and Quick Fix

Fix-its offer suggestions to correct problems in your code, for example a missing bracket or semicolon.

Either:

- Click an element underlined with a squiggly line to display corrections.

A lightbulb icon appears next to the problematic line. Click the lightbulb and check the correction suggested. If you are happy with the suggestion, click the popup message.

Figure 10-12: Fix-it example

```
8
9 // Blinking rate in milliseconds
10 #define BLINKING_RATE 500ms
11
12
13 int main()
14 {
15     // Initialise the digital pin LED1 as an output
16     DigitalOut led(LED1);
17
18     while (true) {
19         led = !led;
20         ThisThread::sleep_for(BLINKING_RATE);
21     }
22 }
23
```

- Use the Peek Problem and Quick Fix... options to correct your code. Hover over a red squiggly line in your code to display these options. Peek Problem shows the errors inline. Quick Fix... operates the same way as fix-its, click the popup message to apply a correction.

Figure 10-13: Peek problem example


```
9 // Blinking rate in milliseconds
10 #define BLINKING_RATE 500ms
11
12
13 int main()
14 {
15     // Initialise the digital pin LED1 as an output
16     DigitalOut ledA(LED1);
17
18     while (true) {
19         led = !led;
20         ThisThread::sleep_for(BLINKING_RATE);
21     }
22 }
23
```


Problems view

Keil Studio lists detected problems in your code in the Problems view.

To use the Problems view:

1. To open the view, go the View menu and select Problems, or click the Problems option in the status bar.

2. To collapse the list of problems in the Problems view, click the Collapse All icon .

You can also close the error messages one by one or click the Clear All icon  to clear all the messages at once.

Refactoring

Keil Studio supports refactoring operations such as extract function and extract variable.

You can either use the steps described below, or alternatively, you can click the lightbulb next to the code you want to refactor.

To refactor your code:

1. Select the source code you want to extract, right-click the code, and select Refactor....:
 - If you are extracting a function, an Extract to function popup message appears. Source code fragments can be extracted into a new function.
 - If you are extracting variable, an Extract subexpression to variable popup message appears. You can create a new local variable for the currently selected expression.
2. To refactor the code, click the popup message for your function or variable extraction.

Examples

- Extract a function:

Figure 10-14: Extract to function refactoring example

```
9
10 // Blinking rate in milliseconds
11 #define BLINKING_RATE 500ms
12
13
14
15 int main()
16 {
17     // Initialise the digital pin LED1 as an output
18     DigitalOut led(LED1);
19
20     while (true) {
21         led = !led;
22         ThisThread::sleep_for(BLINKING_RATE);
23     }
24 }
25
```

- Extract a variable:

Figure 10-15: Extract to variable refactoring example

```
8
9
10 // Blinking rate in milliseconds
11 #define BLINKING_RATE    500ms
12
13
14
15 int main()
16 {
17     // Initialise the digital pin LED1 as an output
18     DigitalOut led(LED1);
19
20     while (true) {
21         led = !led;
22         ThisThread::sleep_for(BLINKING_RATE);
23     }
24 }
25
```

10.2 Code linting

Linting is the automated checking of your source code for programmatic and stylistic errors. Linting intends to improve the overall quality of your code and make code reviews and tests more efficient.

Keil Studio uses clang-tidy, a clang-based C/C++ linter tool. Clang-tidy is an extensible framework for diagnosing typical programming errors, or style issues (generally anything which can be detected during static analysis of the code). Clang-tidy checks your code as you type and uses squiggly lines to highlight problems and lightbulbs to suggest fixes in the editor. The clang-tidy linter checks complement existing IntelliSense capabilities that were already available with Keil Studio and brought by clang.

10.2.1 Enable clang-tidy

Clang-tidy is not enabled by default. This topic describes how to enable clang-tidy for the active project.

About this task

To enable clang-tidy, create a `.clang-tidy` file in the root folder of the active project.

Procedure

1. From the Explorer view, right-click on the active project and select New File.
2. In the New File dialog box, type `.clang-tidy` and click OK.
Keil Studio creates a `.clang-tidy` file in your project folder.
3. To enable all the default clang-tidy checks in the `.clang-tidy` file, type `checks: '*'`.

10.2.2 Configure clang-tidy checks

For each of your projects, you can configure the checks that clang-tidy should carry out using a `.clang-tidy` file. You can decide which clang-tidy checks you want to enable or disable and configure advanced custom checks.

10.2.2.1 Enable or disable checks

If you must enable or disable particular checks, use the clang-tidy command-line format in your `.clang-tidy` files. The command-line format specifies a comma-separated list of positive and negative globs: positive globs add subsets of checks, and negative globs (prefixed with `-`) remove subsets of checks.

For example, to enable `modernize-` checks that advocate usage of modern (currently “modern” means “C++11”) language constructs, use:

```
Checks: 'modernize-*
```

See the [Clang-Tidy Checks page](#) for a complete list of available checks. A “Yes” in the Offers fixes column indicates that the check is supported. For more general information, see the [Clang-Tidy page](#).

10.2.2.2 Configure advanced custom checks

Shows an example of how to configure advanced custom checks.

In the following example, we enable all checks and provide the additional option to `modernize-use-nullptr`:

```
Checks: '*'
CheckOptions:
- key:      modernize-use-nullptr.NullMacros
  value:    NULL,CUSTOM_NULL
```

For more information, see the [official YAML website](#).

11. Manage files

This chapter describes how to search the content of files, compare, upload, download, and move files in Keil Studio.

11.1 Search the content of files

You can search the content of multiple files for the active project or for a specific folder. You can also search the content of a single file. Keil Studio searches through both saved and unsaved changes.

11.1.1 Search the content of multiple files

This section describes how to search for a term or a regular expression across all files in an active project.

Procedure

1. Click the Search icon or go to the Edit menu and select Find in Files. You can also press Ctrl+Shift+F (Windows) or Cmd+Shift+F (macOS).
The Search view opens. Enter text in the search box.
2. To search inside a single folder, go to the Explorer view, right-click the folder, and select Find in Folder.
The lower part of the view displays the search results, sorted under the files in which they appear.
3. To view only the list of files in which the text you searched for appears, click the Collapse All




icon

Note: If the Search view is hiding the Explorer view, you can click the Keil Studio icon or press Ctrl+Shift+E (Windows) or Cmd+Shift+E (macOS) to return to it.

Example 11-1: Search examples

- Including or excluding files, folders and path segments in your search:

To specify which files to include or exclude in your search:

1. Click the ellipsis (Toggle Search Details) .
2. In the files to include and files to exclude search boxes, enter file names, separated by commas, to include or exclude in your search.

For example, if you enter `resources`, `sources` in the files to exclude search box, the search excludes all files and folders named `resources` OR `sources`.

- Find and replace:

To replace the searched-for text with new text, click Toggle Replace , and enter your new text.

Click Replace All  to replace all instances of the text in all files.

To replace all instances of the text in a specific file, hover the cursor over the file name in the search results, and click the Replace All button that appears near the file name.

11.1.2 Search the content of a specific file

This section describes how to search for a term or a regular expression in a specific file.

Procedure

1. Go to the Edit menu and select Find. You can also press Ctrl+F (Windows) or Cmd+F (macOS). A search box opens at the top of the file tab. Enter text in the search box.
2. To do a search on one part of the file only, enter text in the search box, then select the lines

you want to check and click the Find in selection button .

Example 11-2: Find and replace

To replace the searched-for text with new text, click the Toggle Replace mode arrow and enter your new text. The replace mode is also available from the Edit menu > Replace option.

To replace a single instance or all instances of the text in the file, click the Replace or Replace All buttons .

11.1.3 Include or exclude search patterns

You can include or exclude search patterns in files, folders, and path segments using glob syntax (wildcards). You can also search using regular expressions.

Wildcard	Description
*	Matches zero or more characters. For example, *.html matches all HTML files.
**	Matches zero or more path segments. For example, tools** matches tools/python and tools/python/Scripts.
?	Matches any single character. For example, 3.? matches 3.4.
[]	Matches one character in the bracket. For example, 3.[0-9] matches 3.1 and 3.2.
{ }	Matches any of the conditions in the bracket. For example, {*.html, *.json} matches all HTML and JSON files.

Regular expression example: With `LED\d` you can find all instances of the string `LED` followed by a one number character, for example `LED1` and `LED2`.

11.2 Compare files

Keil Studio supports a side-by-side view of files, both for comparing local and remote versions in source control and comparing two local files. You can compare any two files in your workspace, even if they are not in the same project.

Procedure

1. Right-click a file and select Select for Compare.
2. Right-click a second file and select Compare with Selected.
Keil Studio displays both files in a single tab, with highlighted differences.

Related information

[Compare local and remote versions in Git source control](#) on page 104

[Compare local and remote versions in Mercurial source control](#) on page 115

11.3 Upload and download files or projects

This section describes how to upload or download a file or project in Keil Studio.

Procedure

- Either:
 - Upload: You can use the File > Upload Files... option to upload files to the root directory of the project selected in the list of projects available. You can also drag and drop files to add them to your project.
 - Download: You can download a project or any file in a project with the File > Download Current Selection option. You can also download the active project with the File > Download Active Project option.

11.4 Change file locations

Keil Studio puts new files and folders in the root directory. If you move these files to other folders in your project structure, Keil Studio detects the location change and builds your project with the correct file paths.

Procedure

1. To create a folder, either:
 - Right-click the project and select New Folder.
 - Go to File > New Folder.
2. If necessary, you can drag and drop files to move files around.
Warning: For Mbed projects only - You cannot move the `mbed-os` library; the `mbed-os` library must remain in the root folder.

11.5 Copy the path of a file or folder

You can copy the full or the relative path of a file or folder to the clipboard.

To copy the full path, right-click a file or folder in the Explorer view and select Copy Path.

To copy the relative path, right-click a file or folder and select Copy Relative Path.

12. Source control

This chapter describes how to use Keil Studio with the Git and Mercurial source control technologies, and how to use the History view to view your commit history.

12.1 Work with Git

Keil Studio supports the most common Git actions for your projects, including branching, stashing and synching with the remote repository.

The flow of Keil Studio matches the Git flow with the “stage” step:


1. Set a remote repository.
2. Branch.
3. Edit files.
4. Stage changes.
5. Commit.
6. Push.

Before you begin, set your GitHub credentials.

12.1.1 Set credentials for GitHub

Working with Git-based hosting services requires authentication against those services. You can connect to your Git hosting service directly from Keil Studio and link your Arm account and your GitHub account.

Procedure

1. Go to the User Profile view, click the Accounts icon  in the activity bar and select Show User Profile.
2. Click Connect to GitHub:
 - If you are already logged into GitHub in your browser, you are directed to a GitHub Application authentication screen where you can authorize the Keil Studio application to connect with your GitHub account. Read the authentication terms, and if you accept the terms, click Authorize <github-account-name>.
 - If you are not logged into GitHub in your browser, you are directed to the GitHub login screen. Provide your GitHub credentials and click Sign in.

Next, the GitHub Application authentication screen displays where you can authorize the Keil Studio application to be used with your GitHub account. Read the authentication terms, and if you accept the terms, click Authorize <github-account-name>.

Your browser returns you to the User Profile view in Keil Studio. Your GitHub account is now listed in the view, under your Keil Studio account information.

Related information

[Manage accounts](#) on page 14



[User Profile view](#) on page 14

12.1.2 Interface and features reference

Source control in Keil Studio is handled in two views: The Source Control view, for handling the current work, and the History view to see previous commits. They always show the active project.

The image highlights:

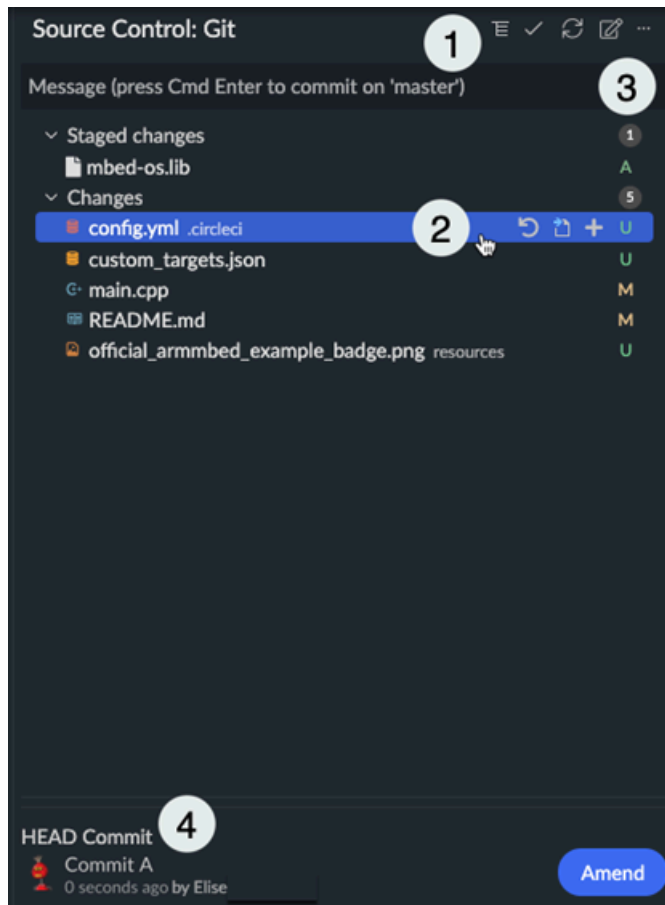
1. The actions menu. Both:

- Toggle to List View 
- Toggle to Tree view 

buttons allow you to display your changes as a list or as a tree view. Note the ... button for more actions.

2. The buttons available on each file when you hover over a file name: Discard Changes, Open File, Stage Changes (+), and Unstage Changes (-).
3. Five changed files and one staged file.
4. The last (head) commit.

Figure 12-1: The 'Source Control' view for Git.



Group	Features	Comments
Local changes	Discard all changes	Revert all changed files to their state as of the last time they were pushed (or their starting state if they have never been pushed).
Local changes	Stage and Stage all changes	Use the Stage option for files you want to add to your next commit. Staging files manually breaks your work into logical units, each in its own commit, rather than having to commit all files together.
Local changes	Unstage and Unstage all changes	Return a staged file, or all staged files, to the Changes list.
Local changes	Refresh	>Update the list of local changes.
Commits	Commit	Put all staged files into a single record of local changes. The commit is the record you can push to the remote repository.
Commits	Add signed-off-by	Tag your commit and add comments, a signature, the date, and your email. For some projects, this is a requirement for all contributions submitted as pull requests.
Commits	Commit (amend)	Change your last commit (head commit) by amending the message and the content of the commit, or only the content. This option combines your staged changes with the head commit.
Commits	Amend (button)	Change your last commit (head commit) by amending the message and the content of the commit, or only the content. This option combines your staged changes with the head commit.
Branch management	Branch	Create a new local branch or checkout an existing branch. The default branch for a newly set repository is Master.

Group	Features	Comments
Branch management	Fetch	Fetch changes from the remote, but do not merge them into the current local branch.
Branch management	Merge	Incorporate changes from another branch into the current branch. For example, add the latest changes from the remote repository to your local copy.
Branch management	Pull	Apply the latest changes from the default remote repository to your local repository (fetch and merge). We recommend pulling only with a clean working copy; if you have any uncommitted local changes, use fetch and merge.
Branch management	Pull from...	Pull from a specific remote, rather than the current one.
Branch management	Push	Send new local commits to the remote.
Branch management	Push to...	Push to a specific remote, rather than the current one.
Stash	Stash	Set aside your changed files. This allows: a) Switching branches without committing local changes; apply the stash when you are ready to go back to the branch. b) Applying work from one branch to another; apply the stash in the new branch.
Stash	Apply latest stash	dd the latest stashed changes to the branch, without clearing the stash.
Stash	Apply stash	Add a specified stash to the branch, without clearing the stash.
Stash	Drop stash	Discard all changes in the stash.
Stash	Pop latest stash	Apply the latest stash and clear the stash.
Stash	Pop stash	Apply a specified stash and clear the stash.
Initialize	Initialize the active project	Turn an unversioned project into a Git repository.
Publish	Publish project > Set remote URL	Publish your new or initialized project to an existing Git remote repository.
Publish	Publish project > Publish to a new GitHub repository	Publish your new or initialized project to a new GitHub remote repository.
Publish	Fork on GitHub	Fork a Git project and publish it to a new GitHub repository.

12.1.3 Configure a project for source control and collaboration

Whether you have decided to work from a new example project, or you have cloned an existing project, several options are made available to you when working with Git in Keil Studio.

For newly created projects (which are not yet configured for source control):

- You can initialize a project for Git, in other words, turn your unversioned project into a Git repository. This means you can work locally on your project and then, when you are ready to share your work with other people, decide to publish your local project to a remote repository.
- You can publish a project right away. Two options are available. You can either:
 - Point to an existing remote repository (without any commits).
 - Publish a new repository on GitHub from Keil Studio.

For cloned projects:

A project cloned from GitHub or another Git hosting service automatically points to the remote repository you cloned the project from. Keil Studio offers you the possibility to fork a cloned project and publish it to your GitHub account in one go.

12.1.3.1 Initialize and publish a project

Learn how to initialize your project for Git and publish your project.

Before you begin

- Your active project must not yet be configured for source control.
- You must connect your GitHub account with Keil Studio. To learn how to connect your GitHub account with Keil studio, see [Set credentials for GitHub](#).

Procedure

1. Go to the Source Control view.
 - If you did not previously initialize the project as a Git repository when creating the project, an “Active project is not under version control” message displays.
 - a. You can decide to simply initialize the project for Git (turn your project into a Git repository) without publishing it yet. Click the ... at the top of the Source Control view and select Initialize the Active Project.
 - b. You can also publish the project (this initializes the project for Git and creates a remote repository on GitHub to publish your changes). Click the Publish Project button or click the ... at the top of the Source Control view and select Publish Project.
 - If the project is already initialized for Git, click the ... at the top of the Source Control view and select Publish Project.

The Publish dialog box opens.

2. Use the Publish dialog box to publish your repository:
 - To use an existing remote repository (without any commits):
 - a. Select the Set remote URL radio button and enter a URL for your remote repository. The format can be either SSH (if you have set up an SSH key) or HTTPS. The URL must not contain a branch name. For example, for GitHub:
 - SSH: `git@github.com:user-name/repo-name`
 - HTTPS: `https://github.com/user-name/repo-name`
 - b. Click Set remote repository.
 - To publish a new repository:
 - a. Select the Publish to a new GitHub repository radio button and enter a repository name and description.
 - b. By default Keil Studio creates a private repository. If you want to create a public repository, clear the Private repository checkbox.
 - c. Click Publish.

Setting a remote repository or publishing a new repository through the UI is only possible once for each new project. Setting a remote repository is not possible for a project you clone from Git (the remote repository is automatically set to the repository you cloned it from).

12.1.3.2 Fork a Git project

Learn how to fork a project cloned from GitHub or another Git hosting service and publish it to your GitHub account in one go.

Procedure

1. In the Explorer view, set the project you want to fork as the active project.
2. Move to the Source Control view.
3. Click the ... button and select Fork on GitHub.
4. Depending on where the project comes from, Keil Studio forks the project directly or asks you to set up a destination repository.
 - If the project was cloned from GitHub, then Keil Studio forks the project directly. Check your GitHub account.
 - If the project was cloned from another Git hosting service, the Set Up Destination GitHub Repository dialog box opens:
 - a. Modify the repository name if needed and add a description.
 - b. By default Keil Studio creates a private repository. If you want to create a public repository, clear the Private repository checkbox.
 - c. Click Next.

Keil Studio forks the project and creates a repository in your personal workspace on GitHub.

12.1.4 Create or switch branches

This section describes how to create a branch or switch to a different branch.

About this task

Git uses branches to isolate your work from the work of other people or from code that must remain stable. Keil Studio allows creating branches, and syncing the remote and local branches. You can see which branch you are working on from the status bar (when first setting a repository, you are on the master branch by default).

Procedure

1. Click the current branch in the status bar.
The available branches are listed at the top of the window.
2. Create or switch:
 - To create a branch: Select Create new branch... and enter a name to create a branch.

The name of the new branch should not contain spaces.

The branch is created locally; the branch publishes to the remote repository the first time you push from the branch.

- To switch to a different branch, search for and select an existing branch in the list.

12.1.5 Manage local files

This section describes the different options to manage your local files and explains how to update the remote.

12.1.5.1 File statuses

Statuses display to the right of each file to indicate changes.

- U: Untracked - a new file in the Changes list (not yet staged).
- A: Added - a new file in the Staged changes list.
- M: Modified - an existing file in either the Changes or Staged changes list.
- D: Deleted - a deleted file.
- C: Merge Conflict - you must resolve the merge conflict before you can push.

12.1.5.2 View changes

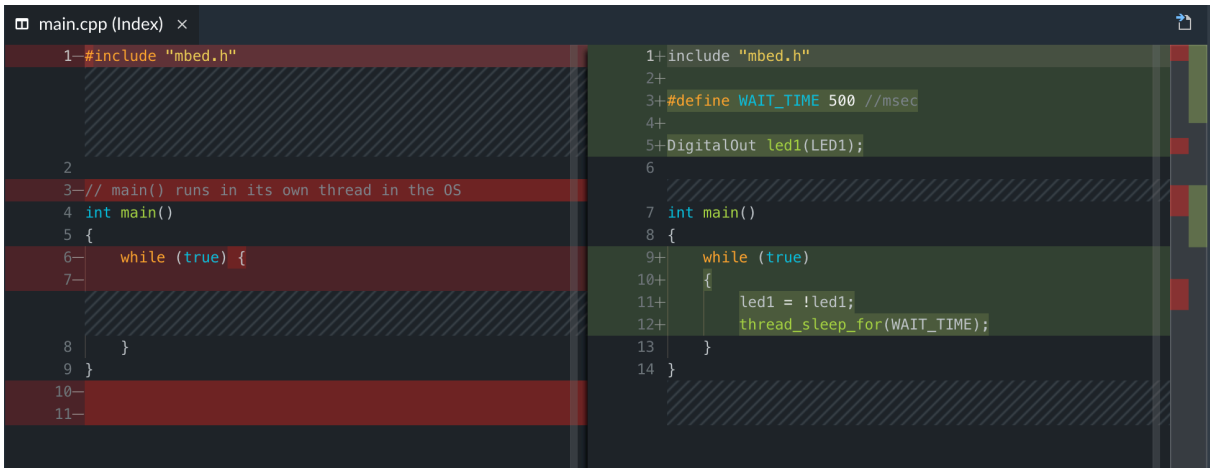
This section describes how to view the changes you have made to a file in Keil Studio.

Procedure

- To view the changes you have made to a file, double-click its name. The file opens.

The changes open in a new tab, comparing what was available in the file before the changes (in red) and what has changed (in green).

Figure 12-2: Double-click a file to view its side-by-side comparison

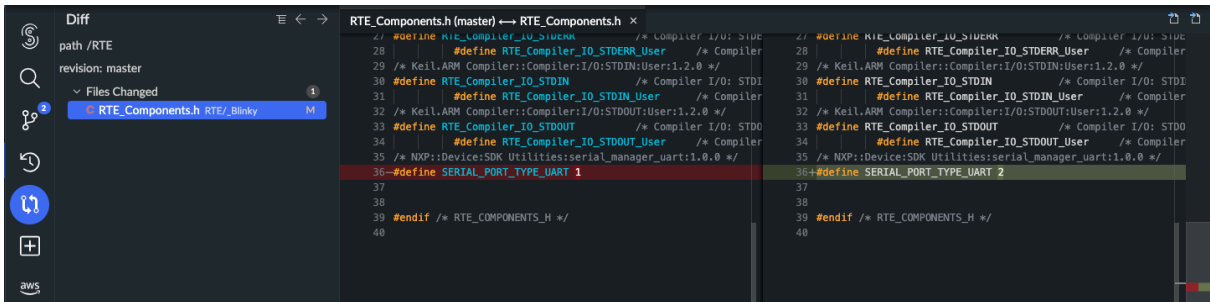


- To compare the same file or folder across multiple Git branches, right-click a file or folder and select Compare With. Keil Studio shows a list of available branches. Select a branch.

If you compare a single file, Keil Studio opens a diff view of that file.

If you compare an entire folder, Keil Studio opens a tab listing all the files. Click a file to open a diff view:

Figure 12-3: List of files changed in the folder



Next steps

You can open the file for editing by clicking the Open File icon in the top right-hand corner of the screen.

Go back to the changes by clicking the Open Changes icon.

Note that you can open several files at once for editing: hold Shift or else Cmd (macOS) or Ctrl (Windows and Linux) and select the files, then right-click and select Open File.

When in the editor, if there are changes to a file, the Open Changes icon is also available when you open the file.

12.1.5.3 Stage, commit, and push changes

This section describes how to stage, commit, and push changes in Keil Studio.

About this task

Use staging to manually control which of your edited files are added into each commit. This breaks your work into logical units, each in its own commit, rather than having to commit all files together. You can always remove a file from the Staged changes list, and later add it to a different commit. When you are ready, commit and push your changes.

Procedure

1. In the Source Control view, all new and modified files are listed under Changes.
2. Hover your mouse over the file you want to commit and click + to stage your changes. The file is listed under Staged changes. To stage several files at once, hold Shift or else Cmd (macOS) or Ctrl (Windows and Linux) and click the files you want to stage, then click +.
3. In the Message box, enter a commit message for the staged changes.
4. Click Commit or Commit (Signed Off).
The committed changes are visible at the bottom of the Source Control view, and in the History view.
5. To push the commit to the remote branch, click ... > Push.

Note:

- To stage or unstage one or several files, you can also select the file or files, right-click and select Stage Changes or Unstage Changes.
- If you try to push your changes to a repository on which you do not have permission, you'll get an error message with a Fork on GitHub button. Click this button to fork the repository and then try to push your changes again. See also [Fork a Git project](#).
- If you try to push your changes to a repository on which you do not have permission and a fork already exists, you'll get an error message with a Set Fork as Default button. Click this button to use your fork and then try to push your changes again.

12.1.5.4 Amend local commits

Before pushing to the remote to share your work with others, you can rewrite local commits.

To rewrite local commits, you can:

- Add, update, or remove files in your last commit and change the commit message.
- Change multiple commits at once.



Note

If you have already pushed your changes to the remote and notice that something is not quite right, you can still fix your commits and force push the changes. However, you must be absolutely certain that nobody has pushed commits to the remote before doing a force push because force pushes overwrite commits.

12.1.5.4.1 Change the last commit

This section describes how to change your last commit in Keil Studio.

About this task

With the Commit (amend) option, you can amend your most recent commit and change the content of the commit by adding, updating, or removing files. You can also optionally change the commit message.

Procedure

1. Stage the files that you want to include in your last commit.
Note: To remove a file from your last commit, first delete it from the Explorer view (the file appears as deleted D in the Changes list), then stage the deleted file.
2. Click ... > Commit (amend).
3. Add a commit message in the message box that opens. Or, to reuse the last commit message and only update the content of the commit, press Enter.

12.1.5.4.2 Change multiple commits

This section describes how to change multiple commits in Keil Studio.

About this task

With the Amend button, you can roll back several commits to reset your head commit to a prior state. The changes done between the original head commit and the current head commit are staged. This way you can add, modify, or remove files and stage new changes.

Procedure

1. Click the Amend button for each commit that you want to amend.
All the changes you had committed are staged. You can decide which changes to stage or unstage and combine commits.

Use the Unamend button if you change your mind and prefer to keep a commit untouched. This resets the head commit to the commit you have just “unamended”. If there are several commits to “unamend”, click Unamend all commits (-).

2. Once you are happy with the changes, in the Message box, enter a commit message for the staged changes.
3. Click Commit or Commit (Signed Off).

Results

The changes commit and the commit can be seen at the bottom of the Source Control view, and in the History view. In the History view, you can see that the amended commits are replaced by the new commit.



You can clear all the commits in the Commits being amended list at once with the Clear amending commits (x) button. The files staged while amending the commits remain in the Staged changes list.

12.1.5.5 Ignore files

To keep your file list tidy and navigable, you can exclude files from the Source Control view. For example, by default Keil Studio does not list any Mbed OS files.

Use `.gitignore` to list the files you want to exclude. The `.gitignore` file exists by default in all Mbed projects, and is visible and editable in Keil Studio.

For more information about how `.gitignore` files work and how to use them, see the [Ignoring Files chapter](#) on git-scm.com.

12.1.6 Synchronize

If the remote repository has changed since you last pulled, you must synchronize your local copy before you can push any of your own changes to the remote.

The Synchronize Changes indicators on the left-hand side of the status bar show:

- How many changes have been pushed to the remote repository since your last pull.
- How many commits you have on your local copy.

The Synchronize Changes indicators are also a button that displays the available synchronization options.

Synchronization options

Keil Studio offers three synchronization options. Each one combines different Git operations to automatically manage the changes.

Click the Synchronize Changes button to display the available options:

- Pull and push commits from and to 'origin/<branch name>': Apply the latest changes from the remote repository to your local repository by doing a pull (fetch and merge), then push your changes to the remote repository. This option might generate merge conflicts.
- Fetch, rebase, and push commits from and to 'origin/<branch name>': Put aside the local changes, fetch the remote changes to bring the local up to date, reapply your local changes, and push to the remote. The rebase option compresses all the changes into a single "patch" to integrate with the remote branch. This option might generate merge conflicts.
- Force push commits to 'origin/<branch name>': Overwrite the remote branch with your local branch, regardless of the status of that remote branch.

Resolve merge conflicts

Synchronizing remote and local changes by pulling or rebasing can lead to merge conflicts when a single file has been edited in both locations. Merge conflicts can also happen when you apply or pop a stash, if the stash contains changes that contradict further work on the branch.

By default, when Git sees a conflict between two branches being merged, Git:

1. Adds merge conflict markers, <<<<<< ===== >>>>>>, into your code.
2. Marks the file as conflicted.
3. Lets you resolve the merge conflicts.

The top half of a conflict is the branch you are merging into, and the bottom half is from the commit that you are trying to merge in.

So, for example, if you pull (fetch and merge):

- The top half shows your local changes.
- The bottom half shows the remote changes which you were trying to merge in.

To resolve a conflict, you have to decide which part you want to keep or merge the contents yourself, and then remove all the merge conflict markers. Once you are happy with the corrections on a given file, click + to stage your changes.

For more information about merge conflicts, see the [About merge conflicts page](#) of the GitHub Help.

12.2 Work with Mercurial

Keil Studio supports the most common Mercurial actions for Mbed projects hosted on os.mbed.com, including branching and synching with the remote repository.

The Mercurial flow in Keil Studio:

1. Set a remote repository.
2. Branch.
3. Track files.
4. Edit files.
5. Commit.
6. Push.



Keil Studio includes its own version of Mercurial; you do not need to manually install it.

Before you begin, check the [Credentials](#) section.

12.2.1 Credentials

For Mercurial, on any operating system, your credentials are taken from the Arm account with which you logged into Keil Studio. You can work with all public repositories, and any private repository to which you have access through os.mbed.com.





Keil Studio supports Mercurial repositories from os.mbed.com only.

12.2.2 Interface and features reference

Source control in Keil Studio is handled in two views: The Source Control view, for handling the current work, and the History view to see previous commits. Both views always show the active project.

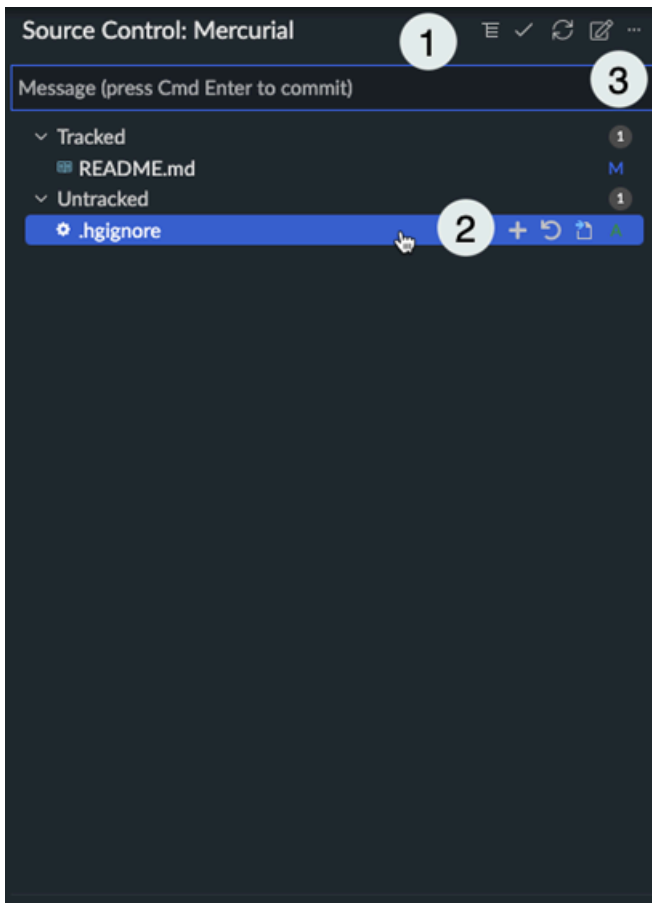
The image highlights:

1. The actions menu. Both the Toggle to List View and the Toggle to Tree view buttons allow you to display your changes as a list or as a tree view:
 - Toggle to List View: 
 - Toggle to Tree view: 

Note the ... button for more actions.

2. The buttons available on each file when you hover over a file name: Track (+), Untrack (-), Discard Changes, and Open File.
3. One tracked file - a Modified (M) file, which had already been committed to Mercurial once and has changed since. One untracked file - an Added (A) file, which has not been committed yet.

Figure 12-4: The 'Source Control' view for Mercurial



Group	Features	Comments
Local changes	Track and Track all files	Use the Track option to monitor the files for changes. A tracked file with changes is automatically added to a commit. You can move files back to the Untracked list only if you have never committed them (these files are marked with "A", for "Added"). Once you commit files, they remain tracked.
Local changes	Discard all changes	Revert all changed files to their state as of the last time they were pushed (or their starting state if they have never been pushed).<
Local changes	Refresh	Update the list of local changes.
Commits	Commit	Put all tracked files into a single record of local changes. The commit is the record you can push to the remote repository.
Commits	Add signed-off-by	Tag your commit and add comments, a signature, the date, and your email. For some projects, this is a requirement for all contributions submitted as pull requests.
Branch management	Branch	Create a new local branch or checkout an existing branch. The default branch for a newly set repository is "default".
Branch management	Pull	Apply the latest changes from the default remote repository to your local repository (pull and merge).
Branch management	Push	Send new local commits to the remote.
Branch management	Push to...	Push all branches to a specific remote, rather than the default remote.

Group	Features	Comments
Branch management	Merge heads	Merge two or more heads of a branch into a single head.
Publish	Set remote URL	Publish your new project to an existing Mercurial remote repository.
Publish	Convert to a Git Repository and Fork on GitHub	Convert your Mercurial project to a Git project and publish it to a new GitHub repository.

12.2.3 Configure a project for source control and collaboration

For newly created projects (which are not yet configured for source control):

You must manually set the remote repository for a given project (that is to say point to an existing remote repository without any commits). You must create remote repositories directly through os.mbed.com.

For cloned projects:

A project cloned from Mercurial automatically points to the remote repository you cloned the project from. Keil Studio offers you the possibility to convert a project cloned from Mercurial to Git and publish the converted project to your GitHub account in one go.

12.2.3.1 Set a remote repository

Learn how to set a remote repository.

Before you begin

Your active project must not yet be configured for source control.

Procedure

1. Initialize the project for source control:
 - a. Go to the Source Control view.
 - b. Click the Set remote URL button.

The Set Remote URL dialog box opens.
 - c. Enter a URL for your remote repository.

Note: The URL must not contain a branch name.

2. Set a remote repository. Click Set remote repository.

Setting a remote repository through the UI is only possible once for each new project.

12.2.3.2 Convert a Mercurial project to Git

Learn how to convert a project cloned from Mercurial to Git and publish the converted project to your GitHub account in one go.

Procedure

1. In the Explorer view, set the project you want to convert as the active project.
2. Move to the Source Control view.
3. Click the ... button and select Convert to a Git Repository and Fork on GitHub. The Set Up Destination GitHub Repository dialog box opens.
4. Modify the repository name if needed and add a description.
5. By default Keil Studio creates a private repository. If you want to create a public repository, clear the Private repository checkbox.
6. Click Next. The Match authors to GitHub users dialog box opens.
7. This step is not mandatory, but Arm recommends including the GitHub username of each contributor to the project (or alternatively providing their GitHub email address). Including the GitHub username of each contributor preserves the commit history of the project you are converting.
8. Click Migrate now. Keil Studio converts the project to Git and creates a repository on GitHub.

12.2.4 Create or switch branches

This section describes how to create a branch or switch to a different branch in Keil Studio.

About this task

Mercurial uses branches to isolate your work from other people's work or from code that must remain stable. Keil Studio allows creating branches, and synching the remote and local branches. You can see which branch you are working on from the status bar (when first setting a repository, you are on the "default" branch by default).

Procedure

1. Click the current branch in the status bar. The available branches are listed at the top of the window.
2. Create or switch:
 - To create a branch: Select Create new branch... and enter a name to create a branch.

The name of the new branch must not contain spaces.

The branch is created locally; the branch publishes to the remote repository the first time you push from it.

- To switch to a different branch, search for and select an existing branch in the list.

12.2.5 Manage local files

This section describes the different options to manage your local files and explains how to update the remote.

12.2.5.1 File statuses

Statuses display to the right of each file to indicate changes.

- A: Added - a new file in either the Tracked or Untracked list.
- M: Modified - an existing file in the Tracked list.
- D: Deleted - a deleted file.
- C: Merge Conflict - you must resolve it before you can push.

12.2.5.2 View changes

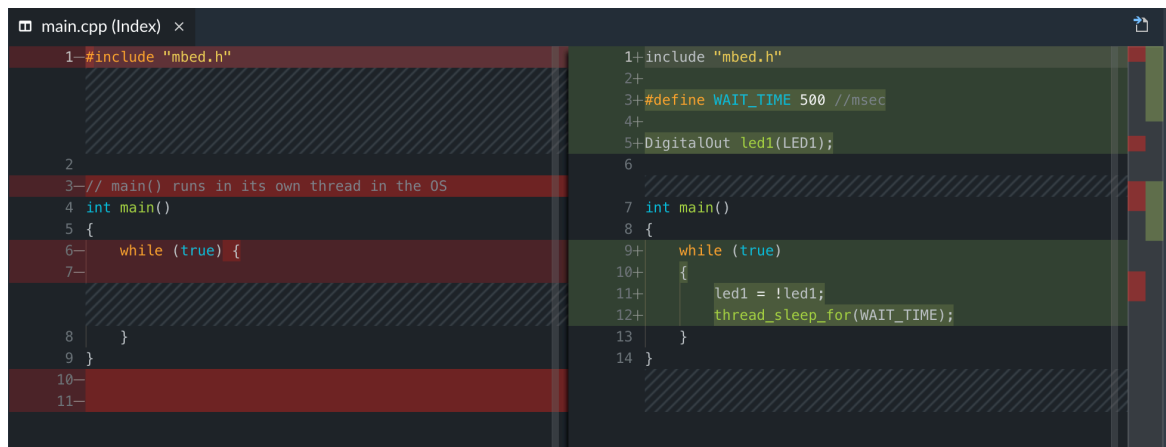
This section describes how to view the changes previously made to a file.

Procedure

- Either:
 - Double-click its name to open the file.

The changes open in a new tab, comparing what was available in the file before the changes (in red) and what has changed (in green).

Figure 12-5: Double-click a file to view its side-by-side comparison



```
main.cpp (Index) ×
1-#include "mbed.h"
2+
3-// main() runs in its own thread in the OS
4 int main()
5 {
6- while (true) {
7-
8 }
9 }
10-
11-

1+include "mbed.h"
2+
3+#define WAIT_TIME 500 //msec
4+
5+DigitalOut led1(LED1);
6
7 int main()
8 {
9+ while (true)
10+ {
11+     led1 = !led1;
12+     thread_sleep_for(WAIT_TIME);
13 }
14 }
```

- Open the file for editing by clicking the Open File icon in the top right-hand corner of the screen. Go back to the changes by clicking the Open Changes icon.

When in the editor, if there are changes to a file, the Open Changes icon is also available when you open the file.

12.2.5.3 Track, commit, and push changes

This section describes how to track, commit and push changes in Keil Studio.

Procedure

1. In the Source Control view, all new files are listed under Untracked.
Files that have already been committed once are automatically shown in the Tracked list when you modify them.
2. Hover your mouse over the file you want to commit and click + to track your changes. The file is listed under Tracked.
3. In the Message box, enter a commit message for the tracked changes.
4. Click Commit or Commit (Signed Off).
The committed changes can be seen in the History view.
5. To push the commit to the remote branch, click ... > Push.
Note: To track or untrack one or several files, you can also select the file or files, right-click, and select Track or Untrack.

12.2.5.4 Ignore files

You can exclude files from the Source Control view to keep your file list tidy and navigable. For example, by default Keil Studio does not list any Mbed OS files.

Use `.hgignore` to list the files you want to exclude. The `.hgignore` file exists by default in all Mbed projects, and is visible and editable in Keil Studio.

For more information about how `.hgignore` files work and how to use them, see the [Mercurial documentation](#).

12.2.6 Synchronize

If the remote repository has changed since you last pulled, you must synchronize your local copy before you can push any of your own changes to the remote.

Resolve merge conflicts

Synchronizing remote and local changes can lead to merge conflicts when a single file has been edited in both locations.

By default, when Mercurial sees a conflict between two branches being merged, Mercurial:

1. Adds merge conflict markers, `<<<<<< ===== ||| ||| >>>>>>`, into your code.
2. Marks the file as conflicted.
3. Lets you resolve the merge conflicts.

The top half of a conflict is the branch you are merging into, and the bottom half is from the commit that you are trying to merge in.

So, for example, if you pull (pull and merge):


- The top half shows your local changes and your base version (after the ||||| markers).
- The bottom half (after the ===== markers) shows the remote changes which you were trying to merge in.

To resolve a conflict, you have to decide which part you want to keep or merge the contents yourself, and then remove all the merge conflict markers. Once you are happy with the corrections on a given file, click + on the file entry in the Merge changes list to resolve the conflict and move the file to the Tracked list.

For more information about merge conflicts with Mercurial, see the [Tutorial - Merging conflicting Changes](#).

12.3 History view

The History view shows your commit history.

To view the changes that have been made for a given commit, click the eye icon . The commit info includes the commit message, date, and a list of changed files.

To see the details of the changes on a given file, double-click the file. The changes open in a new tab.

13. Monitor and debug

This chapter describes the Serial Monitor view, and how to debug supported development boards in Keil Studio.

13.1 Use the Serial Monitor view

The Serial Monitor view displays the output of the board. By default, Keil Studio automatically detects any board with a WebUSB-enabled CMSIS-DAP interface that you connect to your computer and opens a Serial Monitor view. If you connect multiple boards, Keil Studio opens a view for each board.



The first time you connect your board, you must click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target.

You can have only one Serial Monitor view open for each connected board. Before you open an external serial monitoring program, close the Serial Monitor view for your board in Keil Studio.

To reopen the Serial Monitor view, either:

- Select View > Serial Monitor.
-

In the Explorer view, click the Open serial monitor button .

You can disable the automatic opening of the Serial Monitor view from the preferences. Select File > Settings > Open Preferences and search for the Serial Monitor preference. Clear the Automatically Open checkbox if you do not want the Serial Monitor view to open automatically when you connect your board.

To view the output of a board correctly, select the appropriate value from the Baud rate drop-down list. The baud rate you select must be the same as the baud rate of your active project.

To download the output, click the Save output icon .

To clear the Serial Monitor view, click the Clear output icon .

13.2 Debug a project with Keil Studio

This section describes the debugger mode and debugging tools available in Keil Studio.

13.2.1 Introduction

You can step debug a project on any connected board that supports a WebUSB-enabled CMSIS-DAP interface or an ST-LINK interface. The debugger mode provides different ways of checking what your code is doing while it runs. You can step through your code, examine the execution path of your code, or look at the values stored in variables, and more.

The section explains how to debug with development boards that are automatically detected and configured. For more information about supported hardware, see [Supported development boards](#) and [Supported debug probes](#). If you are using an external stand-alone debug probe, or if auto-detection is not supported, you must first [set up a custom target](#).



Debugging features are not available for Mbed 2 projects. Arm recommends that you upgrade to Mbed OS 5 or 6 to benefit from all the features. See [Upgrade from Mbed 2 to Mbed OS 5 or 6](#) for more details.



13.2.2 Debug first steps

This section explains how to start a debug session.

Before you begin

If you are experiencing problems running an Mbed project on an ST board, set the Connect Mode preference to underReset to be able to flash a project to your board and do debugging. See [Advanced debugger settings](#) for more details on the different connect modes.



Procedure

1. Set the project you want to debug as the active project.
2. Connect your board over USB and ensure the correct target hardware is selected for your project. The first time you connect your board, you must click the Connect to target hardware button  to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target.
3. Click the debug button, . Keil Studio automatically builds and flashes your project to the connected board. The Debug view displays and the debug session starts. The debugger stops at the function “main” of your application.

13.2.3 Restart or stop the debugger




This section explains how to restart or stop the debugger.

Procedure

- To restart the debugger, click the Restart button 
 - If you click the Restart button when the debugger is running, the debugger continues to run until a breakpoint is hit or until you click Pause.
 - If you click the Restart button when the debugger is paused, the debugger starts to debug the code from the start. Restarting from a paused state is faster than stopping the debugger and initiating a new debug session from the Explorer view.
- To stop the debugger and return to the editor, click the Stop button .

13.2.4 Navigate your code using the step buttons

There are several options available to help you navigate your code quickly.

- Step over : Advance the debugger to the next source line that follows in the program execution to go straight to the parts of code you are more interested in.
- Step into : Advance the debugger into each function. The debugger then breaks on the first line that gets executed in the function.
- Step out : Advance the debugger until the current function returns (in other words, advance all the way through the current function).



Note that Step over and Step out operations halt on breakpoints.


13.2.5 Set breakpoints

Breakpoints are useful when you know which part of your code must be examined. To look at values of variables, or to check if a block of code is getting executed, you can set one or more breakpoints to suspend your running code.



Before you begin

By default, Keil Studio stops on the first line of `main()`. To change this default behavior, select File > Settings > Open Preferences > Debug > Run To Main. To keep the program running until the first user breakpoint, clear the checkbox.

Procedure

1. Click the margin to the left of a line of code in the editor. A red dot displays. You can also right-click the margin and select Add Breakpoint.
2. To start debugging, click the Continue button . The debugger runs to the first breakpoint it encounters and stops. A yellow arrow displays next to the statement on which the debugger paused. The statement highlights in yellow.


Next steps

- Once you have set a breakpoint, you can remove the breakpoint, or disable and enable the breakpoint again. Right-click the breakpoint and select Remove Breakpoint or Disable Breakpoint (or Enable Breakpoint if the breakpoint is disabled). You can also remove a breakpoint by clicking that breakpoint in the margin.
- To remove, or disable, and enable breakpoints, more options are available from the Breakpoints list:
 - Activate/Deactivate Breakpoints : Activate or deactivate all breakpoints at once. Activating or deactivating breakpoints does not change the enable or disable state of each breakpoint.
 - Remove All Breakpoints : Remove all breakpoints at once.
 - When you right-click a breakpoint from the list, the following options are available: Remove Breakpoint, Remove All Breakpoints, Enable All Breakpoints, Disable All Breakpoints.
 - To disable or enable a breakpoint, select the checkbox next to the breakpoint in the list.

13.2.6 Set function breakpoints

With function breakpoints, you can break execution when a function is called. Breaking execution is useful, for example, when you know the function name but not the functions location.

Procedure

1. Click the Add Function Breakpoint button  in the Breakpoints list. The Add Function Breakpoint dialog box opens.
2. Type the name of the function you are looking for (function names are case-sensitive) and click OK.

13.2.7 Examine threads and call stacks

You can examine and work with threads in the code that you are debugging. Working with threads is useful for debugging multithreaded applications. Each thread appears on a separate row in the

Threads list and has a call stack. In the current version of Keil Studio, only one thread displays in the Threads list (Main).

Procedure

- When you select the Main thread, the call stack displays in the Call Stack list. The call stack shows the order in which methods and functions are called. The call stack list is a good way to examine and understand the execution flow of your code.
- When you right-click the Main thread, the following options are available:
 - When the debugger is paused, click Continue, Step over, Step into or Step out to navigate the code.
 - When the debugger is running, click Pause to pause the debugger.
- When you right-click a call stack from the Call Stack list, the Copy Call Stack option becomes available. The Copy Call Stack option allows you to copy a call stack to the clipboard to further investigate a problem.

13.2.8 Inspect variables

You can inspect variables and check if they are storing the values you expect. If you find an incorrect value, find out where it was set (you might need to restart the debugger, look at the call stack, or both).

Local variables, global variables, and function arguments are shown in the Variables list.

When the debugger is paused, to see a properties current value, hover over the object with your cursor. You can also view variable values in the Variables list.

The value <not in scope> means that a variable is known but is not visible at the current location of the program.

Global variables are grouped by modules (compilation units) in the Variables list under the Global entry. Modules display as <file_name>: <{alias}/file_path>. Where alias is either {cmsisPacks} or {workspace}. Click a module, to see the global variables defined for that module.



Only modules with global variables are visible.

You can display variable values inline in the editor with the Inline Values setting. Go to File > Settings > Open Preferences > Debug and select the Inline Values checkbox.

When the debugger is paused, you can copy the value of a variable. From the Variables list, right-click a variable and select Copy Value.



Double-click an entry in the Call Stack list to see the Local Variable values for that entry.

13.2.9 Inspect registers

The Registers list displays register contents for the detected CPU.

The following tables provide register details for the Armv6, Armv7, and Armv8 architectures.

Armv6

Register category	Description
Core	Shows the CPU core registers (R0 to R15) and the processor status register (xPSR). Note that R13 (SP) means current Stack Pointer, R14 (LR) means Link Register, R15 (PC) means Program Counter.
Banked	Shows the Main Stack Pointer (MSP) and Process Stack Pointer (PSP). Depending on the currently active stack pointer of the CPU, R13 (SP) either shows the value of MSP or PSP. See the Status category below.
System	Shows more CPU register values (BASEPRI, PRIMASK, FAULTMASK, CONTROL).
Status	Shows CPU state details. Mode can be "Thread" or "Handler". Privilege shows the CPU code execution and can be "Privileged" or "Unprivileged". Stack shows the currently selected stack pointer (MSP or PSP).

Armv7

Register category	Description
Core	Shows the CPU core registers (R0 to R15) and the processor status register (xPSR). Note that R13 (SP) means current Stack Pointer, R14 (LR) means Link Register, R15 (PC) means Program Counter.
Banked	Shows the Main Stack Pointer (MSP) and Process Stack Pointer (PSP). Depending on the currently active stack pointer of the CPU, R13 (SP) either shows the value of MSP or PSP. See the Status category below.
System	Shows more CPU register values (BASEPRI, PRIMASK, FAULTMASK, CONTROL).
Status	Shows CPU state details. Mode can be "Thread" or "Handler". Privilege shows the CPU code execution and can be "Privileged" or "Unprivileged". Stack shows the currently selected stack pointer (MSP or PSP).
FPU	FPU (Floating Point Unit) is only shown if the Floating Point (FP) extension is enabled. Shows FP extension registers: thirty-two 32-bit single-precision registers (S0 to S31) and sixteen 64-bit double-precision registers (D0 to D15). Also shows selected bits of the Floating-Point Status and Control Register (FPSCR). See the ARMv7-M Architecture Reference Manual for more details on FPSCR. Values are shown in both binary and numeric format.

Armv8

Register category	Description
Core	Shows the CPU core registers (R0 to R15) and the processor status register (xPSR). Note that R13 (SP) means current Stack Pointer, R14 (LR) means Link Register, R15 (PC) means Program Counter.
Banked	Shows the Main Stack Pointer (MSP) and Process Stack Pointer (PSP). Depending on the currently active stack pointer of the CPU, R13 (SP) either shows the value of MSP or PSP. See the Internal category below. Shows also special-purpose registers: PRIMASK, BASEPRI, and FAULTMASK (Mask registers), and CONTROL. MSPLIM and PSPLIM (Main and Process stack pointer Limit registers) can also display depending on the CPU.

Register category	Description
FPU	FPU (Floating Point Unit) is shown if only the Floating Point (FP) extension is enabled. Shows FP extension registers: thirty-two 32-bit single-precision registers (S0 to S31) and sixteen 64-bit double-precision registers (D0 to D15). Also shows selected bits of the Floating-Point Status and Control Register (FPSCR). See the Arm v8-M Architecture Reference Manual for more details on FPSCR. Values are shown in both binary and numeric format.
FPU-MVE	This category is only shown if the MVE (M-Profile Vector Extension) is enabled (not the FP extension). MVE uses registers of the FP extension.
MVE	This category is shown when both the FPU and MVE extensions are enabled. The MVE extension provides operations on various SIMD data types.
Secure	This category is only shown when the Security (TrustZone) extension is enabled. When the core is in Secure state, it can access both Secure and Non-secure memories. If the Security extension is enabled, the processor starts up in Secure state.
Non-Secure	This category is only shown when the Security (TrustZone) extension is enabled. When the core is in Non-secure state, it can access Non-secure memories only.
Internal	Shows CPU state details. Mode can be "Thread" or "Handler". Privilege shows the CPU code execution and can be "Privileged" or "Unprivileged". Stack shows the currently selected stack pointer (MSP or PSP).

13.2.10 Advanced debugger settings

The following tables describe advanced settings for the debugger. The settings are available in the Debug or Run categories in File > Settings > Open Preferences.

Most of the settings in the Debug and Run categories are related to how Keil Studio connects to and resets a board, and how the flash download of code works.

For flash download, you can either:

- Use DAPLink (Use Daplink option in the Run category). In this case, the DAPLink firmware takes care of the flash download.
- Use a WebUSB-enabled CMSIS-DAP firmware or an ST-LINK firmware. In this case, flash algorithms contained in the CMSIS-Pack that is used in your project are deployed in the background to do the flash download. You can refine how the debugger behaves by selecting your preferred options.

See the CMSIS-Pack documentation for more information on [flash programming algorithms](#).



There are also options to change how panels are displayed in the user interface in the Debug category.

Connection and flash download options in the Debug category:

Setting	Description
Connect Mode	Controls the operations that are executed when the debugger connects to the board. The options are: <ul style="list-style-type: none"> haltOnConnect (default): Stops the CPU of the board for a reset before the flash download or the debug session. underReset: Asserts the hardware reset during the connection. preReset: Triggers a hardware reset pulse before the connection. running: Connects to the CPU without stopping the program execution during the connection.
Reset Mode	Controls the reset operations performed by the debugger. The options are: <ul style="list-style-type: none"> auto (default): The debugger decides which reset to use based on information from the CMSIS-Pack. system: This mode uses the ResetSystem sequence from the CMSIS-Pack. hardware: This mode uses the ResetHardware sequence from the CMSIS-Pack. processor: This mode uses the ResetProcessor sequence from the CMSIS-Pack.
Reset After Connect	Performs a reset operation as defined in Reset Mode after connecting to the board.
Verify Application	Compares the content of the target memory with the program loaded in the debugger. Enable this option to ensure that the image loaded in the target system matches the image loaded in the debugger. This prevents debugging the wrong code when working with various targets or more instances of Keil Studio.
Deploy First	If selected, a flash download is triggered when you start a debug session. To save connection time, the flash download only happens if the code has changed, the active project has changed, or the board has been disconnected and reconnected.

Flash download options in the Run category:

Setting	Description
Use Daplink	If selected, the DAPLink firmware does the flash download of programs to your board.
Erase Mode	The erase modes available if you are not using DAPLink. The options are: <ul style="list-style-type: none"> sectors (default): Erase only sectors to be programmed. full: Erase full chip. none: Skip flash erase step.
Program Flash	Flash program option if you are not using DAPLink. Writes code into the flash memory.
Verify Flash	Flash verify option if you are not using DAPLink. Verifies the content downloaded to the flash memory during the flash download.
Reset Run	Reset option if you are not using DAPLink. Triggers a hardware reset after the flash download.

Options to change how panels display in the user interface in the Debug category:

Setting	Description
Debug View Location	Controls where the Debug view displays in the user interface. The options are: <ul style="list-style-type: none"> left (default): The Debug view displays on the left-hand side. right: The Debug view displays on the right-hand side. bottom: The Debug view displays at the bottom of the screen.

Setting	Description
Internal Console Options	Controls when the Debug Console view displays. The options are: <ul style="list-style-type: none"> neverOpen: The Debug Console view never displays. openOnSessionStart (default): The Debug Console view displays whenever a debug session starts. openOnFirstSessionStart: The Debug Console view displays the first time a debug session is started only.
Open Debug	Controls when the Debug view displays. The options are: <ul style="list-style-type: none"> neverOpen: The Debug view never displays. openOnSessionStart (default): The Debug view displays whenever a debug session starts. openOnFirstSessionStart: The Debug view displays the first time a debug session is started only.

13.2.11 Switch to RAM debugging

This section explains how to switch from flash debugging to RAM debugging.

Before you begin

When debugging a program, the debugger loads instructions to the flash memory and data to the RAM memory. This is known as flash debugging and it is the default behavior in Keil Studio.

Keil Studio also supports RAM debugging, which is faster and does not require flash algorithms like flash debugging. With RAM debugging, everything gets loaded to the RAM.

No specific settings are needed to do RAM debugging in Keil Studio but your program must be configured in a certain way.

Look at the example provided below to find out how to configure your project.



Some settings used for flash debugging might affect how your RAM debugging program runs. Check the advanced settings available for the debugger in the Debug or Run categories in File > Settings > Open Preferences. Unselect Use Daplink, select Deploy First and make sure hardware is not selected for Reset Mode.

Procedure

1. Import the [Blinky example for the FRDM-K32L3A6 board from keil.arm.com](#) (Projects tab > Open in Keil Studio button) and set it as the active project (right-click the project and select Set Active Project).
2. Open the `Blinky.cprj` file included in the project.
Note: The Keil Studio Explorer view hides `.cprj` files by default. Go to File > Settings > Open Preferences and search for the Exclude preference. Open the `settings.json` file and check that `.cprj` files are not listed under `"files.exclude"`.
3. In the `target` section, go to this block of code:

```


...
<ldflags add="--callgraph --diag_suppress 6314 --entry=Reset_Handler --
info sizes --info summarysizes --info totals --info unused --info veneers

```

```
--load_addr_map_info --map --strict --summary_stderr --symbols --xref"
compiler="AC6" file="./RTE/Device/K32L3A60VPJ1A_cm4/K32L3A60xxx_cm4_flash.scf"/>
```

4. Replace it by this block to point at the K32L3A60xxx_cm4_ram.scf file in the project (this file is provided by default in the RTE > Device folder):

```
...
<ldflags add="--diag_suppress 6314 --strict --summary_stderr --info summariesizes
--map --load_addr_map_info --xref --callgraph --symbols --info sizes --info
totals --info unused --info veneers --entry=Reset_Handler" compiler="AC6"
file="./RTE/Device/K32L3A60VPJ1A_cm4/K32L3A60xxx_cm4_ram.scf"/>
```

5. Click the Debug project  button to start a debug session.
6. Check the debugging output in the Debug Console.

13.3 Use the Memory view

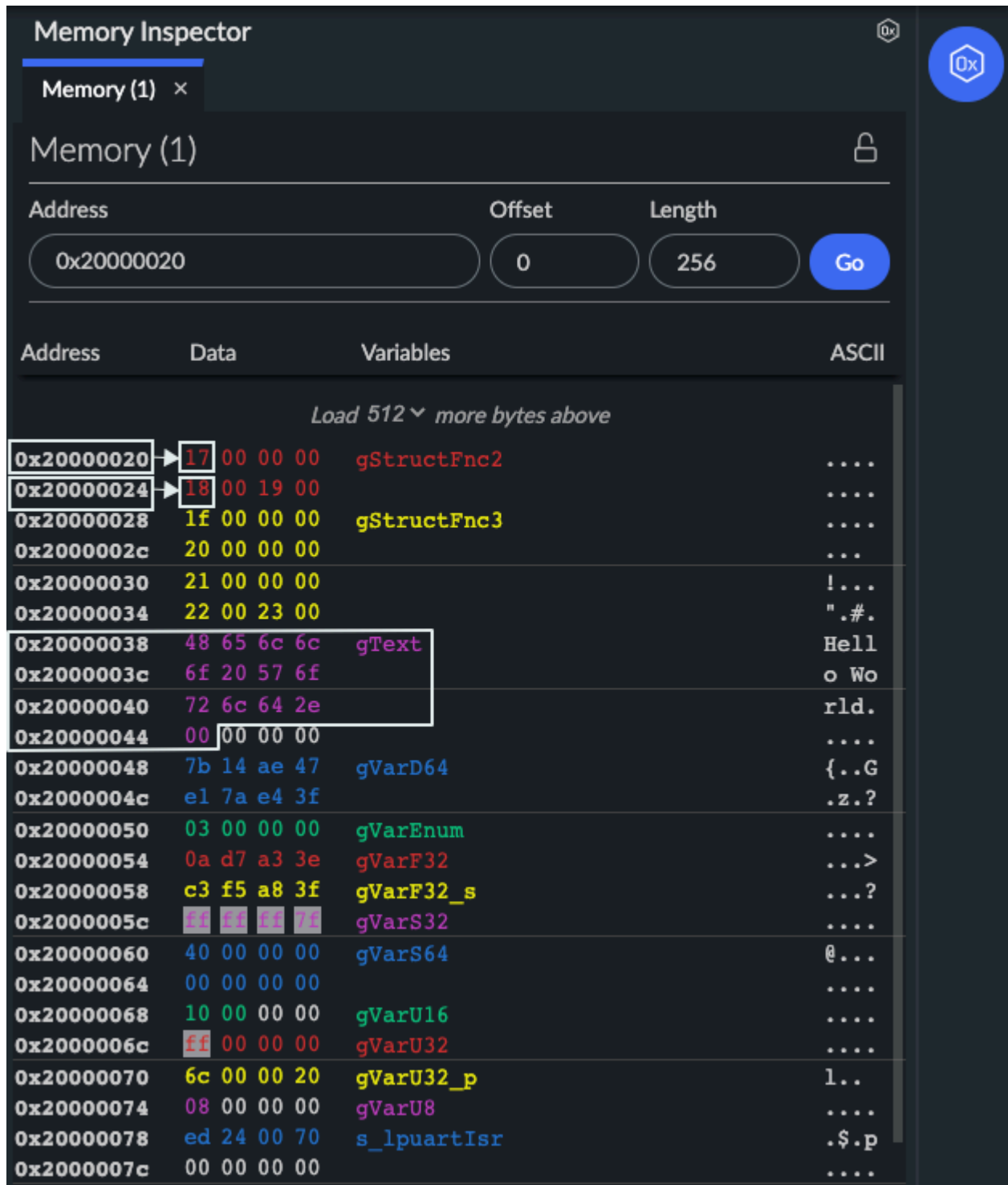
This section describes the Memory view and explains how to monitor the memory usage of your board as you step through the code of your project during a debug session.

The Memory view allows you to see the changes as they occur. You can also open multiple memory tabs to compare memory at different times or memory from different regions.

Interface

Here are the details you can see in the Memory view:

Figure 13-1: The 'Memory' view



- The Address column shows the addresses of memory locations (or cells in memory) in hexadecimal format. The addresses provided correspond to the first memory location on each line.

- The Data column shows the values stored in each memory location in hexadecimal format. When you hover over one memory location, the value stored is also available in binary, decimal, or UTF8 format.
- The Variables column shows the names of the variables stored in memory. Colors help you distinguish the memory locations used by variables. For example: `gText` uses 13 bytes in memory.
- The ASCII column shows the ASCII character equivalents of the values stored.


Monitor the memory usage of your board


You should first start a debug session and set breakpoints to be able to monitor the memory usage of your board.

To start a debug session and check the Memory view:

1. Set the project you want to debug as the active project and start a debug session as explained in [Debug first steps](#).

The debugger stops at the function “main” of your application. The Memory view opens with an empty Memory (1) tab. You can rename that tab.

You can click the Memory button  in the top right-hand corner of the screen to hide or display the view.

2. Set breakpoints on the parts of code you want to examine as explained in [Set breakpoints](#).
3. Click the Continue button .

The debugger runs to the first breakpoint it encounters and stops.

4. In the Variables list, find the variable you want to check.
5. Right-click the variable and select Show variable in memory.

Note that you can also directly type the address of a variable in the Address field and click the Go button. You can use pointers' addresses too.

The Memory view initially shows a 256-byte region that starts from the chosen address, with higher memory addresses at the bottom of the tab.

6. To navigate the memory locations, you can modify the Offset and Length.

For example, if the current address is `0x2000006c` and you type 2 in the Offset field, the Memory view displays a memory region that starts at `0x2000006e`. If you type 64 (bytes) in the Length field, the Memory view displays 64 bytes from the current memory location, so 4-byte values per line * 16 lines.

Note that you can also display more lines using the Load more bytes above and Load more bytes below drop-down lists.

7. Step through the code and observe the changes.



Memory tabs are dynamic and update as you step through the code. Memory locations that have changed from one frame to the next are highlighted.

You can write a new value to any memory location displayed by typing over the value shown in the Data column and clicking the Apply Changes button. Updated memory values are highlighted with a yellow square.

Compare memory locations

You can open multiple memory tabs to compare memory at different times or memory from different regions. This is particularly helpful when you are copying or moving data from one region to another and need to verify an operation is occurring as it should.

To compare memory locations:

1. Follow the steps in [Monitor the memory usage of your board](#) and look for the address you want to check.
2. Click the Create new memory inspector  icon to open a new memory tab and look for the same address (to compare memory at different times) or a different address (to compare memory from different regions).
3. You can click the Freeze memory view  icon on a tab to freeze the data displayed.
4. Drag and drop the new tab next to the first tab.
5. Step through the code and observe the changes.

To visually compare memory locations easily, you can also display a diff view:

1. Click the Toggle Comparison Widget Visibility  icon.

A comparison panel opens at the bottom of the screen.

2. If you have more than two tabs open, select the tabs you want to compare in the drop-down lists and click Go.

Note that you must load memory in both tabs to be able to display a diff. The content of the tabs is displayed side by side and differences are highlighted in red (before) and green (after), as in a diff view with Git.

14. Supported and custom targets

This chapter describes the hardware that Keil Studio supports and how to set up custom targets.

14.1 Supported development boards

This section describes the development boards that Keil studio supports.

For CMSIS projects, Keil Studio supports the following [development boards](#).

For Mbed projects, Keil Studio supports the following [development boards](#).

14.2 Supported debug probes

This section describes the debug probes that Keil studio supports.

WebUSB-enabled CMSIS-DAP debug probes

For DAPLink: see the [ARMmbed/DAPLink](#) repository for more information.

For ULINKplus (firmware version 2): see the [Keil MDK](#) documentation.

For other WebUSB-enabled CMSIS-DAP debug probes: see the [CMSIS-DAP](#) documentation.

ST-LINK debug probes

Keil Studio supports ST-LINK/V2 probes and later, and the ST-LINK firmware available for these probes.

The recommended debug implementation versions of the ST-LINK firmware are:

- For ST-LINK/V2 and ST-LINK/V2-1 probes: J36 and later.
- For STLINK-V3 probes: J6 and later.

See “Firmware naming rules” in [Overview of ST-LINK derivatives](#) for more details on naming conventions.



Keil Studio notifies you when older firmware versions are detected but does not stop you from using them. Visit the [STSW-LINK007](#) page to stay up-to-date with the latest firmware versions.

14.3 Custom targets

A build target (or target hardware) is a destination for a software build, for example an MCU, development board or custom board. A build is always done for a specific target hardware.

Keil Studio has a list of build targets for all compatible boards. Sometimes, you might want to create a custom target. For example, you might have a debug probe connected to a board that does not have an integrated interface chip. When you plug in a debug probe, Keil Studio needs to know the target to build and debug for, so you must specify it.

A custom target is tied to an individual board or debug probe (based on its USB Serial Number). Every time you plug a custom target in, Keil Studio automatically chooses the custom target you set up for it.



Flashing and debugging in an incompatible way (for example, if you select an incorrect build target) can damage your board and make it unusable.

14.3.1 Set up a custom target

This section describes how to set up a custom target in Keil Studio.

Procedure

1. Connect your hardware over USB. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests a matching target. If your device is a debug probe, or for Mbed projects, if your board is not supported by the Mbed OS version for your project, the Manage Custom Targets dialog box opens automatically.

If your device is a development board, open the Target hardware drop-down list and click the


Manage custom targets button .

2. The Manage Custom Targets dialog box shows all USB-connected devices in the USB device drop-down list. Select the one you want to create a custom target for.
3. In the Build target drop-down list, select a build target. A build target is the target hardware (device) you build the project for. Selecting a build target populates the Target name field with a matching value, unless you have already manually edited this field.
4. Click the Save All button.

14.3.2 Reset a target

This section describes how to reset a target in Keil Studio.

Procedure

1. Open the Manage Custom Targets dialog box.
2. Click the Reset custom target button .

15. Extensions


Keil Studio supports a fixed list of Visual Studio Code compatible extensions to take your programming experience to the next level. Extensions can add support for additional programming languages, integrate with cloud service providers, or help configure your MCU.

The AWS Toolkit is the first extension made available to you to support your Amazon Web Services IoT workflows.

15.1 Install the AWS Toolkit extension

The AWS Toolkit extension is available from the Extensions view.

Procedure

1. To open the Extensions view, click the Extensions icon  in the Keil Studio activity bar. The AWS Toolkit extension is ready to be installed.
2. Switch on the toggle button to install the extension. A popup message appears to indicate that the Toolkit collects metrics.
3. Click OK in the popup message.

Once the AWS Toolkit extension has been installed, an AWS icon  displays in the activity bar.

4. Click the AWS icon. This opens the AWS Explorer and CDK Explorer. Another popup message appears to indicate which version of the Toolkit has been installed. Click the View Quick Start button in the popup message to check the Readme available and learn about the AWS Toolkit. Next, follow the steps below to [Connect to AWS from Keil Studio](#).

Notes:

- The Readme is also available when you click the extension from the Extensions view.
- To uninstall the AWS Toolkit extension, switch off the toggle button from the Extensions view.

15.2 Connect to AWS from Keil Studio

Create an AWS account, if you do not already have one, and configure credentials to access AWS services and resources from Keil Studio.

Before you begin

To create an account, see [How do I create and activate a new AWS account?](#).

To access AWS services and resources, you need an access key. An access key consists of an access key ID and a secret access key. This access key is used to sign programmatic requests that you make to AWS. See [Obtaining AWS access keys](#) to create an access key.

Procedure

1. Open the AWS Toolkit extension in Keil Studio and go to the AWS Explorer.
A “Connect to AWS...” message displays.
2. Click the message.
A popup message appears to indicate that no credentials were found and invite you to create credentials.
3. Click Yes to create credentials.
A popup appears and asks you for a credential profile name.
4. Type a profile name and press Enter.
A popup appears and asks you for an AWS Access Key (this is the access key ID and looks like this: `AKIAIOSFODNN7EXAMPLE`).
5. Provide your access key and press Enter.
A popup appears and asks you for an AWS Secret Key (this is the secret access key and looks like this: `wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY`).
6. Provide your secret key and press Enter.
A popup appears and asks you if you want to show the default region your profile is associated with in the AWS Explorer.
7. Click Yes.
The region is added to the AWS Explorer. You can now start using the AWS services and resources available. Next, try out our [AWS MQTT Demo tutorial](#) to understand how to connect your device to AWS cloud services.

16. Known issues and troubleshooting

This chapter describes Keil Studio known issues, how to troubleshoot some common issues, and how to contact Arm to report issues or suggest enhancements.

16.1 Known issues

This section describes the known issues in Keil Studio.

Keil Studio has the following known issues:

- Projects that you download from the Online Compiler as ZIP files contain a cut-down version of Mbed OS specific to the selected target. To select a different target, delete the Mbed OS library and replace it with a full copy of the library.
- Some users have reported licensing issues for Arm Compiler 6. These issues might be related to the system clock and region format.
- On Linux, there is a limit on the number of files that can be watched in the workspace. To increase the limit, follow the instructions [on the Visual Studio documentation site](#).
- On Linux, source control management requires the `libcurl` package to work correctly.

16.2 Troubleshooting

Provides solutions to some common issues you might experience when you use Keil Studio.

16.2.1 Keil Studio does not load

Keil Studio hangs on the loading screen.

Solution

Keil Studio takes longer to load when you have many tabs opened in your browser and change tabs while Keil Studio is loading. Stay on the tab where Keil Studio is loading to solve the issue.

16.2.2 Cannot log into Keil Studio

You are unable to log into Keil Studio.

Solution

Ensure that your Arm account or Mbed account credentials are correct.

16.2.3 Connected development board or debug probe not found

You have connected your development board or debug probe, but Keil Studio cannot detect the device.

Solution

- Run Device Manager (Windows), System Information (Mac), or a Linux system utility tool like `hardinfo` (Linux) and check for warnings beside your devices. Warnings can indicate that device drivers are not installed. If necessary, obtain and install the appropriate drivers for your device.
- Check that the firmware version of your board or debug probe is supported and update the firmware to the latest version. See the [Out-of-date firmware](#) section for more details.

16.2.4 Out-of-date firmware

You have connected your development board or debug probe and a popup message appears mentioning that the firmware is out of date.

Solution

Update the firmware of the board or debug probe to the latest version:

- [DAPLink](#). If you cannot find your board or probe on daplink.io, then check your device manufacturer's website.
- [ST-LINK](#). Note that ST development boards and probes on Windows require extra drivers. You can [download them from the ST site](#).
- For other WebUSB-enabled CMSIS-DAP firmware updates, please contact your board or debug probe vendor.



If you are using an FRDM-KL25Z board and the standard DAPLink firmware update procedure does not work, follow this [procedure](#) (requires Windows 7 or Windows XP).

16.2.5 Connection fails when clicking Run project or Debug project

Keil Studio has correctly detected your development board or debug probe and the device shows as active, but the connection fails when you click Run project or Debug project. There can be multiple reasons for a connection failure.

Solution: check your hardware setup

Cables:

- Check that all USB cables and power cables are connected correctly.
- Check that no cable is damaged. Note that low-quality cables can also impact the debug connectivity.

- Disconnect and reconnect the board or the debug probe, or both, to recover from an unexpected state.

Jumpers and switches:

Check that all jumpers and switches are configured correctly. Development boards often have different jumpers and switches that can be set for specific use cases.

For example:

- Jumpers to select or connect to the power supply (for example, there can be different USB connectors or external power supply units).
- Multiple jumpers to enable, connect, or configure different subsystems (core, peripherals, GPIO pad) or allow different voltage levels.
- Jumpers to enable and disable, or configure connections to other on-board chips like code flash devices (for example, address lines).
- Jumpers that connect and disconnect the on-board debugger to and from the target device.
- Switches to configure the target device boot mode (for example, to select the code memory device to boot from).
- On and off switch or switches (for example, if the on-board debug probe is continuously powered after plugging in the board, but the target system has to be turned on separately).

Firmware:

Check that the firmware version of your board or debug probe is supported and update the firmware to the latest version. See the [Out-of-date firmware](#) section for more details.

External debug probe:

- Check that the debug probe is connected to the correct debug connector on the board. Some boards have multiple debug connectors, for example, one to debug the target device and one to debug an on-board debugger.
- Check that the board is powered and turned on. An external debug probe can be successfully detected even if the attached target system is not connected.
- Check that there is no conflict with an on-board debugger:
 - Check if you have to change the jumper settings to disconnect the on-board debugger.
 - Check if you have to change a jumper to connect the external debug probe to the target device.
 - Check if the board is powered through the USB port for the on-board debugger. If that is the case, try to choose a different power source for the board.

Solution: check the connect mode or target device boot mode

Problems can happen with code that was flashed to your device using a specific build target. The code can be at the origin of some debug connectivity issues.

Problems can occur when:

- One or more device-specific low-power modes are used which affect the debug connectivity. It is the case for Mbed projects running on ST boards, for example.
- Watchdog timers or other device-specific peripherals that reset the system and the debug interface are activated.
- Reconfigured GPIO pins are required for a debug connection (SWD/JTAG pins).
- Faulty code runs and puts the device in a lockup state, and a debug/flash download connection cannot be established as a result.

Possible solutions to overcome this are:

- Change the debug Connect Mode preference to underReset to keep the device or the CPU or both in reset mode during the debug connection.
- If the target device does not support underReset connections, then change the connect mode to preReset. Depending on the project that is running, preReset can delay the execution of problematic parts of code long enough to establish a debug connection. The success of that depends however on timings in the program running on your target, your debug unit, and the host computer. For example, a fast debug unit has a higher chance to connect than a slow debug unit.
- Boot the target device in a different boot mode. It can require changing on-board switches or jumpers or both and disconnecting and reconnecting the board. For some boards, you can change the boot mode to a system boot mode (for example, you can do that with the NXP i.MX RT boards).

If the previous solutions do not work, try alternative methods to flash download your project without the code triggering debug connectivity problems:

- If you are using a DAPLink debug probe, activate the Use Daplink option in the Run category in the preferences.
- If your board allows programming with a mass storage device (MSD) or a serial connection, then download the built program from Keil Studio to your computer and use this alternative way of programming.

16.2.6 Development board not showing serial data

Serial data for a development board is not being shown in Keil studio.

Solution

- Disconnect and reconnect the board.
- Check that no other project is accessing the serial data on the board; the device can only connect to one serial monitor at a time.
- On Linux, ensure that the current user belongs to the `dialout` group. To check that the current user belongs to the `dialout` group, run the command: `sudo adduser "$USER" dialout`.

16.2.7 Cannot make an Mbed project the Active Project

If there is no Mbed OS library imported in a project, then you cannot set the project as the Active Project. A popup warning message such as “Cannot set <project-name > as Active Project as it is not recognized as a valid project.” displays.

Solution

In the Explorer view, check if you have an Mbed OS library imported to the project, but with a directory name that is not `mbed-os`.

- If you find an Mbed OS library directory with a directory name that is not `mbed-os` (in other words, a custom name), rename the directory as `mbed-os`.
- If you do not find an Mbed OS library directory with a custom name, you must add an Mbed library to that project:
 1. Right-click on the project name and select Add Mbed Library.
 2. Provide the URL to import the Mbed OS library from.

If you import your Mbed OS library from <https://github.com/ARMmbed/mbed-os/>, the Library Name pre-populates as `mbed-os`. Do not change the pre-populated `mbed-os` name.

If you import your Mbed OS library from another URL, ensure the Library Name given becomes `mbed-os`.

3. Click Next.
4. Select the branch or tag to import for your Mbed OS library.
5. Click Finish.

Your library starts to import. When the import completes, you can set the project as the Active Project.

16.2.8 Code editor shows many red squiggly lines in an Mbed project

Your project must contain an `mbed-os` directory. Without an `mbed-os` directory, Keil Studio shows many red squiggly lines where something in your code (header files, types, and similar) requires an Mbed OS library component.

Solution

1. Open the Mbed Libraries view, select View > Mbed Libraries.
2. Check that the Mbed OS library for your project is there. To the left of each library that is listed, there is a drop-down button which you can select to see where the library was imported from.
 - a. If there is not an imported Mbed OS library listed, you must add an Mbed OS library to your project. To learn how to add and manage Mbed libraries, see [Manage an Mbed project and its libraries](#).

- b. If there is an imported Mbed OS library listed, but it does not have the name `mbed-os`, you must rename the library directory to be `mbed-os`. Go to the Explorer view, find the library to rename, right-click on that library directory, and select Rename.

16.2.9 Linker failing with file not found for Mbed OS 15.4.0 and older

Mbed OS 15.4.0 and older do not support spaces in project names. The linker fails and states "file not found".

Solution

Change your project name and try to compile again.

16.3 Report an issue or suggest an enhancement

Help us improve Keil Studio.

Report any issues you experience, or suggest enhancements on the forums:

1. From the Help menu, select Report an Issue. The Report an Issue or Provide Feedback dialog box opens.
2. Click Report Issue, which takes you to the Keil forum.

You can also access the [Keil forum](#) and [Mbed forum](#), through the Help > Forums menu selection.