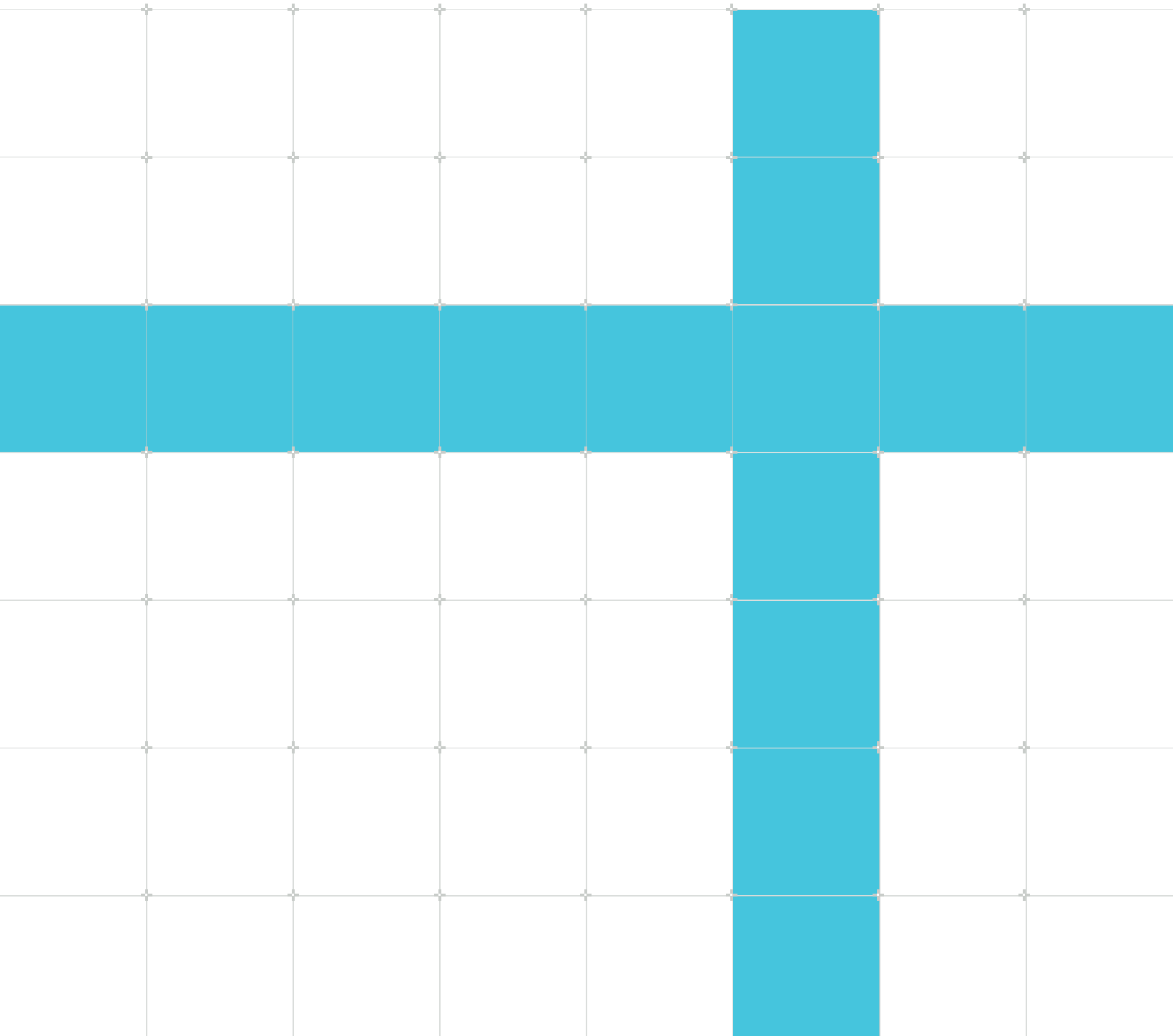
Arm Keil Studio Cloud

1.5

User Guide

Issue 06
102497_1.5_06_en



Arm Keil Studio Cloud

User Guide

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

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web address

developer.arm.com

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

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
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
`monospace underline`	Denotes 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: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
	This represents a recommendation which, if not followed, might lead to system failure or damage.
	This represents a requirement for the system that, if not followed, might result in system failure or damage.
	This represents a requirement for the system that, if not followed, will result in system failure or damage.

Convention	Use
 Note	This represents an important piece of information that needs your attention.
 Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
 Remember	This is a reminder of something important that relates to the information you are reading.

1.2 Feedback

Arm welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title Arm Keil Studio Cloud User Guide.
- The number 102497_1.5_06_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.


Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

1.3 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 µVision. 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 section 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 Create or import a project 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 in the User Profile view.

4 Tutorials

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

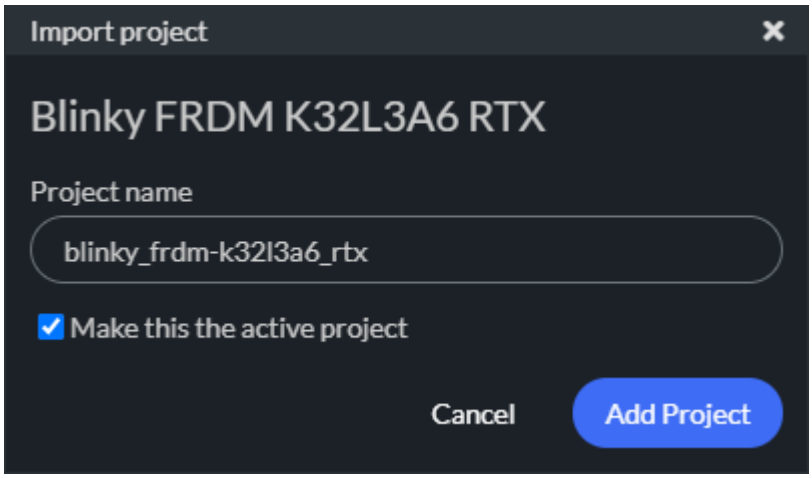
4.1 Get started with a CMSIS Blinky example

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

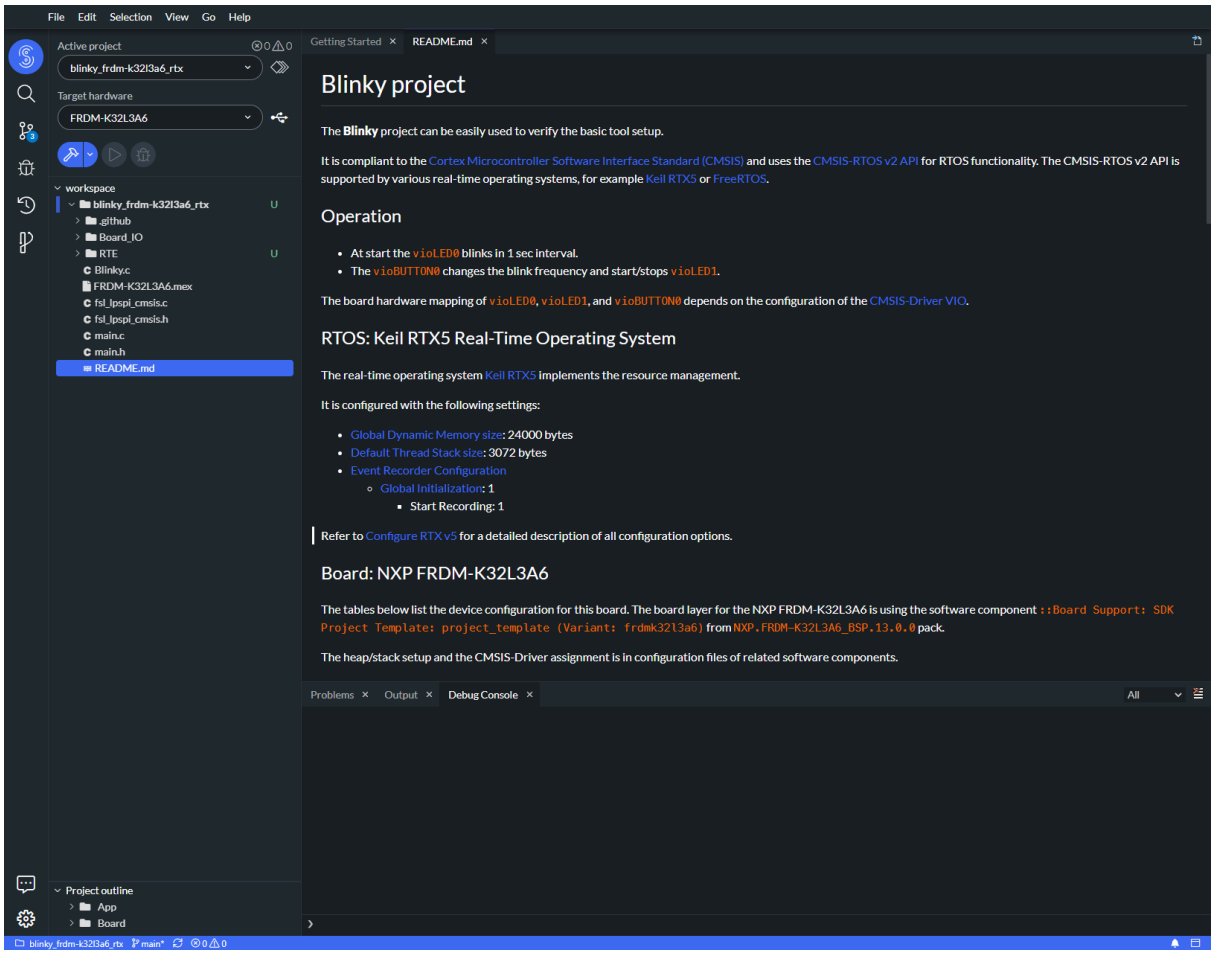
1. Go to keil.arm.com.
2. Click the Hardware menu to see the list of supported hardware and select the Only show boards with example projects checkbox to find examples.
3. Click the FRDM-K32L3A6 board in the list.
   You can either download or import example projects into Keil Studio from the Projects 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 4-1: Import project



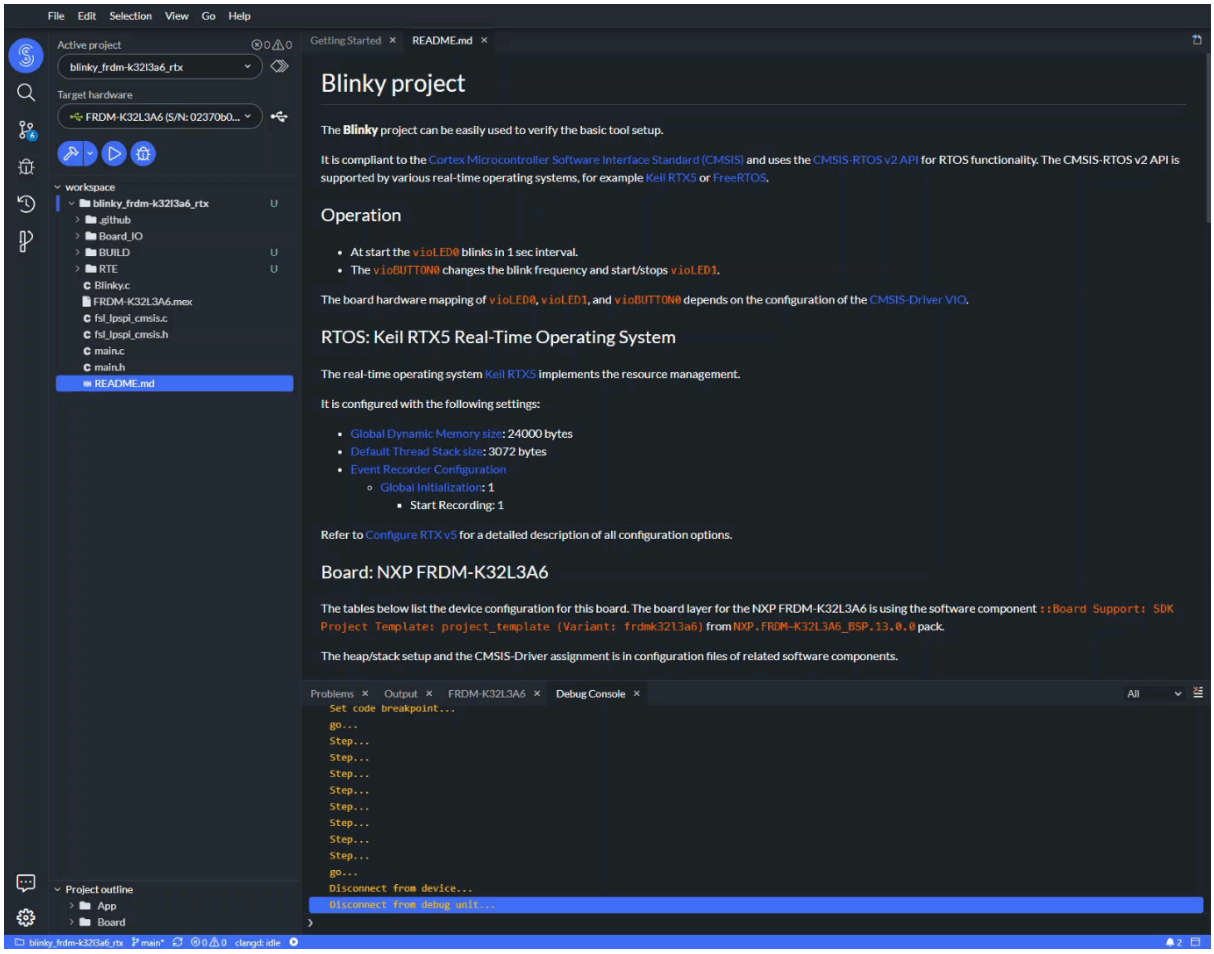
7. Click Add Project.
   The project loads to your workspace and is the active project. The `README.md` file of the project displays. Review the file to learn more about project settings and board requirements.

Figure 4-2: Open project showing README.md



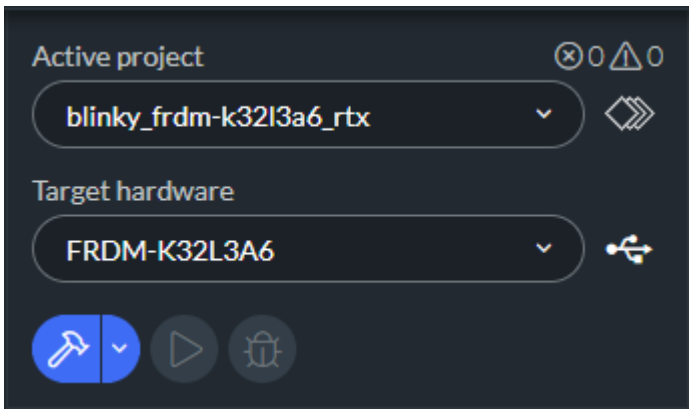
8. Inspect the software components that are used in the project and access their documentation:

Figure 4-3: Manage software components



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

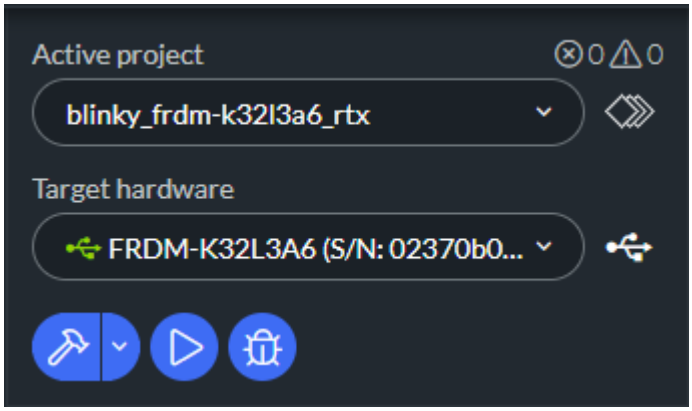
Figure 4-4: Target hardware



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 4-5: 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.

4.2 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 Project....
   The New project dialog box opens.
2. Click the Example project drop-down list and go to the Mbed list. 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.

4.3 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.

4.3.1 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

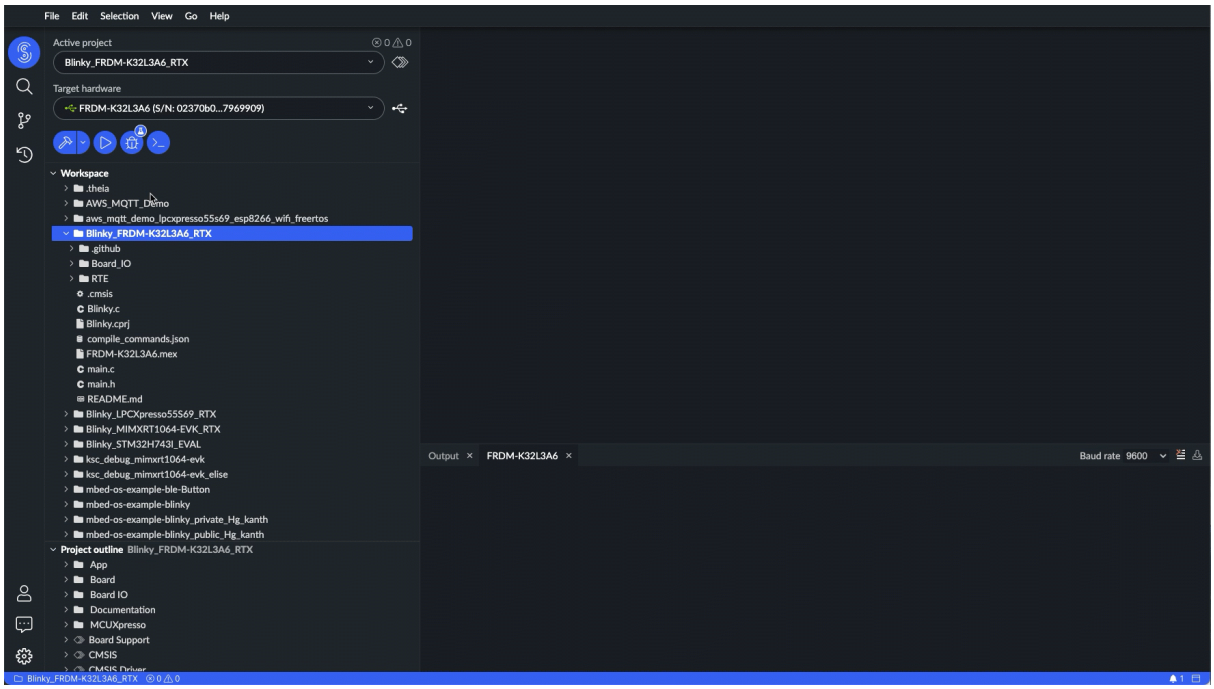
1. You need a GitHub account for this tutorial. Create an account on github.com or, if you already have an account, sign in. Once this is done, check that your Arm account and your GitHub account are connected as described in Set credentials for GitHub.
2. Check that you have created a `Blinky_FRDM-K32L3A6_RTX` 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 4-6: Configure your project for source control



4.3.2 Use source control and publish your changes

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

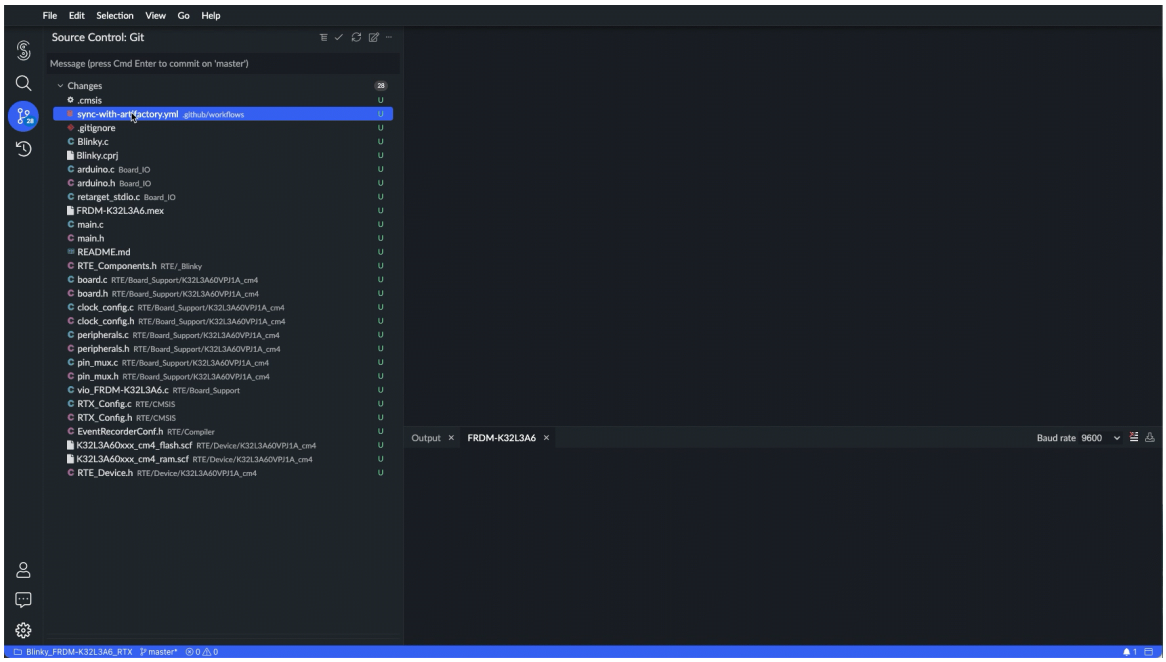
Procedure

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 4-7: Create a first commit

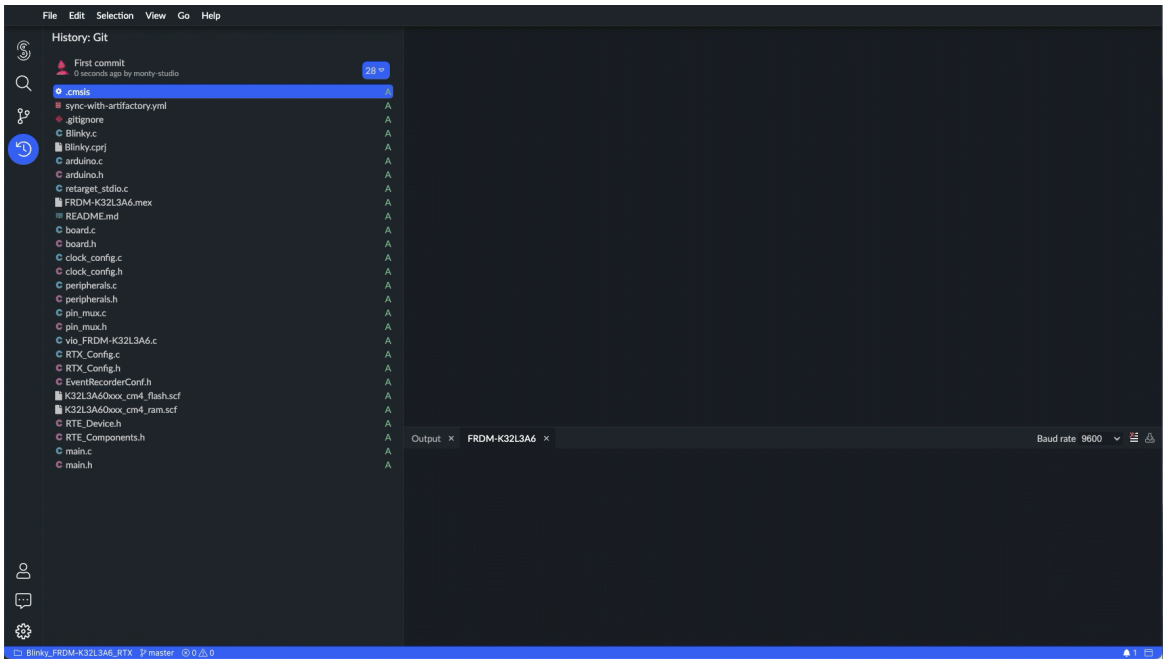


2. Create a working branch:
   a. Click the master branch in the status bar.

      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`.

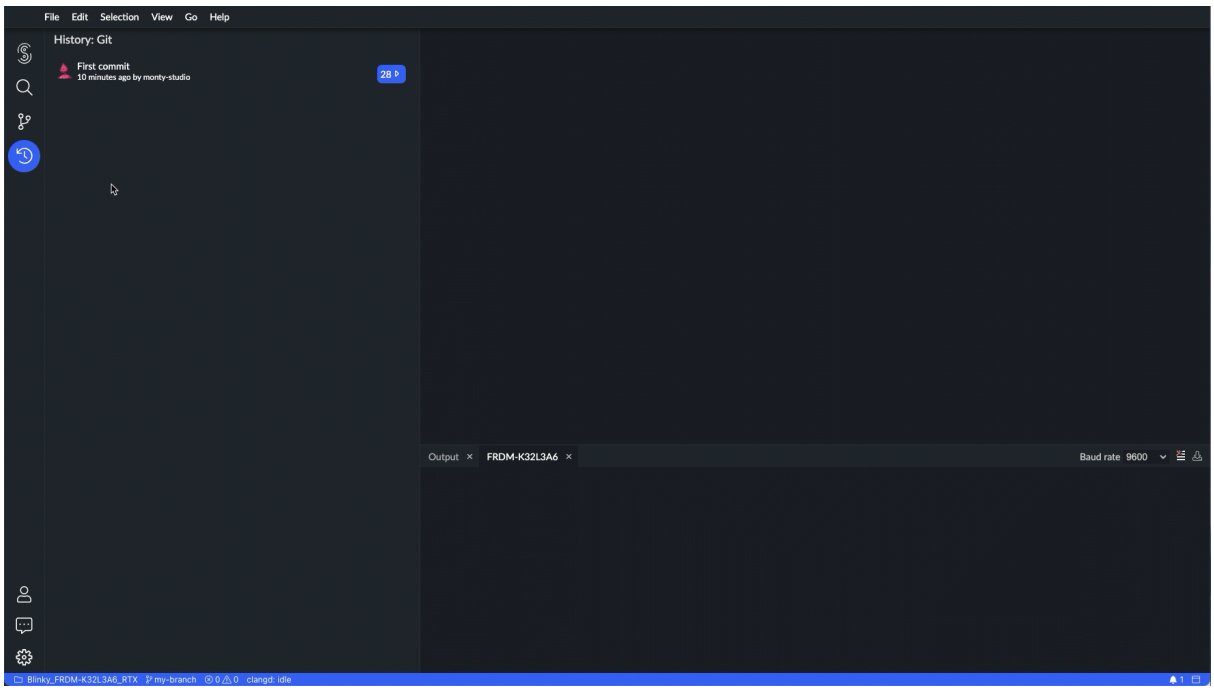
      Figure 4-8: Create a working branch



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.

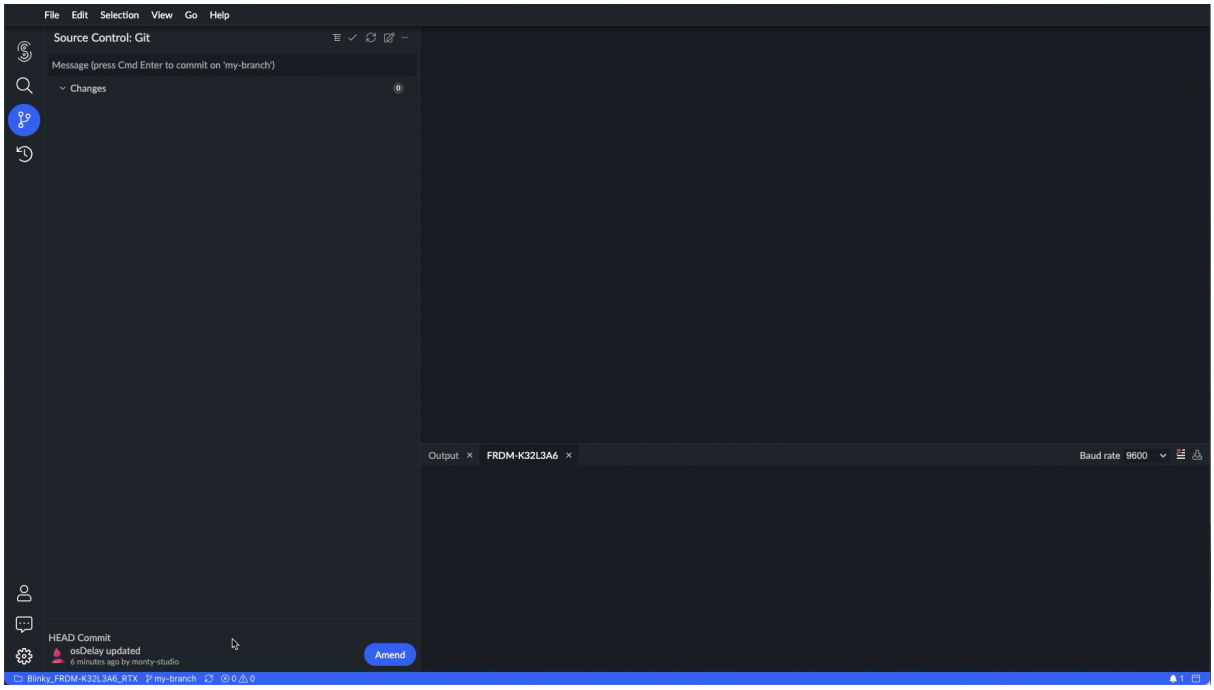
Figure 4-9: Stage, commit, and push your changes



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 4-10: Merge your changes into master



4.4 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.

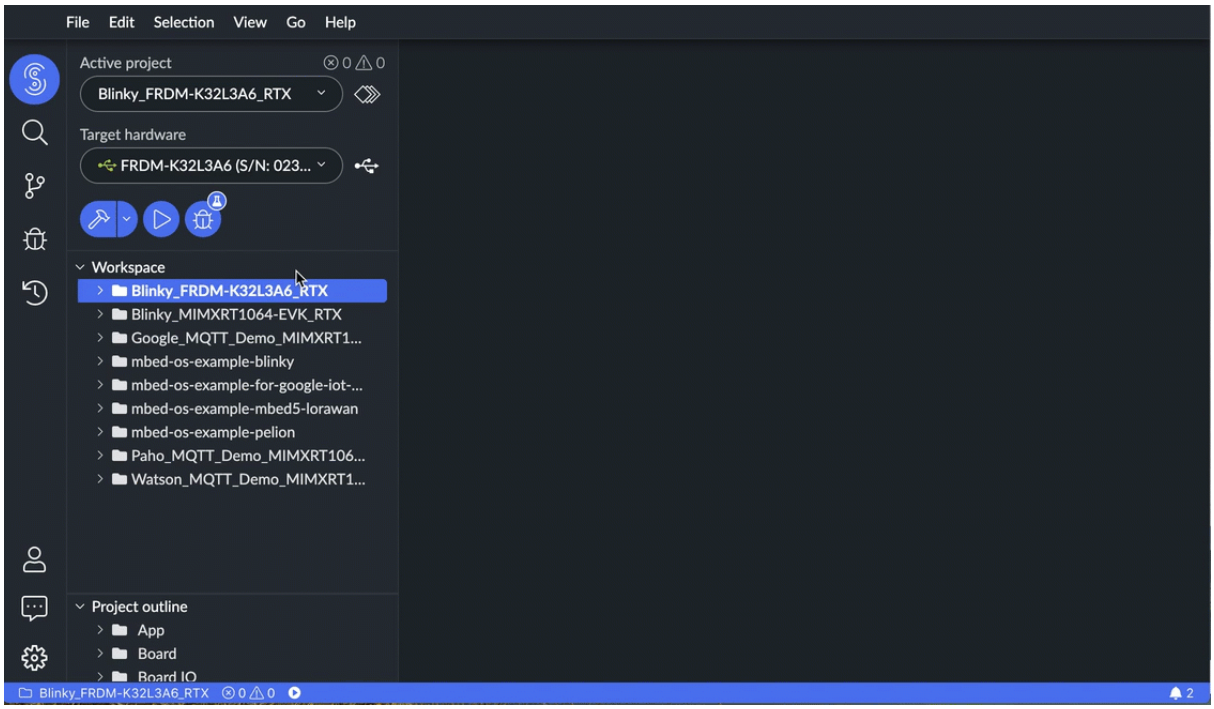
Before you begin

1. Check that you have imported the `Blinky_FRDM-K32L3A6_RTX` example and set it as the active project (right-click the project and select Set Active Project).
2. 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.
3. 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()`.

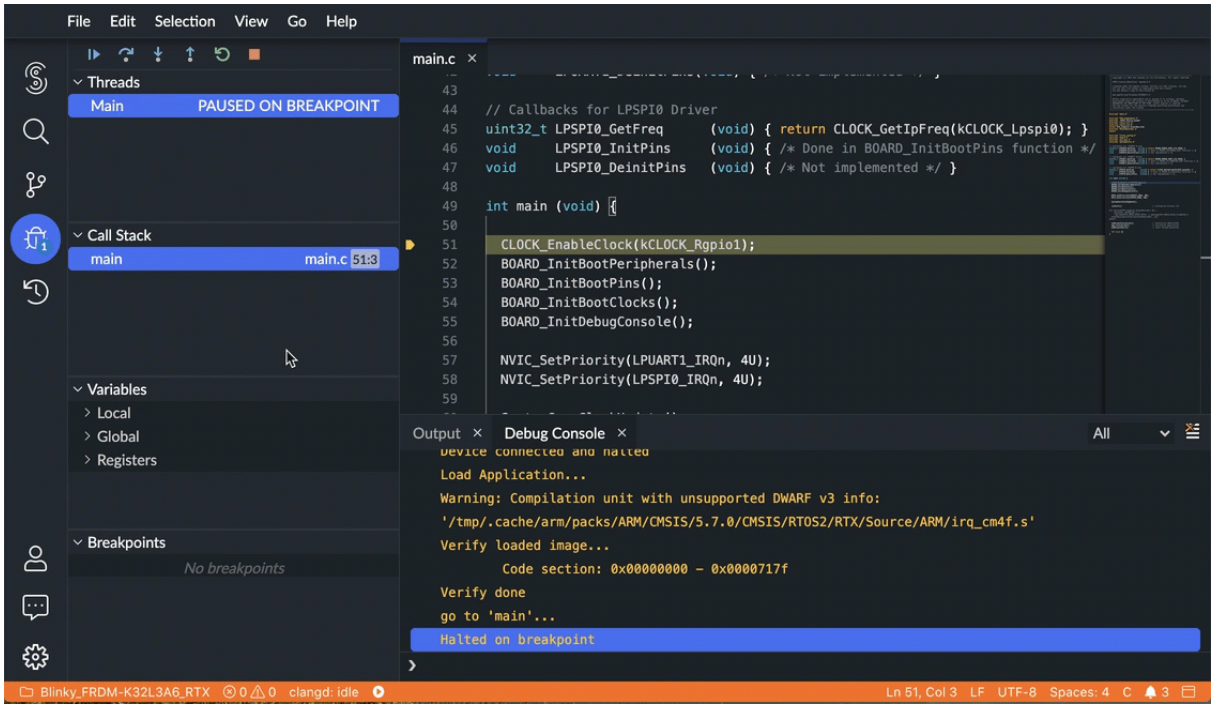
Figure 4-11: Start a debug session



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 `thrLED: 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.

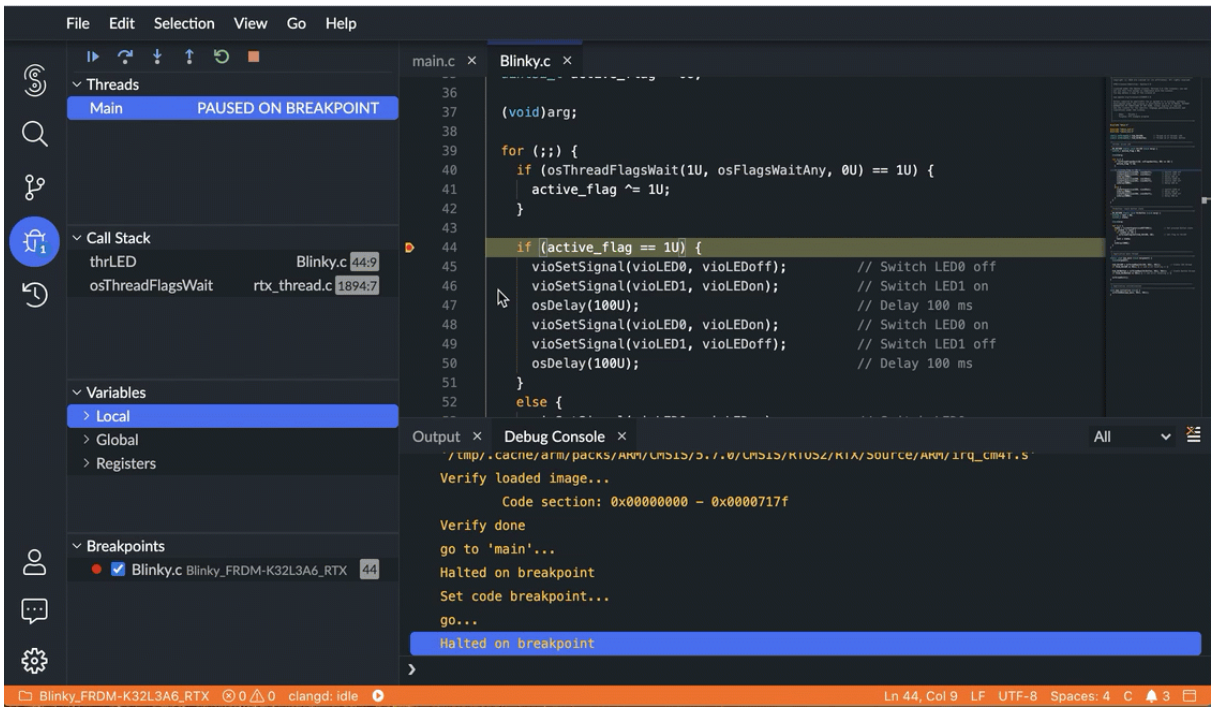
Figure 4-12: Set a breakpoint



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 4-13: Check variables

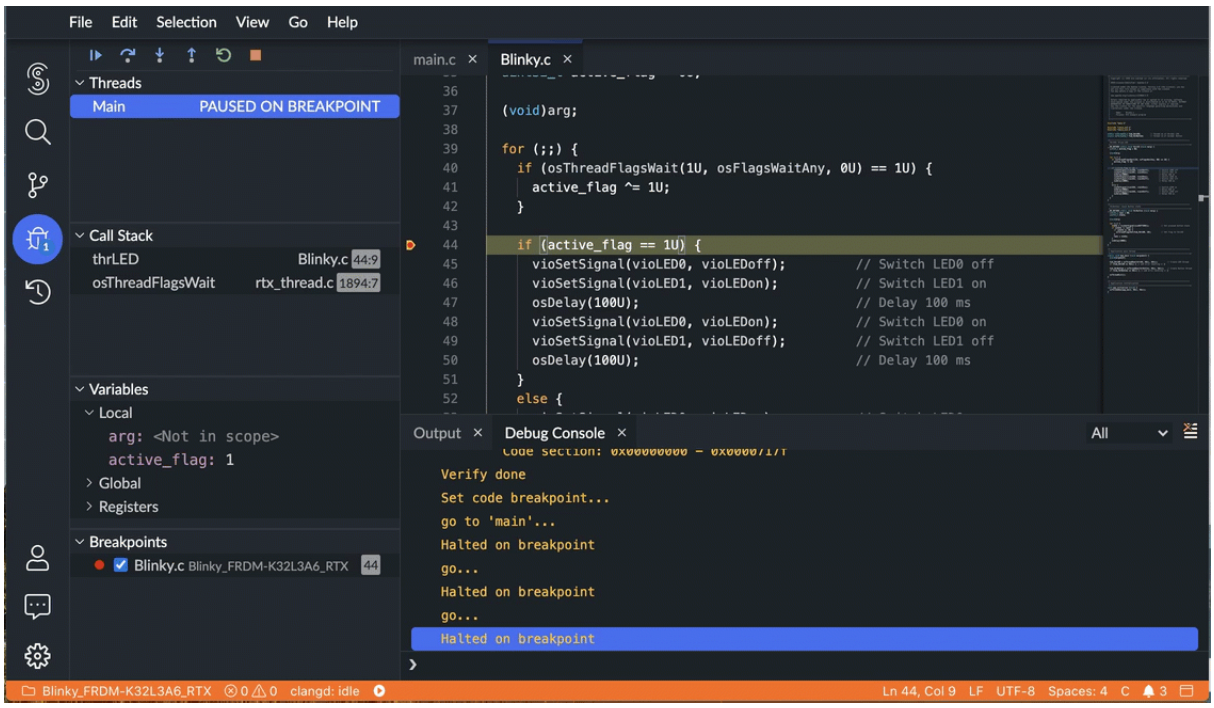


9. Click the Step Over button  three times to go straight to `osDelay(100U);`.
10. Click the Step Into button  to check this function.
    This opens the `rtx_delay.c` file where the function is called.
11. You can now examine what happens in the Threads and Call Stack lists.
    In the current version 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 4-14: Examine the call stack



5 Create or import a project

This section describes how to create CMSIS or Mbed projects, or import existing Mbed projects, in Keil Studio.

For CMSIS projects, you can:

- Create a new project from an example shipped with Keil Studio.
- Create a new project from an example available on keil.arm.com.

Arm recommends that you use the example projects provided. The examples use predefined `.cprj` templates. For more information on CMSIS-Build and the CMSIS Project description format (`.cprj`), see the Build section of the CMSIS documentation.

For Mbed projects, 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 existing Mbed projects from:

- Your local file system.
- os.mbed.com/code (as a URL) or a Git hosting service (GitHub).
- Your Mbed Online Compiler workspace.

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.

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.

5.1 Your Keil Studio workspace

Keil Studio provides you with a cloud-hosted workspace for your code. Your workspace is the location that contains all your CMSIS and Mbed projects.

5.2 Create a project from a CMSIS or Mbed example project or an empty Mbed project

You can create a CMSIS project from an example, or 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 Project....

   The New project dialog box opens.
2. Click the Example project drop-down list and select an example from the CMSIS or the Mbed 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), go to the Mbed list and 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.

5.3 Create a project from a CMSIS example project from the Keil Studio website

You can create a CMSIS project from an example imported from keil.arm.com.

Procedure

1. Go to keil.arm.com.
2. Click the Hardware menu to see the list of supported hardware and select the Only show boards with example projects checkbox to find examples.
3. 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.

5.4 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 Git or os.mbed.com 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.

5.5 Import an Mbed project

There are some important points to consider when you import an Mbed project, for example, whether the project has all the required libraries and default configuration files. This topic describes how to import an Mbed project, and what to consider when you import.

Notes about importing Mbed projects:

- If you import 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.
- Imported 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.

Import an Mbed project from a URL

You can import from public repositories without setting up your Git credentials. If you want to import from a private repository on Git, set up Git credentials. Mercurial uses your Arm account (or Mbed account) credentials, so you have access to the same Mercurial repositories you do on os.mbed.com.

To import a project from a URL:

1. Open the File menu and select Import Project....
2. Paste the full HTTPS or SSH URL of the relevant web page and (optionally) edit the project name.

   The relevant web page can be a page listed on os.mbed.com/code or from another Git hosting service.

   Keil Studio sets the newly imported 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 imports the project with the version of Mbed OS it was originally created with.

When you import a project from os.mbed.com or a Git hosting service, Keil Studio clones the repository. You can push changes back to the remote repository from Keil Studio. For more information, see Source control.

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.

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

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.

To import copies of your Mbed projects in Keil Studio:

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.

Upgrade from Mbed 2 to Mbed OS 5 or 6

The main steps are as follows:

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

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.

Upgrade example:

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

1. Select File > New Project....
2. Click the Example project drop-down list and open the Mbed 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.bld` file and select Delete.

   Note: If you cannot see the `mbed.bld` 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 `"**/*.bld"` 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 Git or os.mbed.com 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.

5.6 Further resources about creating and importing projects

You are now ready to work on your project. You can learn more about CMSIS and find out more about Mbed OS APIs and programming on the Mbed OS documentation site.

When you are ready, build and run your project.

6 Build and run a project

Describes how to build and run a project with Keil Studio.

6.1 Select a build target

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

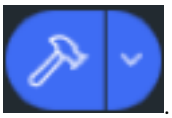
- To select a build target, either:
  - Connect your board to your computer. The first time you connect your board, you have to click the Connect to target hardware button to the right of the Target hardware drop-down list. After the first successful connection, Keil Studio detects the board and suggests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

6.2 Build and flash a project to the board

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.

6.3 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 6-1: The error text is a link to the relevant line of code

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

6.4 Configuring compile-time customizations

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.

7 Search

Keil Studio can search the content of files, quickly find a specific file, or find and execute commands. This section describes searching in Keil Studio in more detail.

7.1 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.

7.1.1 Search the content of multiple files

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 7-1: 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.

7.1.2 Search the content of a specific file

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 7-2: 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 .

7.1.3 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`.

7.2 Find and navigate to a file

Describes how to quickly search for and open a file by name.

Procedure

- Press Ctrl+P (Windows) or Cmd+P (macOS). Recently opened files appear first automatically, then file results are based on the string you enter in the search bar.

7.3 Find and execute a command

Describes how to quickly search for and execute any command in Keil Studio.

Procedure

- Select Find Command... from the View menu or press Ctrl+Shift+P (Windows) or Cmd+Shift+P (macOS). Recently-used commands appear when you open the quick command search. Search results appear when you begin to enter a string in the search bar.

8 IntelliSense

In this section, 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.

8.1 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.

8.1.1 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

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.

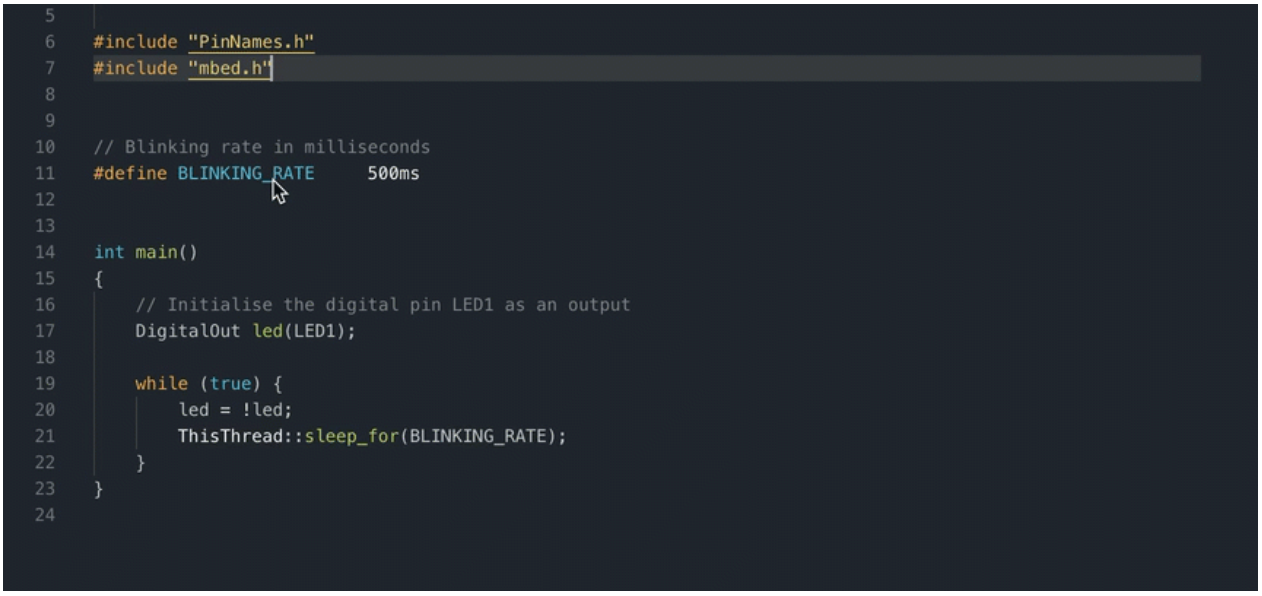
Bracket and quote highlighting

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 8-1: Additional information available when you hover over the code



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 8-2: Declaration presented inline with the Peek Declaration option

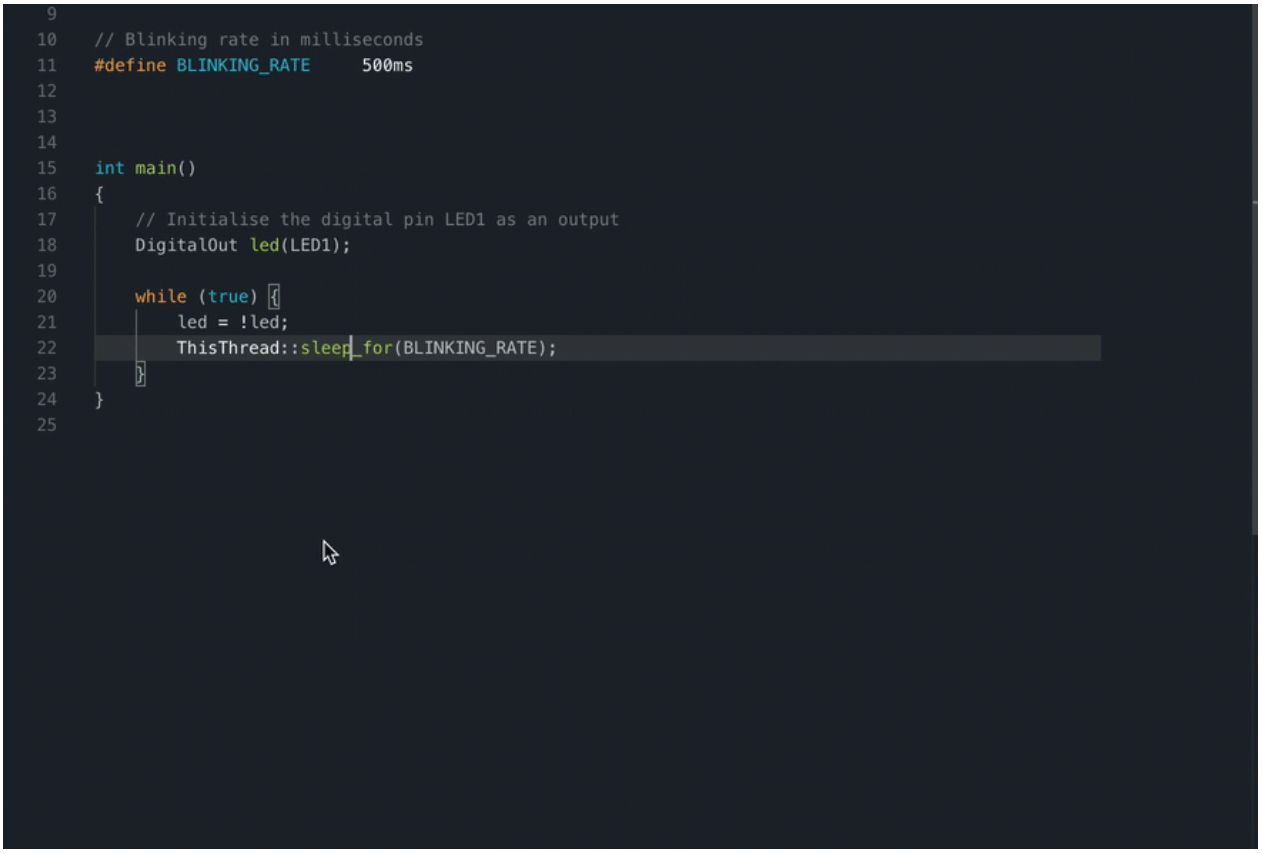
Go to Definition and Peek Definition

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 8-3: Definition presented inline with the Peek Definition option



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 8-4: You can edit directly in the inline editor

Go to Symbol

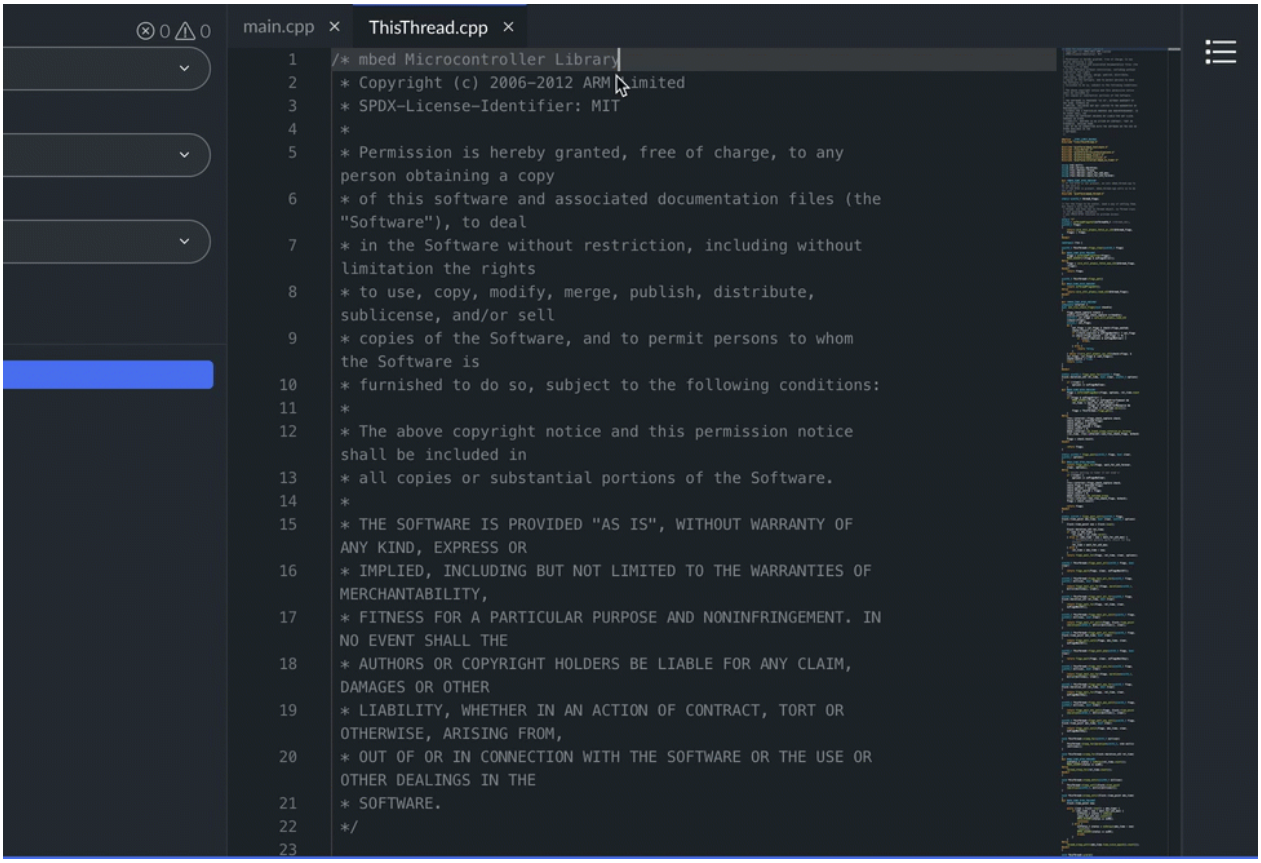
Describes how to use the Go to Symbol... option in Keil Studio to perform a symbol-focused search of your code.

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 8-5: Go to Symbol... option



Open the Outline view

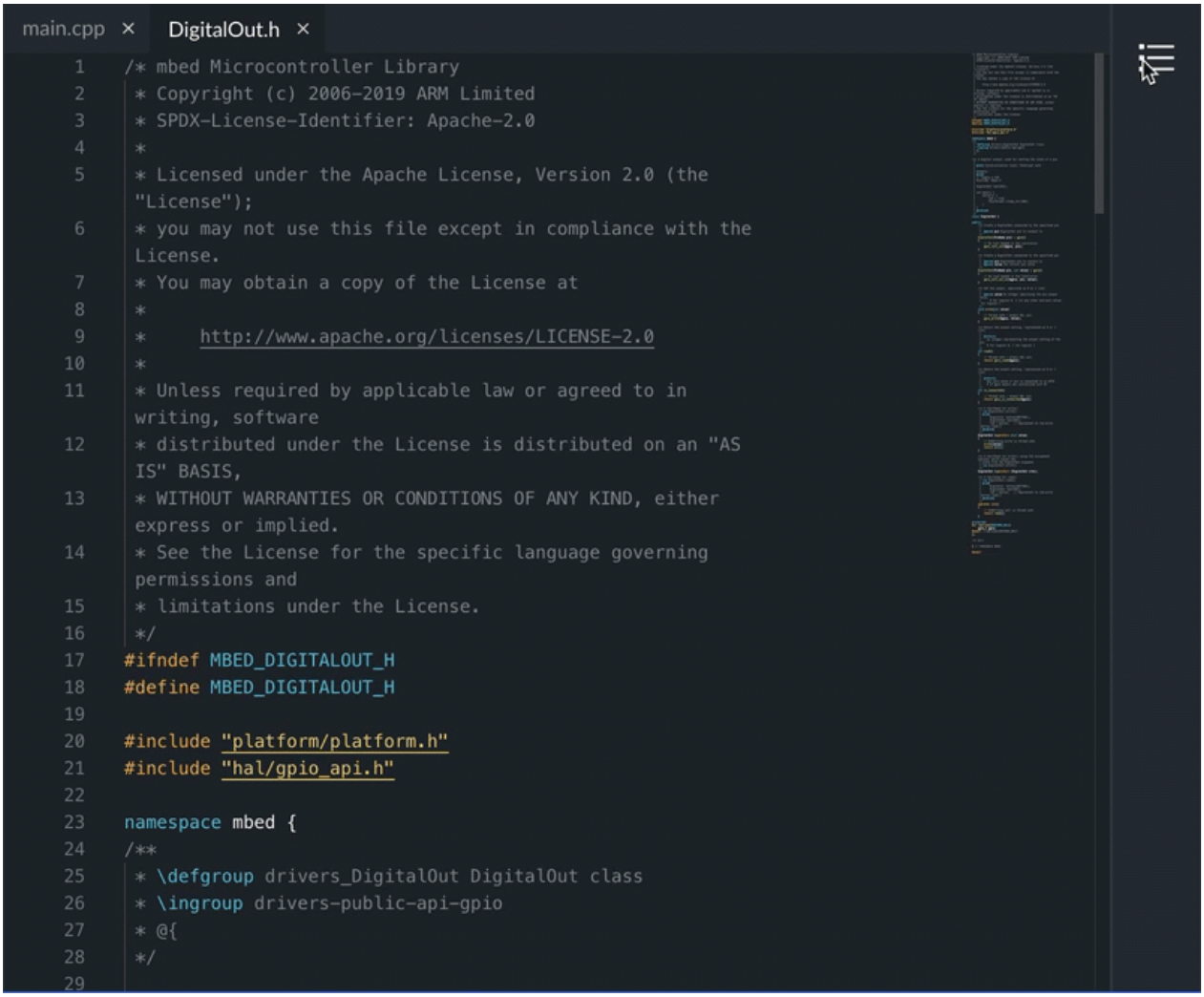
The Outline view gives you a tree view of the classes, variables, and functions in the current file.

To use the Outline view:

1. From the View menu, select Outline or click the Outline button  at the top-right 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 8-6: Intellisense outline view



8.1.2 Edit and refactor your code

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 8-7: Auto-completion and i icon

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 8-8: Renaming a symbol

Code formatting

This topic describes how to change the indentation to format you 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 8-9: Changing the indentation with Format Selection

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

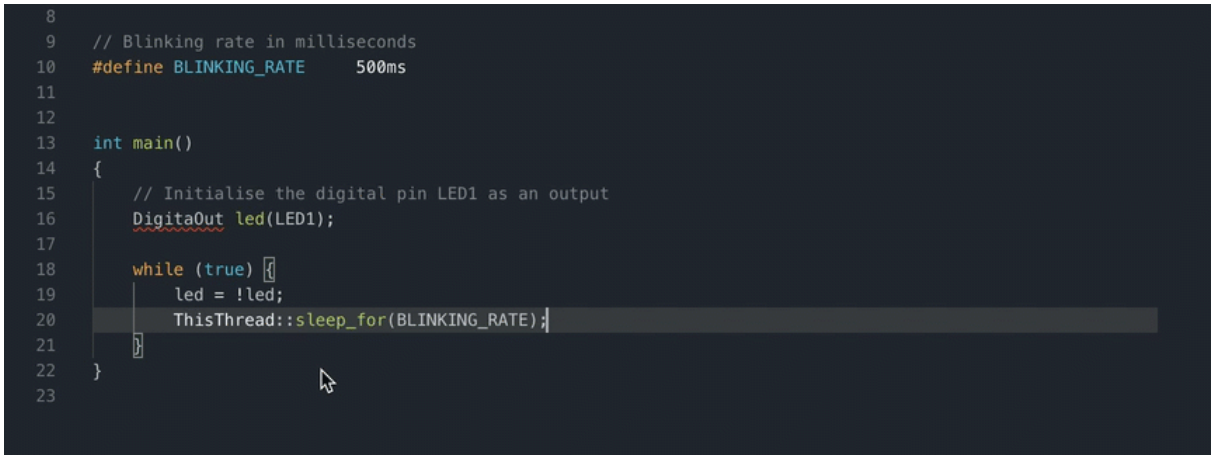
Fix-its offer suggestions to correct problems in your code, for example a missing bracket or semicolon.

Either:

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

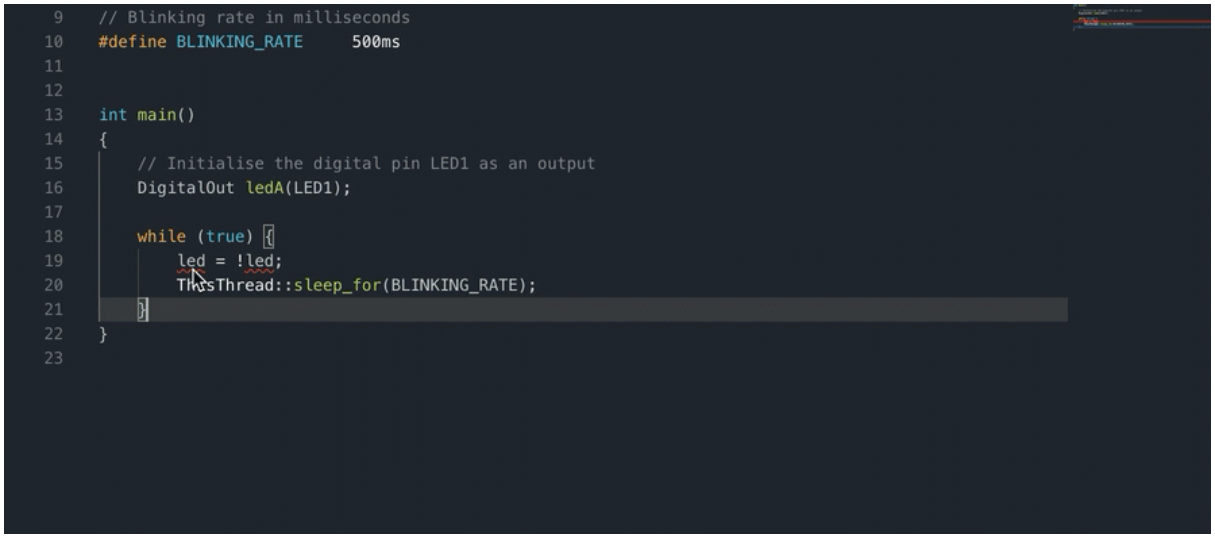
  A lightbulb icon appears next to the problematic line. Click the lightbulb and check the correction suggested. If you are happy with the suggestion, click the popup message.

Figure 8-10: Fix-it example



- 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 8-11: Peek problem example



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.

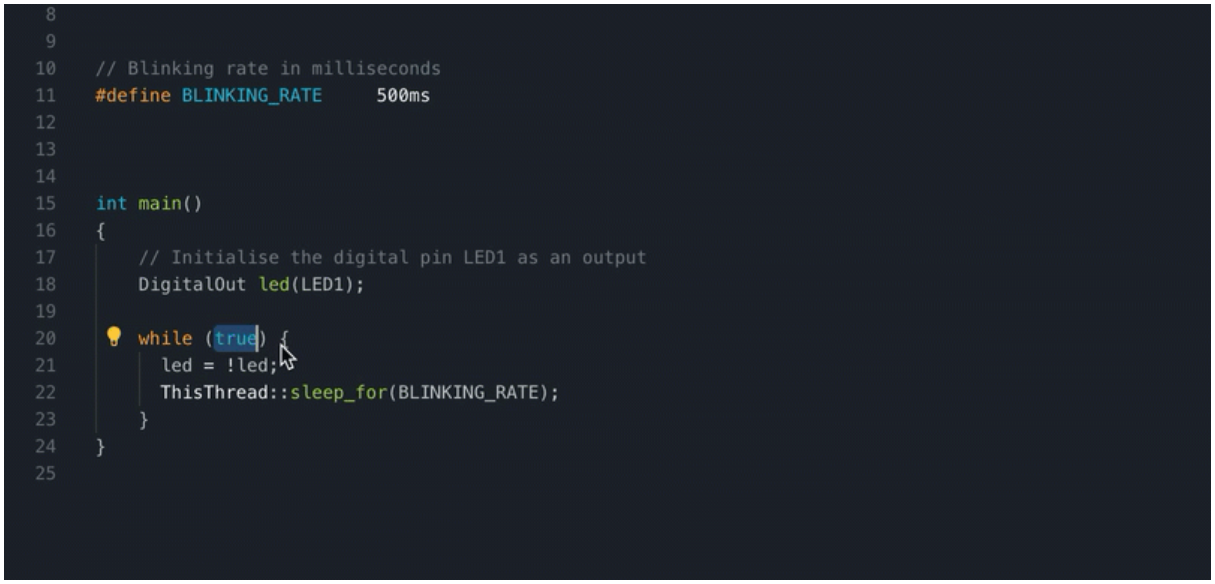
Examples

- Extract a function:

Figure 8-12: Extract to function refactoring example

- Extract a variable:

Figure 8-13: Extract to variable refactoring example



8.2 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 clangd.

8.2.1 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: '*'`.

8.2.2 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.

8.2.2.1 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.

8.2.2.2 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.

9 Manage files

Describes how to compare, upload, download, and move files in Keil Studio.

9.1 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 64
Compare local and remote versions in Mercurial source control on page 76

9.2 Upload and download files or projects

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.

9.3 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.

9.4 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.

10 Managing a project

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

10.1 Manage a CMSIS project and its components

The Manage Software Components view in Keil Studio allows you to see which software components are included in a project. In the current version of Keil Studio, the Manage Software Components view is read-only.

In a future version, you will be able to select the software components to include in your project and manage the dependencies between components. It will be possible to build a 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. The Project outline is read-only. In a future version, you will be able to easily manage your project files from the Project outline.

10.1.1 Open the Manage Software Components view

The Manage Software Components view shows all the software components in the active CMSIS project. This topic describes how to open the Manage Software Components view.

Procedure

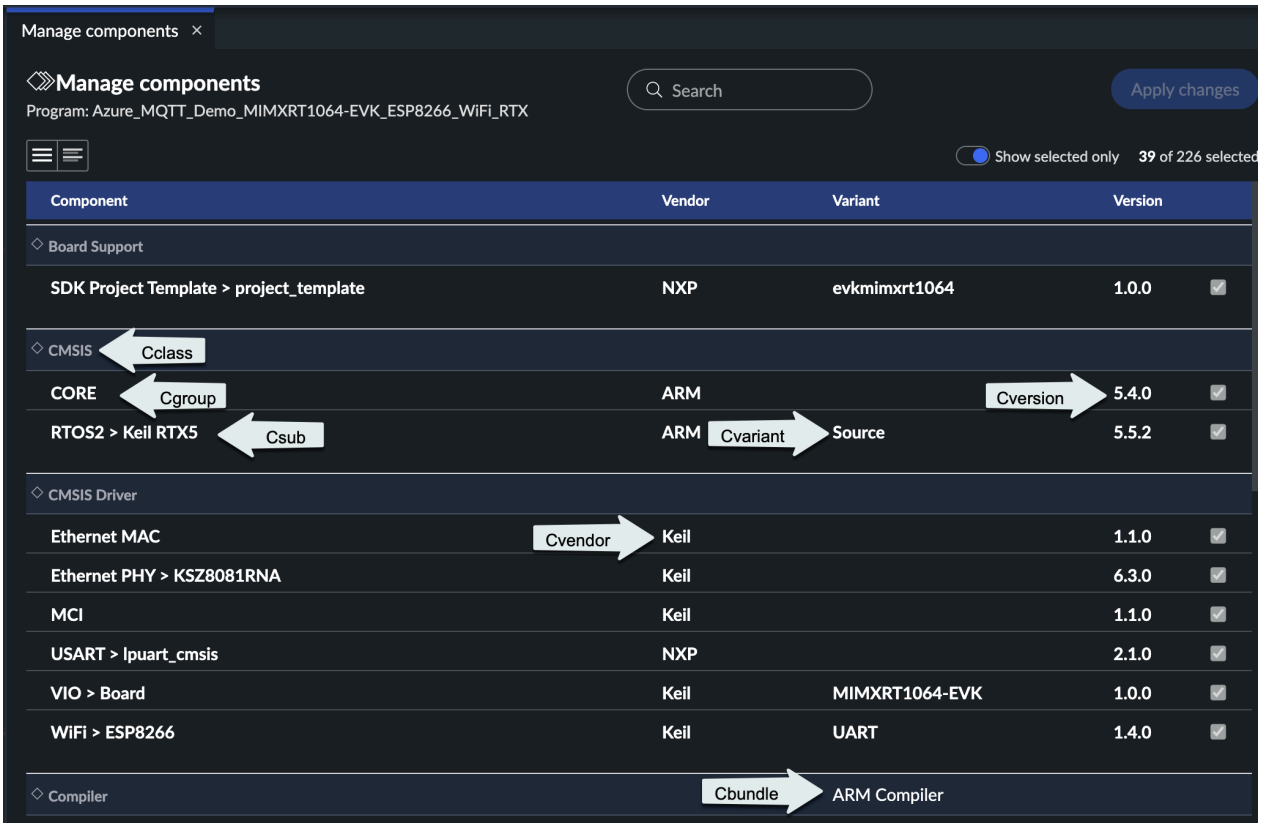
1. Set the project you want to work on as the active project.
2. Click the Open software components view icon  from the Explorer.

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 will be available for use in a future version of Keil Studio.

Figure 10-1: The 'Manage Software Components' view showing the components included in the active project



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

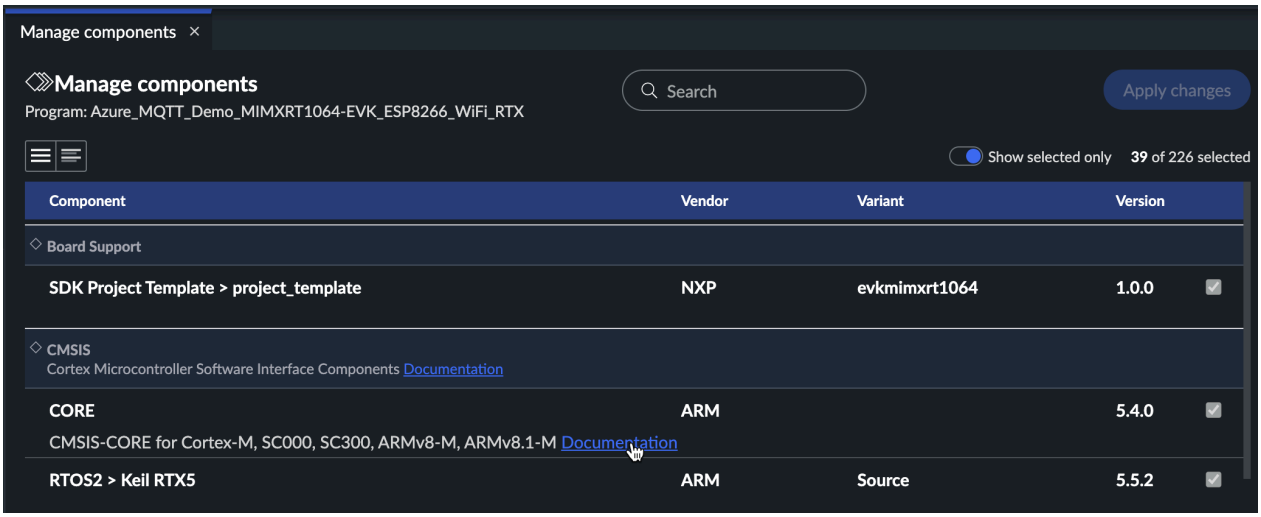
- Component class (Cclass): A top-level component name. For example: CMSIS. For a full list of component classes, see Software Component Cclasses.
- Component group (Cgroup): A component group name. For example: CORE for the CMSIS component class.
- Component version (Cversion): The version number of 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: RTOS2 (API) > Keil RTX5 for the CMSIS component class.
- 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.

The last 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. In a future version of Keil Studio, you will be able to change the number of instances deployed for your project.

Figure 10-2: 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.

10.2 Manage an Mbed project and its libraries

Describes how to manage Mbed projects and libraries.

When you create a new project or import 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 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.bld` file) from the Mbed Libraries view. You can manually change the version of Mbed in the `mbed.bld` file . 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.

10.2.1 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.

10.2.2 Import an Mbed library

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 at 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.

10.2.3 Remove an Mbed library

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.

10.2.4 Update and change Mbed libraries

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).

10.2.4.1 Update all Mbed libraries

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 at the top-right 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. At the top right 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.

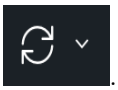
10.2.4.2 Update an Mbed library to the latest version

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  .

10.2.4.3 Update an Mbed library to a specific version

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.

10.2.5 Fix library problems

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 at the top-right 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  at the top-right 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.

10.2.6 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.


Note

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.

11 Source control

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.

11.1 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.

11.1.1 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 navigation sidebar 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 in the User Profile view on page 100
User Profile view on page 100

11.1.2 Interface and features reference

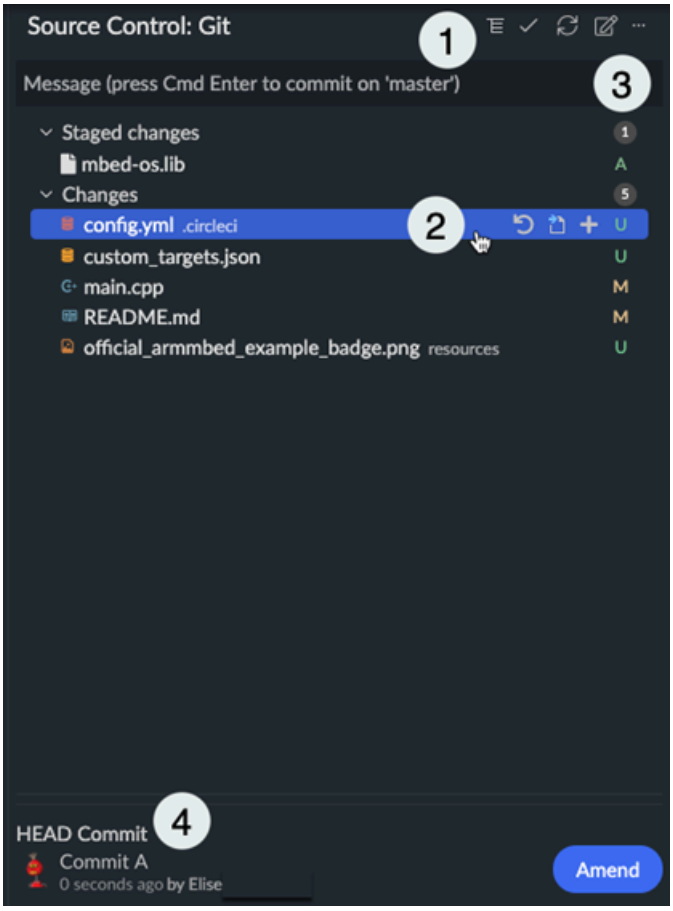
Source control in Keil Studio is handled in two views: The Source Control view, for handling the current work, and the History view to see previous commits. They always show the active project.

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 11-1: 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.

11.1.3 Configure a project for source control and collaboration

Whether you have decided to work from a new example project, or you have imported 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 imported projects:

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

11.1.3.1 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 import from Git (the remote repository is automatically set to the repository you imported it from).

11.1.3.2 Fork a Git project

Learn how to fork a project imported 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 imported from GitHub, then Keil Studio forks the project directly. Check your GitHub account.
   - If the project was imported from another Git hosting service, the Setup 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.

11.1.4 Create or switch branches

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 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 master branch by default).

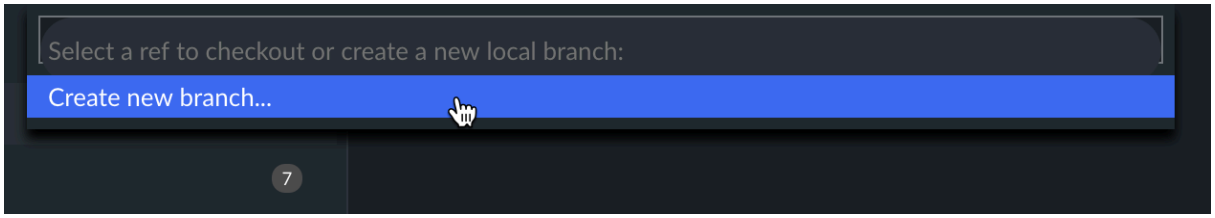
Procedure

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.

Figure 11-2: Create a branch, or select one from the list



11.1.5 Manage local files

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

11.1.5.1 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.

11.1.5.2 View changes

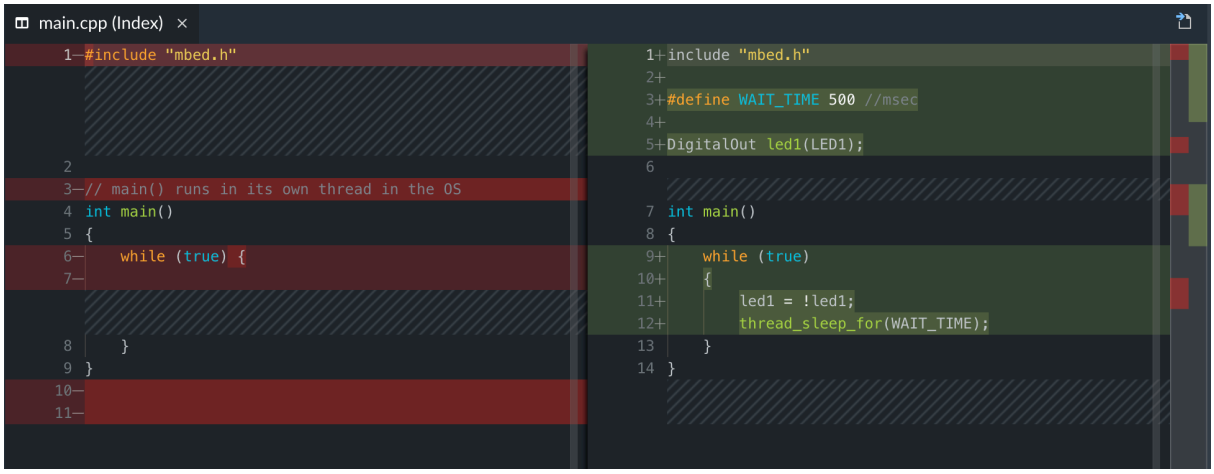
Describes how to view the changes you have made to a file in Keil Studio.

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 11-3: Double-click a file to view its side-by-side comparison

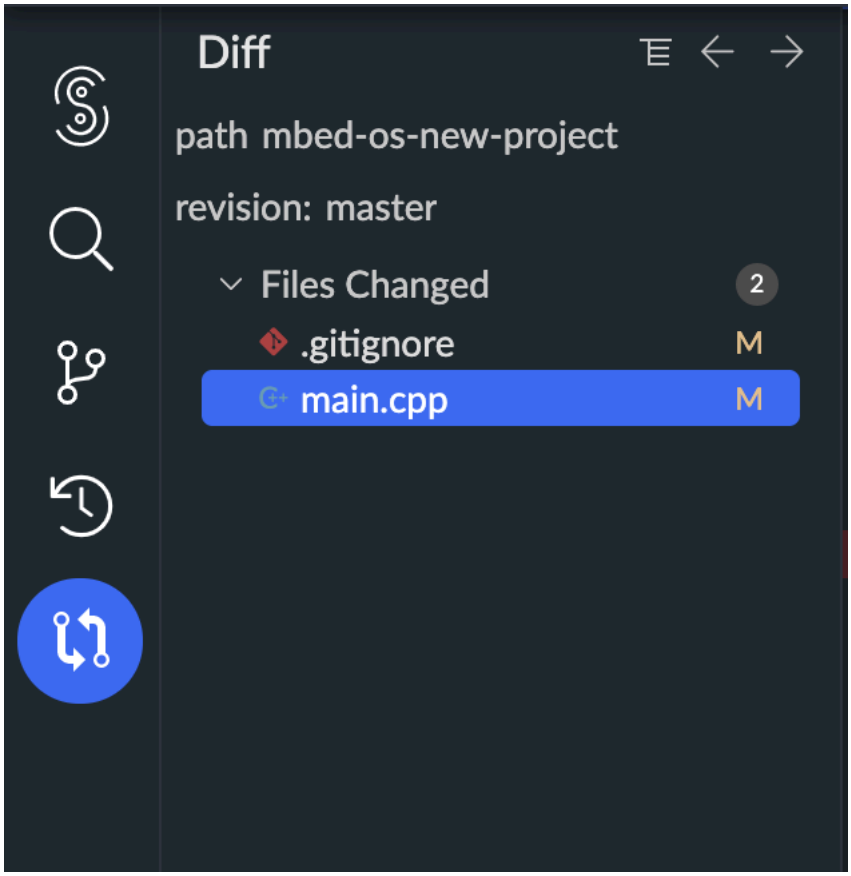


- To compare the same file or folder across multiple Git branches, right-click a file or folder and select Compare with. Keil Studio shows a list of available branches.

  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. To open a diff view for that file, click a file:

  Figure 11-4: List of files changed in the folder



Next steps

You can open the file for editing by clicking the Open File icon at the top-right 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.

11.1.5.3 Stage, commit, and push changes

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.

11.1.5.4 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.

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.

11.1.5.4.1 Change the last commit

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.

11.1.5.4.2 Change multiple commits

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.


Note

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.

11.1.5.5 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 of the Pro Git book.

11.1.6 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.

11.2 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.

11.2.1 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.

11.2.2 Interface and features reference

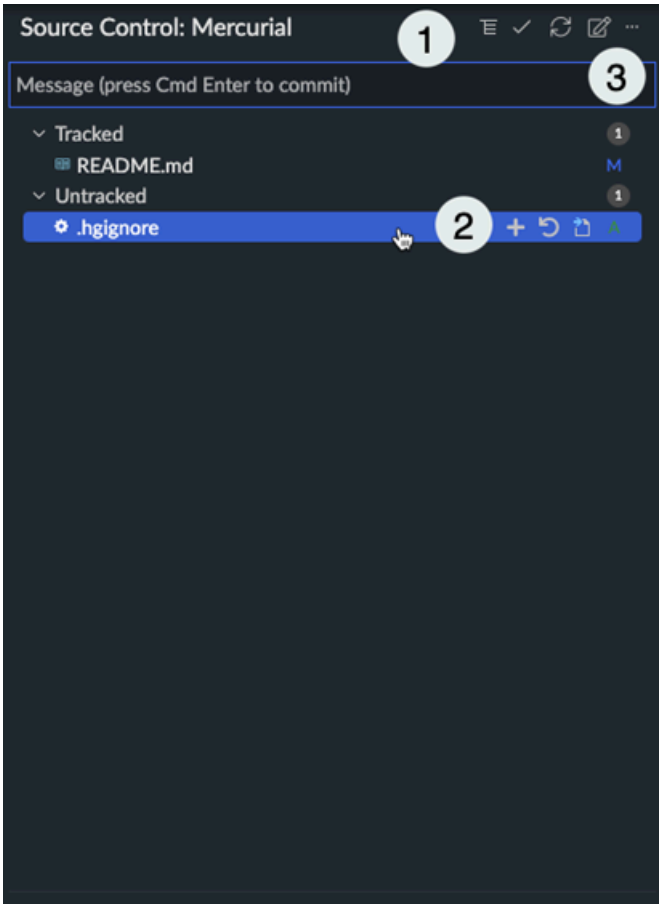
Source control in Keil Studio is handled in two views: The Source Control view, for handling the current work, and the History view to see previous commits. Both views always show the active project.

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 11-5: 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.

11.2.3 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 imported projects:

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

11.2.3.1 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.

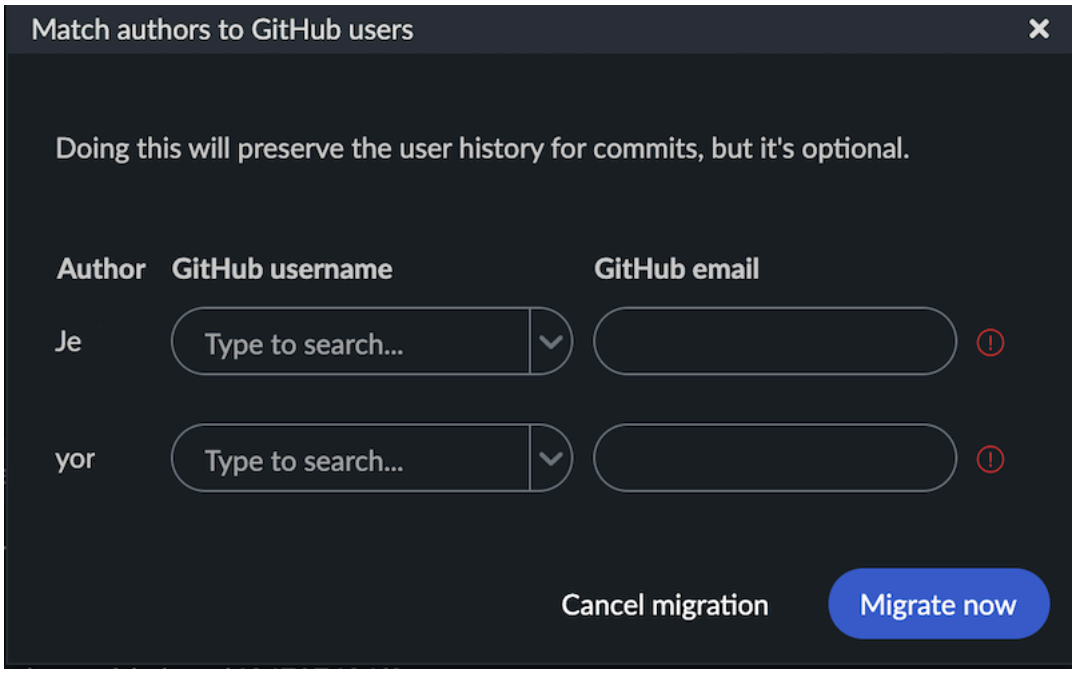
11.2.3.2 Convert a Mercurial project to Git

Learn how to convert a project imported 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 Setup 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.

   Figure 11-6: Matching authors to users



8. Click Migrate now.
   Keil Studio converts the project to Git and creates a repository on GitHub.

11.2.4 Create or switch branches

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).

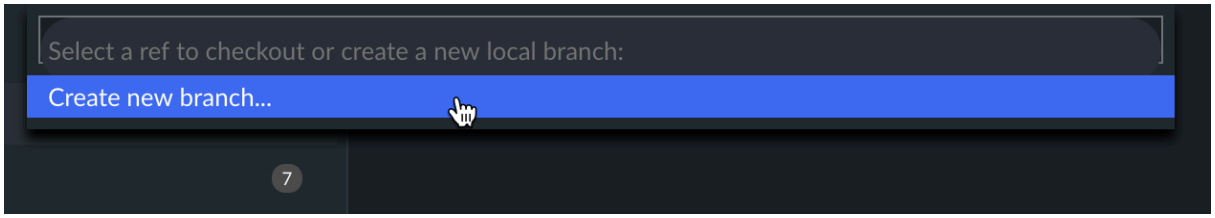
Procedure

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.

Figure 11-7: Create a branch, or select one from the list



11.2.5 Manage local files

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

11.2.5.1 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.

11.2.5.2 View changes

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 11-8: Double-click a file to view its side-by-side comparison

  - Open the file for editing by clicking the Open File icon at the top-right 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.

11.2.5.3 Track, commit, and push changes

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.

11.2.5.4 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.

11.2.6 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.

11.3 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.

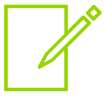
To see the details of the changes on a given file, double-click the file. The changes open in a new tab.

12 Monitor and debug

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

12.1 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  .

12.2 Debug a project with Keil Studio

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

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.

Note

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.

Debug first steps

To debug:

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.

Note

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.

Restart or stop the debugger

If you click the Restart button  when:

- The debugger was running, the debugger continues to run until a breakpoint is hit or until you click Pause.
- The debugger was 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  .

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

Note that Step Over and Step Out operations halt on breakpoints.

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.


Tip

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.

To set a breakpoint:

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.

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.

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.

To set a function breakpoint:

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.

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.


Note

In the current version of Keil Studio, only one thread displays in the Threads list (Main).

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.

Inspect variables

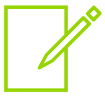
You can inspect variables and check if they are storing the values you expect. If you find an incorrect value, find out where it was set (you might need to restart the debugger, look at the call stack, or both).

Local variables, global variables, function arguments, and register contents 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.


Note

Only modules with global variables are visible.

The following table provides register details:

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).

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.

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: • 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.

Setting	Description
Reset Mode	Controls the reset operations performed by the debugger. The options are: • 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: • 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: • 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.
Internal Console Options	Controls when the Debug Console view displays. The options are: • 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: • 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.

12.3 Use the Memory view

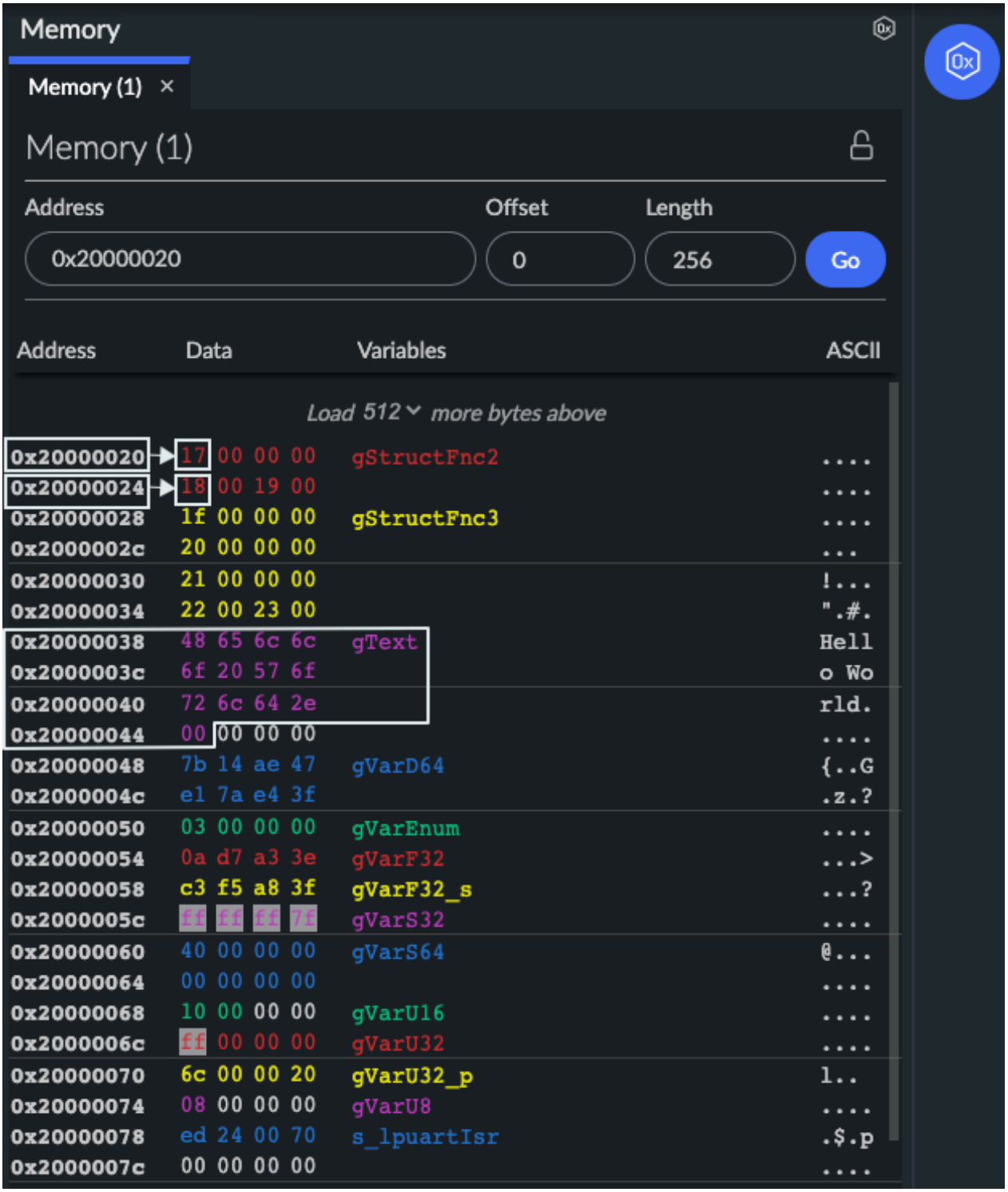
This section describes the Memory view and explains how to monitor the memory usage of your board as you step through the code of your project during a debug session.

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 12-1: 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  at the top-right 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.

13 Exporters

To download any CMSIS or Mbed project, select File > Download Active Project in Keil Studio. Projects are exported as `.tar` files and can be used with the Keil µVision IDE. To import your project, follow the Keil µVision import process.

14 Supported and custom targets

Describes the hardware that Keil Studio supports and how to set up custom targets.

14.1 Supported development boards

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

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

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

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 Preferences

This section discusses the preferences and shortcuts that you can update to tailor Keil Studio to your preferred setup. The section also discusses how to manage your accounts in Keil Studio.

15.1 Manage accounts in Keil Studio

You can go to the User Profile view to manage the accounts you use with Keil Studio.

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.

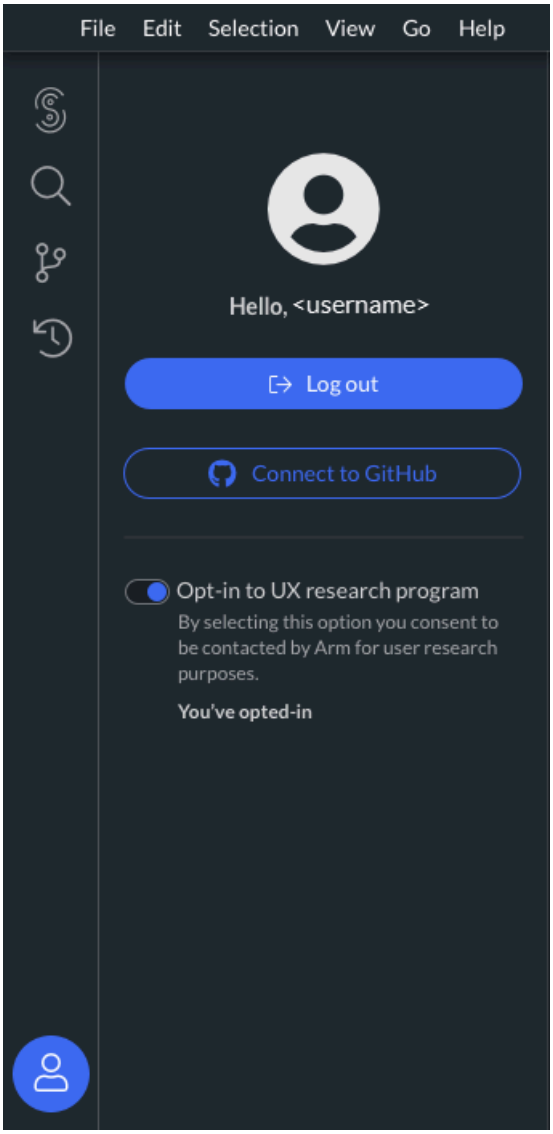
15.1.1 User Profile view

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 15-1: User Profile view



Access the User Profile view

To access the User Profile view, click the Accounts icon  in the Keil Studio navigation sidebar 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.

Feature	Description
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.

15.2 Change preferences

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

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.

   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. If you have several workspaces and want to define preferences by workspace, select the correct workspace. To switch between workspaces, use File > Open Recent Workspace....
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 15-1: 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.

15.3 Set word wrap preferences for the editor

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.

15.4 Customize 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.

15.4.1 Change keyboard shortcuts

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.

15.4.2 Reset keyboard shortcuts

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 Use the JSON file.

15.4.3 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 available at all times.

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.


Note

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 Windows and Linux 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.

16 Integration with Pelion Device Management

For Mbed projects only: The Arm Pelion IoT platform connects devices to the cloud for easy management.

The Pelion Device Management (PDM) services manage the connection of the device and resource management, and remote firmware updates.

The PDM integration in Keil Studio configures your project with the certificates, IDs, and other credentials it requires to connect to PDM. Keil Studio can also perform a firmware update with a single click, making it simple to test the PDM functionality of your project: verify your project can connect to your PDM account, send the correct resources, and correctly apply a new firmware update.

You need a Pelion Device Management account to use the PDM services. If you do not have an account, request access from the Pelion website.

16.1 Overview

Describes what a project requires to connect to PDM and receive updates.

To be able to connect to PDM and receive updates, a project (the first image installed on a device) must have:

- A connect certificate to access a Pelion Device Management account. When you run the project on a device, the bootstrap server uses this connect certificate to authenticate your project with your Pelion Device Management account.
- An API key to access Pelion Device Management services.
- An update certificate to verify that incoming updates come from a trusted source.
- A bootloader to verify and install update images.
- The Pelion Device Management Client library (`mbed-cloud-client.lib`), normally imported as part of the example project `mbed-os-example-pelion`.

For development use cases only, you can use Keil Studio to generate the certificates ("developer" connect and update certificates) and the manifest file. For production use cases, use offline tools to generate and store these certificates securely.

Once the initial, updatable project is on the device, the device itself is considered updatable; the device can receive over-the-air updates. These updates can be sent to the device (or multiple

devices) using Pelion Device Management (all your devices must be connected to Pelion Device Management before rolling out the campaign).

To update a device, you need:

- A new firmware image. This image does not include the certificates, keys and bootloader of the original project; the image contains only the project code.

  A manifest file, which defines the update, including the location of the new firmware image and the type of device the update applies to. The manifest file is signed with a private key to assure the device that the image is from a trusted source and has not been tampered with.
- A private key so you can sign the firmware update manifest file.

From Keil Studio, you can then roll out an update campaign for your image, using the manifest, to your selected devices. Keil Studio monitors the devices and shows applied updates.

16.2 Connect your device to Pelion Device Management

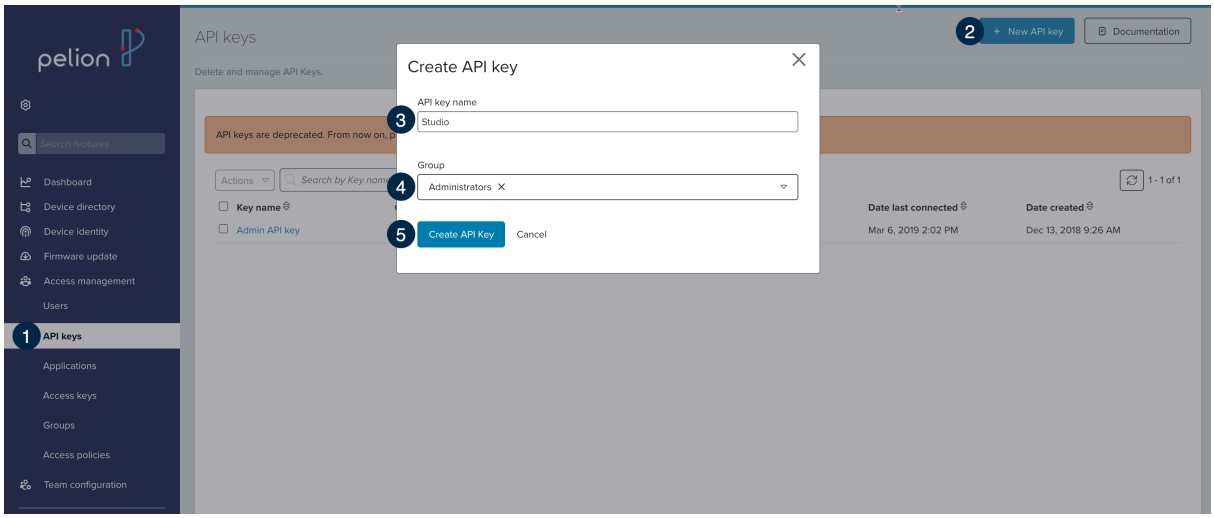
Describes how to create a connect certificate and give your project access to your Pelion Device Management (PDM) account.

Procedure

1. Create a project from the `mbed-os-example-pelion` example.
   Make sure the project is set as active.

2. If you do not have a Pelion Device Management API key, create one:
   a. Go to the Pelion Device Management view.
   b. Click Open Pelion Device Management. Your web browser opens the Pelion Device Management (PDM) portal.
   c. Log in to your Pelion Device Management account.
   d. Select Access management > API keys.
   e. Click + New API key and follow the instruction on the screen.

Figure 16-1: Creating an API key

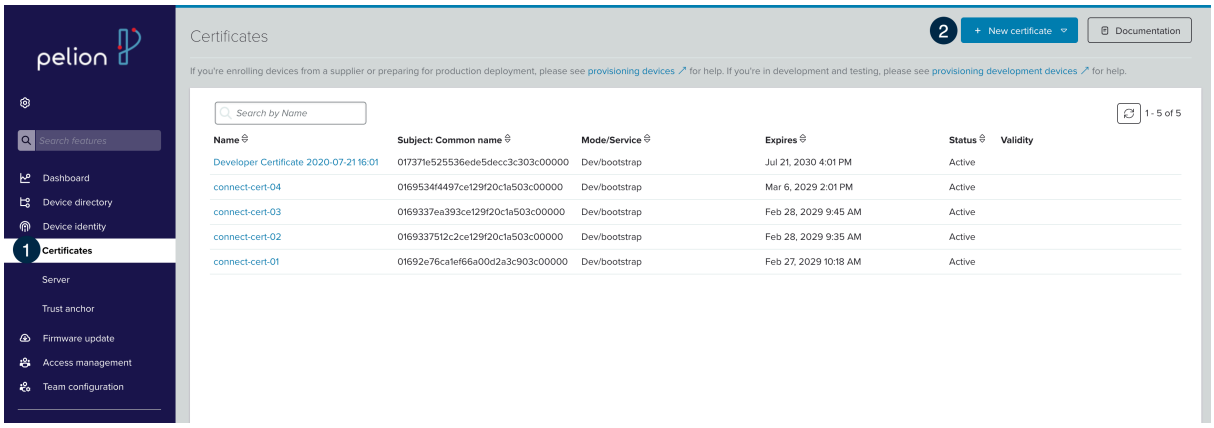


Copy the API key to the clipboard.

3. Add your API key to Keil Studio:
   a. In the Pelion Device Management view, click the cogwheel icon. The Pelion DM Account Settings dialog box opens.
   b. Paste your API key in the API key field and click Save.

4. If you do not have a connect certificate, create one:
   a. In the Pelion Device Management (PDM) portal, select Device identity > Certificates.
   b. Select + New Certificate > Create a developer certificate and follow the instructions to create a developer certificate.

Figure 16-2: Pelion creating a developer certificate



   Warning: The developer connect certificate file contains multiple certificates and a private key. The certificate is not suitable for production and you must keep it safe so as not to expose the private key. Arm recommends adding it to your `.gitignore` (for Git repositories) or `.hgignore` (for Mercurial repositories) file, so that it is not pushed to your source control repositories.
5. To add your certificate to your project, in the Keil Studio Pelion Device Management view, click Configure.
   Keil Studio uses your API key to display your available certificates. Select the one you want to add to the project.
6. To be able to update your project, leave the Enable remote firmware update checkbox selected. Leaving the checkbox unticked adds an update certificate and key to the project.
   Warning: The update certificate and key are not suitable for production use cases.
7. Click Done.
   Keil Studio updates the file `mbed_cloud_dev_credentials.c` in your project with your developer certificate information.

   The project can now connect to your Pelion Device Management account.
8. Set up your project to connect the internet, Wi-Fi or Ethernet:
   - For Wi-Fi: In the `mbed_app.json` file for your project, set up your Wi-Fi credentials: enter a value for `nsapi.default-wifi-ssid` (replace `SSID`) and `nsapi.default-wifi-password` (replace `PASSWORD`). Ensure you leave the quote marks and escape characters, for example `"\"<newname>\""`, and `"\"<password>\""`.
   - For Ethernet: Connect your board with a cable; no project setup is required.
9. Build and run the project to connect your device. When your device goes online, you can see it in the Pelion Device Management (PDM) portal.

16.3 Update workflow

Once you have flashed the initial project to your device (over USB), you can start using Pelion Device Management (PDM) to remotely update new versions of the project. This topic describes how to update your device.

Procedure

1. Make changes to your project.
2. In the Pelion Device Management view, click Push to devices. Keil Studio:
    a. Builds the project as a new image, which does not include the bootloader (that was only needed for the initial project).
    b. Creates a manifest for the update, signed with your private key.
    c. Uploads the image and manifest to Pelion Device Management and updates all connected devices that match the class ID and vendor ID listed in the manifest.

Results

Keil Studio reports the number of successfully updated devices.

The vendor and class IDs are randomly generated on your machine as part of the PDM configuration process, so it is highly unlikely that the update will target anyone else's device. You can also set the vendor and class IDs manually.

You can see your update image, manifest, and campaign in the Firmware update menu of the Pelion Device Management (PDM) portal.

16.4 Pelion troubleshooting

Describes solutions to some issues you might experience when using the Pelion Device Management (PDM) portal.

Using a new update certificate fails

If you must change the update certificate for your project, or if you are using the same development board to test more than one application (each application must have a unique update certificate), you must erase the storage of your board; the certificate is not erased in the normal process of reflashing a board.

Solution

You can erase the storage of your board by:

- Sending `r`: Focus on the Serial Monitor view and press R.
- For an external SD card, connect the card to your computer and format the card as instructed by the manufacturer.

17 Known issues and troubleshooting

Describes Keil Studio known issues, how to troubleshoot some common issues, and how to contact Arm to report issues and make development suggestions.

17.1 Known issues

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. Follow the instructions on the Visual Studio documentation site to increase the limit.
- On Linux, source control management requires the `libcurl4` package to work correctly.

17.2 Troubleshooting

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

Can't log into Keil Studio

You are unable to log into Keil Studio.

Solution

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

Cannot make an Mbed project the Active Project

If a project requires an Mbed OS library, but there is no Mbed OS library imported into that project, you cannot set that project as the Active Project. Instead, you see a warning such as “Cannot set”<project-name >" as Active Project as it is not recognised as a valid project.".

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 to be called `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.

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.

Connected development board or debug probe not found

You have connected your development board or debug probe, but the board or probe is not being detected by Keil Studio.

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.

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

Connection fails when clicking Run project or Debug project

Your development board or debug probe has been correctly detected by Keil Studio and 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 sub-systems (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 previous 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 need to change the jumper settings to disconnect the on-board debugger.
  - Check if you need 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. This 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. This is for example the case for Mbed projects running on ST boards.
- 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 and/or the CPU 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. This can require changing on-board switches and/or jumpers 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 solutions above 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.

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`.

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.

17.3 Report an issue or make a development suggestion

Help us develop Keil Studio.

Report any issues you experience, or make development suggestions 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.