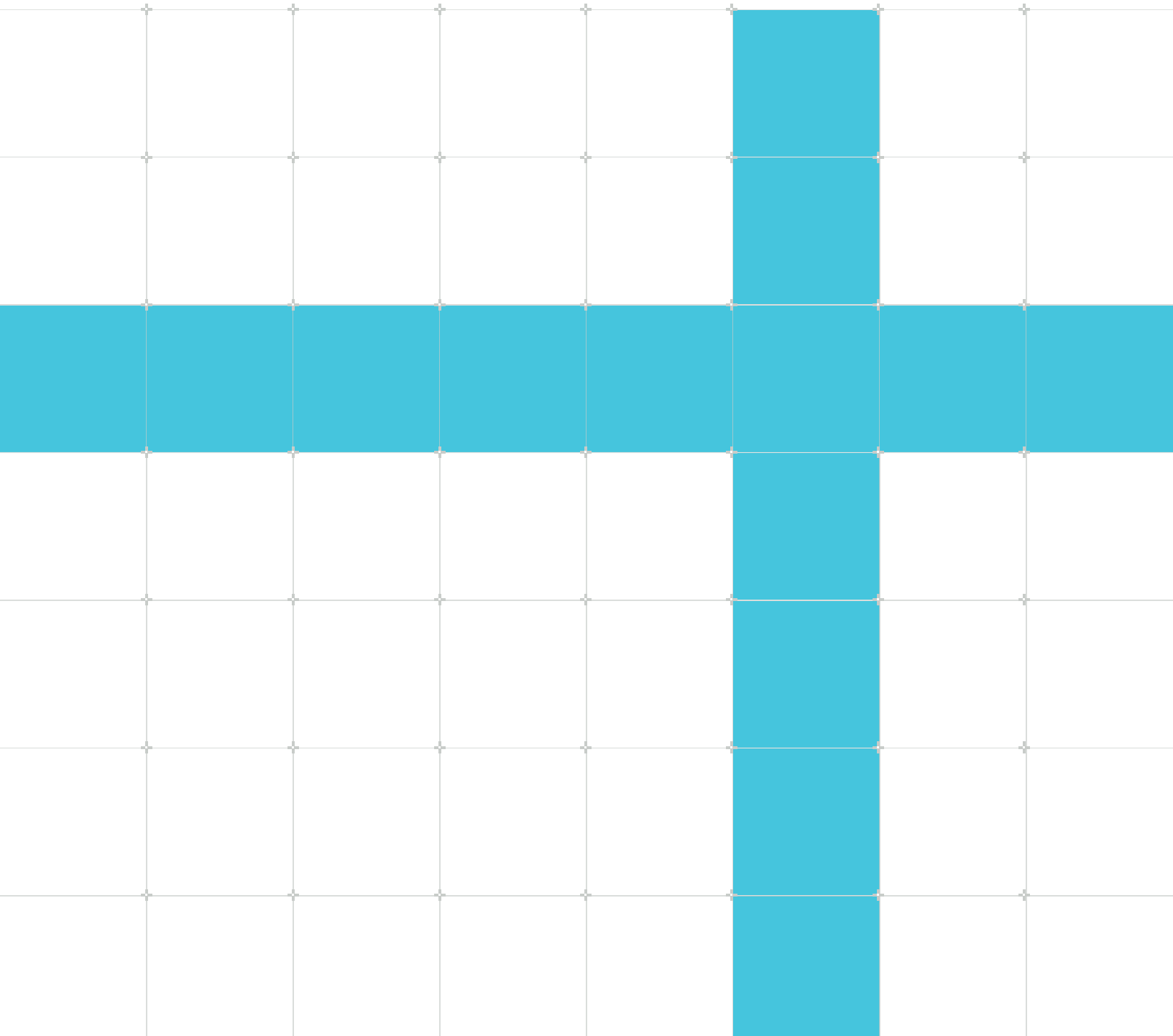
Arm® Keil® Studio Cloud

Version 1.7

User Guide

Issue 11
102497_1.7_11_en



Arm® Keil® Studio Cloud User Guide

This document (102497_1.7_11_en) was issued on 2024-07-25. There might be a later issue at http://developer.arm.com/documentation/102497

The product version is 1.7.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This book is written for all developers who are involved in the development of embedded, IoT and Machine Learning software for Cortex-M devices.

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.

Previous issues of this document included language that can be offensive. We have replaced this language. See Revision history on page 118.

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

Feedback

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

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

Contents

1. Arm Keil Studio Cloud

Arm® Keil® Studio Cloud is a free to use, browser-based integrated development environment (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 the Keil µVision® software development platform. You can compile projects using Arm Compiler 6, run the projects directly on 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 enabling Arm ecosystem partners to provide professional software, tools, and services.

Keil Studio Cloud 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 Cloud 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 Cloud also provides limited support for Mbed 2. To get started, you can import Mbed projects from your Mbed Online Compiler workspace or mbed.com.

Keil Studio Cloud is part of the Arm Keil Microcontroller Development Kit (MDK). MDK is a collection of software tools for developing embedded applications based on Arm Cortex-M processors and Ethos™-U processors. MDK gives you the flexibility to work with a command-line interface (CLI) or an integrated development environment (IDE), or by deploying the tools into a continuous integration workflow.

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

For more information about supported hardware, see Supported development boards and MCUs and Supported debug probes.

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

2. Prerequisites

Find out about the system and account requirements you need to access Keil® Studio Cloud.

2.1 System requirements

Keil® Studio Cloud is supported in the following desktop browsers:

- Google Chrome
- Microsoft Edge
- Opera
- Safari
- Mozilla Firefox

To work with development boards over USB, you must use Keil Studio Cloud in a desktop browser that supports the WebUSB standard: Google Chrome or Microsoft Edge (Chromium).

Keil Studio Cloud fully supports Mbed™ OS 5.12 and newer, and Mbed OS 6. It also provides build and run support for Mbed 2 and earlier versions of Mbed OS 5 (debugging is not supported with these versions). See Manage an Mbed project and its libraries or standalone libraries for more details.

If you are moving a project from Mbed OS 5 to Mbed OS 6, note the deprecated APIs.

Installation of udev rules on Linux

On Linux, udev rules grant permission to access USB devices. You must install udev rules to be able to build a project and run it on your hardware or debug a project with Keil Studio Cloud.

See udev rules for Linux for more information.

2.2 Access Keil Studio Cloud

You need an Arm® account or an existing Mbed™ account to work with Keil® Studio Cloud.

You can create an Arm account from the Keil Studio Cloud page.

Learn how to manage the accounts you use with Keil Studio Cloud.

If you are working behind a firewall and you are having difficulties accessing Keil Studio Cloud, check Keil Studio Cloud domains.

2.3 Keil Studio Cloud domains

Your firewall can block unknown domains. If you are encountering issues accessing Arm® Keil® Studio Cloud, check the following list.

Keil Studio Cloud uses the following domains:

Domains / Used by	Keil Studio Cloud login	Arm account login	Keil Studio Cloud loading	Hardware detection
*.akamaihd.net		x		
*.hotjar.com			x	
684dd331.akstat.io		x		
account.arm.com		x		
arm.com		x		
az416426.vo.msecnd.net		x		
c.go-mpulse.net		x		
cdn.designsystem.arm.com		x		
dc.services.visualstudio.com		x		
developer.arm.com		x		
fast.fonts.net		x		
fonts.googleapis.com	x	x	x	
fonts.gstatic.com	x	x	x	
googletagmanager.com		x		
login.microsoftonline.com		x		
login.windows.net		x		
o37279.ingest.sentry.io			x	
open-vsx.org			x	
os.mbed.com	x			x
resources.digital-cloud.medallia.eu		x		
s.go-mpulse.net		x		
schemastore.azurewebsites.net			x	
schemastore.org			x	
solar-search.api.keil.arm.com				x
studio.keil.arm.com	x		x	
studio.mbed.com			x	

Domains / Used by	Keil Studio Cloud login	Arm account login	Keil Studio Cloud loading	Hardware detection
sw-center.st.com				x
udc-neb.kampyle.com		x		

3. Manage accounts

Describes how to manage the accounts you use with Keil® Studio Cloud from the **User Profile** view.

By default, when you log into Keil Studio Cloud, your Arm® account or Mbed™ account gets listed in the **User Profile** view.

If you have a personal, organization, or enterprise GitHub account, you can connect that account with Keil Studio Cloud. Connecting your GitHub account with Keil Studio Cloud allows you to access and create GitHub repositories from Keil Studio Cloud. To learn how to add your GitHub account in Keil Studio Cloud, see Set credentials for GitHub.

3.1 User Profile view

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

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

- Arm® account or Mbed™ account: Allows you to log out of Keil Studio Cloud. By default, when you initially log into Keil Studio Cloud, your Arm account or Mbed account is listed in the **User Profile** view.
- GitHub account: Connect or disconnect your GitHub account from Keil Studio Cloud.

Access the User Profile view

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

Features

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


Note

Disconnecting your GitHub account using the **Disconnect GitHub account** button does not remove the Keil Studio Cloud application from your list of authenticated GitHub applications. You can manage your authenticated GitHub applications from your account settings in GitHub.

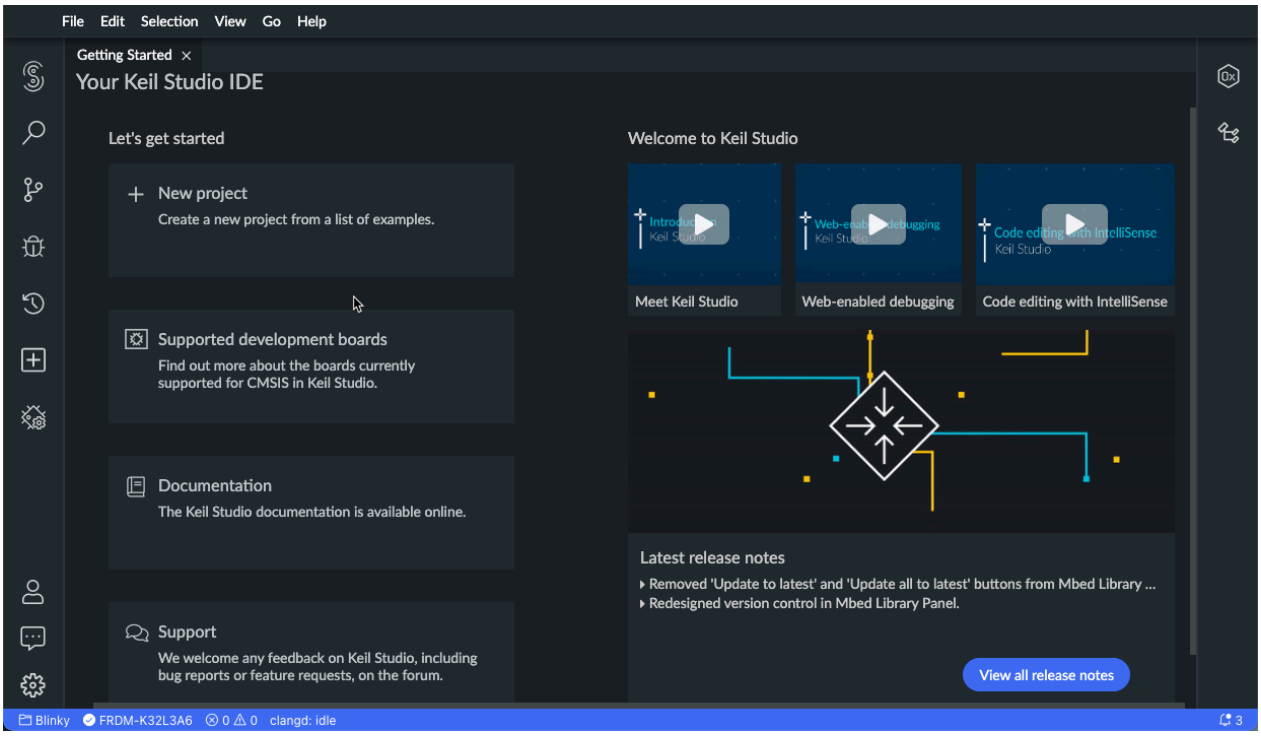
4. User interface

Discover the user interface and the Command palette and learn how to use Keil® Studio Cloud in the best way.

4.1 Getting Started page

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

Figure 4-1: Getting Started page



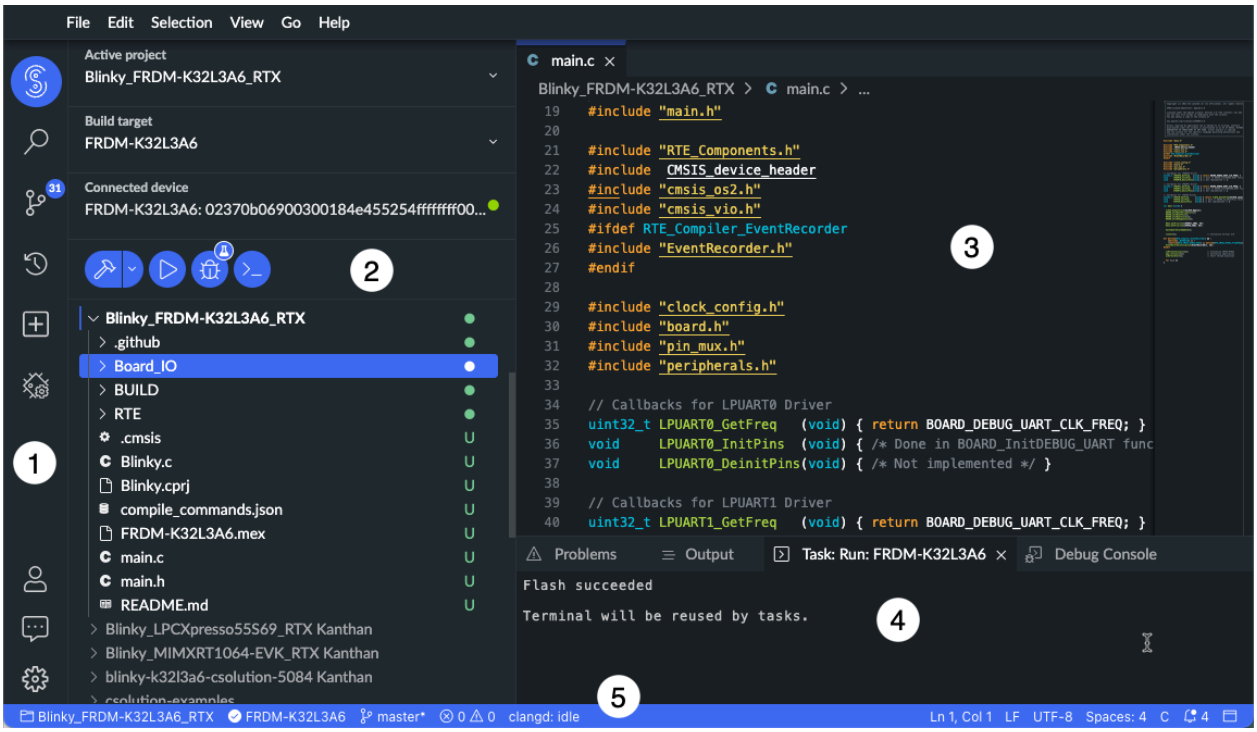
This page gives you useful shortcuts, links to videos presenting Keil Studio Cloud's most popular features and a link to the release notes.

In addition, the tutorials available in this guide provide step-by-step instructions to help you get started with Keil Studio Cloud.

4.2 UI layout

The Keil® Studio Cloud UI has an intuitive and customizable layout made of views.

Figure 4-2: UI layout



The UI is divided into different areas:

1. Activity bar: On the left-hand side, the activity bar allows you to switch between different views and displays context-specific indicators (for example, the number of Git outgoing changes). The activity bar also gives you access to the **User Profile** view, feedback and help options, and customizable settings.
2. Side bar: The side bar contains the **Explorer** view. The **Explorer** shows all the projects contained in your workspace organized by folders. Projects with a red dot next to them contain files with errors. If there are errors in a file, the number of errors displays next to the file name. The letters next to file names are the Git or Mercurial statuses. If you are working with Mbed™ projects, you can see a Symbolic Link icon next to the **mbed-os** library folder of a project. The area just above the workspace tree view shows the active project and build target currently selected for the project, as well as the connected device. The main buttons to build, run, debug the active project, and display the output of the board (**Serial Monitor** view) are also available from here. To search for a file name, click anywhere in the file tree and start typing. View and filter the search results by clicking on the pop-up box that appears on the right side of the file tree.
3. Editor: The editor on the right shows the content of the files that you have opened from the workspace tree view. It is the main area to edit your files. You can open as many editors as you like side by side, vertically or horizontally.

4. Panels: You can display different panels below the editor area for output or debug information, or errors and warnings. Choose the information to display from the drop-down list on the right side of the panel. You can also right-click to search the panel display.
5. Status bar: The status bar gives details about the active project and the files that you are editing. If you are using source control, the branch you are working on displays there too.

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

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

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

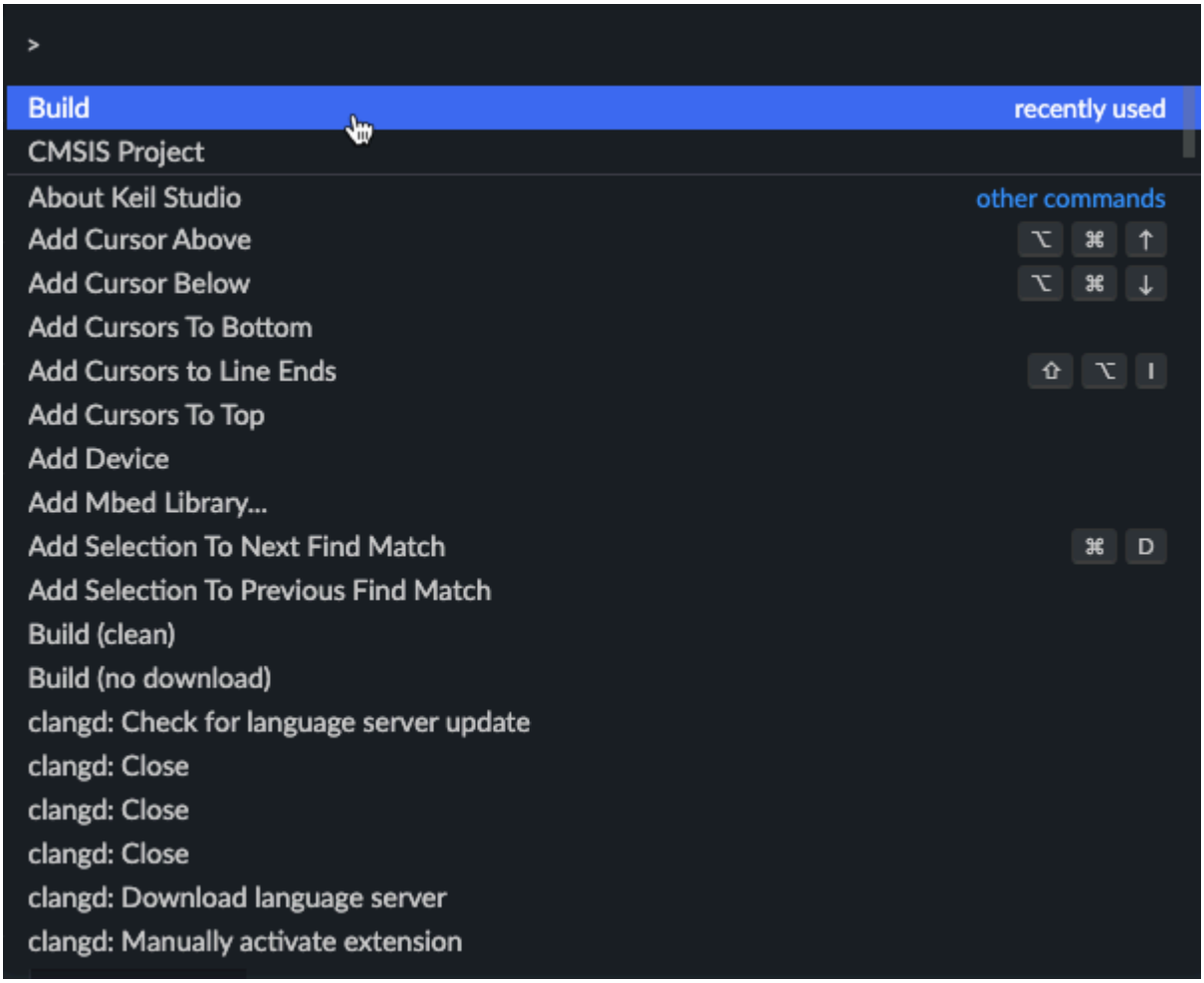
To restore the default layout, go to **View** > **Command Palette...** and look for the **View: Reset Workbench Layout** command. This command reloads Keil Studio Cloud with the default layout.

4.3 Command palette

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

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

Figure 4-3: Recently used commands appear at the top

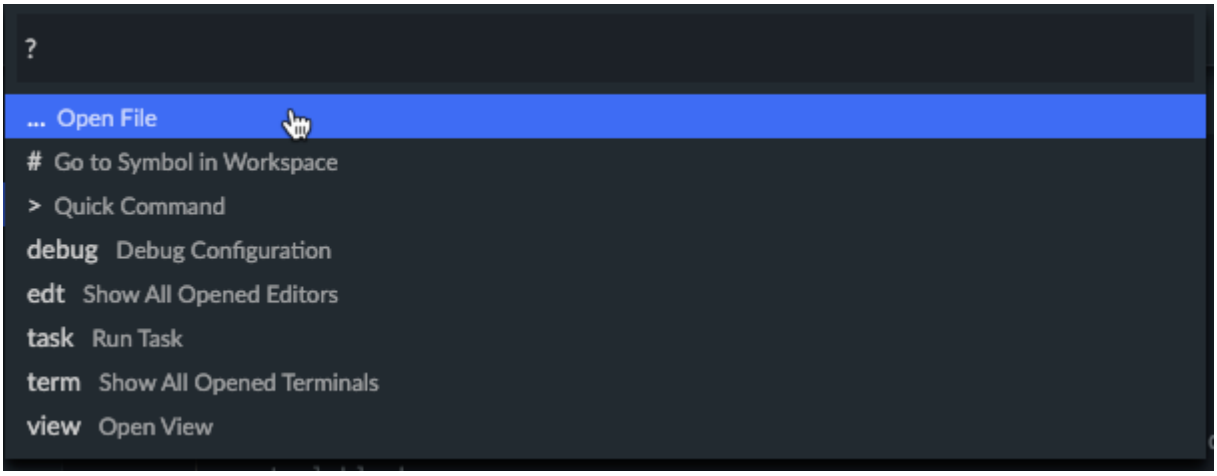


You can also open files or search for symbols from the Command palette.

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

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

Figure 4-4: Available commands



4.4 Editor

An essential part of Keil® Studio Cloud is the editor where you edit your code. Learn about the editor and some useful editing tips.

4.4.1 Use side-by-side editing

You can open as many editors as you like side by side, horizontally or vertically, with the **Split Editor Horizontal** and **Split Editor Vertical** options available in **View** > **Editor Layout**.

Whenever you open a new file, the editor that is active displays the content of that file. You can resize editors and reorder them. Drag and drop the title area of the editor that you want to reposition or resize.

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

Experiment with the **Split Editor Right**, **Split Editor Left**, **Split Editor Up**, and **Split Editor Down** options.

You can maximize the size of the editor region with the **View** > **Appearance** options: **Toggle Bottom Panel**, **Toggle Status Bar Visibility** and **Collapse All Side Panels**.

4.4.2 Minimap

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

If the minimap does not display, close the file, select **View** > **Toggle Minimap**, and open the file again.

4.4.3 Editing tips

Here are some common features to make code editing faster.

Multi cursor selection

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

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

Column selection

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

Copy line up or down

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

Move line up or down

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

Navigate to beginning or end of file

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

4.5 Customize the UI

Describes the preferences and shortcuts that you can update to tailor Keil® Studio Cloud to your preferred setup.

4.5.1 Preferences

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

4.5.1.1 Change preferences

Describes how to change your preferences in Keil® Studio Cloud.

Procedure

1. Go to **File** > **Preferences** > **Open Settings (UI)**.

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 only for the currently open workspace. Preferences that you set in the **Workspace** tab take precedence over preferences that you set in the **User** tab.

In the current version of Keil Studio Cloud, there is only one workspace available for each account, so you can use the **User** tab or the **Workspace** tab interchangeably.

The **Open Settings (JSON)** icon  opens a JSON file storing all the preferences you changed from their default values.

2. Select the **User** or the **Workspace** tab.
3. Click the preference that you want to change and modify the values as needed.

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: Associations** 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 with a language:

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

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

Results

After you have changed a preference, a blue line and a cogwheel icon appear to the left of the preference. To reset the preference, click the cogwheel icon and select **Reset Setting**. 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.

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** > **Preferences** > **Color Theme**. To indent with tabs, click the **Spaces** option on the information bar when the editor is open.

4.5.1.2 Set word wrap preferences

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

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

- **wordWrapColumn**: When you select this option, text wraps at the value set for **Word Wrap Column**.
- **bounded**: When you select this option, 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.

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

4.5.2 Keyboard shortcuts

Keil® Studio Cloud 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.

4.5.2.1 Change keyboard shortcuts

Describes how to change keyboard shortcuts in Keil® Studio Cloud.

Procedure

1. Select **File** > **Preferences** > **Open Keyboard Shortcuts**.
2. In the **Type to search in keybindings** field, type the name of a command or keybinding. You can also search the context and `"when"` clauses. To clear the field, click **Clear Keybindings Search Input**  .
3. Move your cursor over the shortcut that you want to modify and click **Edit Keybinding**  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 **Supported Keys** in Use the JSON file.

   To make further changes, click **Open Keyboard Shortcuts (JSON)**  . This action 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 that you want to modify. For more information, see Use the JSON file.

4.5.2.2 Reset keyboard shortcuts

Describes how to reset keyboard shortcuts in Keil® Studio Cloud.

Procedure

1. Search for the keyboard shortcut that you want to reset.
2. Move your cursor over the shortcut and click **Reset Keybinding**  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 that you want to reset in the `keymaps.json` file. For more information, see Use the JSON file.

4.5.2.3 Use the JSON file

All the changes that you make to keyboard shortcuts are stored in a `keymaps.json` file. 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 the following elements:

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

For example, if we modify the keyboard shortcut for the **Copy Line Down** command to use `shift+down` instead of `shift+alt+down` to trigger the command when using the editor, you get:

```
{
   "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

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 does not work on a Windows or Linux machine because these machines do not have equivalent keys.

You can also use the following strings for special keys:

- `backspace`
- `tab`
- `enter`
- `return`
- `capslock`
- `esc`
- `escape`
- `space`
- `pageup`
- `pagedown`
- `end`
- `home`
- `left`
- `up`
- `right`
- `down`
- `ins`
- `del`
- `plus`

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

5. Tutorials

Get started with Keil® Studio Cloud with our tutorials.

5.1 Get started with an Mbed OS Blinky example

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

Before you begin

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

Procedure

1. Select **File** > **New...** > **Mbed Project**.
   The **New Project** dialog box opens.
2. Click the **Example project** drop-down list and select `mbed-os-example-blinky` under **Mbed OS 6**.

You can also use the `mbed-os-example-blinky-baremetal` example, if you are working with an ultraconstrained device. For more information, see the Mbed OS bare-metal profile page.

3. Use the default project name `mbed-os-example-blinky` or enter your own project name.
4. Check the default options are correct for your project:
   - Keil Studio Cloud sets the newly created project as the active project. Build and run commands apply only to the active project. Clear the checkbox if you do not want to make the new project active.
   - Keil Studio Cloud 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. Connect your board to your computer. If it is the first time that you are connecting your board, follow these steps:
   a. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
   b. Click the **Add Device** button and select the device firmware for your board in the dialog box that displays at the top of the window, then click **Connect**.

Your hardware displays in the **Device Manager**. After the first successful connection, Keil Studio Cloud automatically detects the board. If your hardware is not detected, see Connected development board or debug probe not found.

7. From the **Explorer**, select a build target in the **Build target** drop-down list. Your build target name most likely matches the board name. A build target tells Keil Studio Cloud how to build Mbed OS so that it matches your hardware.
8. To build the project, click the **Build project** button,  . **Build project** builds Blinky and stops.
9. Click **Run project**  . **Run project** builds Blinky and runs it on your board.
You might have to restart your board for Blinky to run.

5.2 Get started with a CMSIS Blinky example

This tutorial explains how to start a project in Keil® Studio Cloud 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 clone a project, and then build and run Blinky on your board.

Before you begin

You can download the example project either from keil.arm.com, or from within Keil Studio Cloud.

Procedure

1. Go to keil.arm.com.
2. Move your cursor over the **Hardware** menu and select **Boards** to see the list of supported boards.
You can search for a board by name or vendor, or filter the list by vendor or core. You can also display boards with example projects with the **Only include boards with example projects** checkbox.
3. Find the **FRDM-K32L3A6** board in the list, and then select it.
You can open example projects from the **Projects** tab, or download a project as a .zip file. You can also get access to board details from the **Features** and **Documentation** tabs.
4. Find the **Blinky** example in the **Projects** tab. Move your cursor over **Get Project**, and then click **Open in Keil Studio Cloud**. Alternatively, you can open the example project from within Keil Studio Cloud by clicking **File** > **New...** > **CMSIS Solutions Examples** and choosing the Blinky_FRDM-K32L3A6 project from the **Example project** drop-down list.
5. Log into Keil Studio Cloud with your Arm® account or your Mbed™ account if you are not already logged in.
Keil Studio Cloud opens.
6. Confirm the project name in the **Clone** dialog box. Keil Studio Cloud sets the newly cloned project as the active project by default.
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.
8. The **Build target** is automatically set to the build target of the project:

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

To build the project, click **Build project**,  . **Build project** builds Blinky and stops.

9. Connect your board to your computer.
10. If it is the first time that you are connecting your board, follow these steps:
    a. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
    b. Click **Add Device** and select the device firmware for your board in the dialog box that displays at the top of the window.
    c. Click **Connect**.

    Your board displays in the **Device Manager**. After the first successful connection, Keil Studio Cloud automatically detects the board.

    The target is connected. Now, **Run project**  is enabled.
11. Go to **File** > **Preferences** > **Open Settings (UI)** and select **cmsis** for **Flash Mode** in the **Run** category.
12. Click **Run project** to build Blinky and run it on your board.
13. Choose the **cm4** processor in the **Select a processor** drop-down list that displays at the top of the window.
    The flashing starts.


Note

For boards with multiple cores, you must select the appropriate processor for your project in the **Select a processor** drop-down list that displays at the top of the window when you click **Run project**. Note that the **Select a processor** drop-down list does not display when the **daplink** option is selected for the **Flash Mode** setting.

5.3 Work with Git source control

This tutorial explains the basics of Git source control in Keil® Studio Cloud 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 carry out the following tasks:

- 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

5.3.1 Configure your project for source control

Learn how to work with Git.

Before you begin

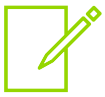
In this tutorial, you will learn how to publish your changes to a new GitHub repository.

There are various ways of working with source control in Keil® Studio Cloud.

- You can fork an existing Git project and publish your changes to your GitHub account.
- You can start from a local project and use an existing (empty) remote repository to publish changes.
- You can publish your changes to a new repository from Keil Studio Cloud.

For this tutorial, you need:

- A GitHub account. Create an account on github.com or, if you already have an account, sign in. When you are signed in, check that your Arm® and your GitHub accounts are connected. See Set credentials for GitHub for more information.
- The `Blinky` example project that you created in Get started with a CMSIS Blinky example.



If you do not have the example yet, you can download it from keil.arm.com, or from https://github.com/Arm-Examples. You can also download it from within Keil Studio Cloud by clicking **File** > **New...** > **CMSIS Solutions Examples** and choosing the Blinky_FRDM-K32L3A6 project from the **Example project** drop-down list.

Procedure

1. Open the `Blinky` example project. If it is not set as the active project, right-click the project name and select **Set Active Project**.
2. 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 **Publish Project**, or click **...** at the top of the **Source Control** view and select **Publish Project**.
   - If the project is already initialized for Git, click **...** at the top of the **Source Control** view and select **Publish Project**.

   The **Publish** dialog box opens.

If you download the example project from keil.arm.com rather than from within Keil Studio Cloud, the **Publish Project** option is not available, and you must select **Fork on GitHub**. A message confirms that the fork has been created, and the fork is listed in your GitHub account. You can then use source control and publish your changes in the usual way. For more information, see the GitHub documentation on forking a repository.

3. Select the **Publish to a new GitHub repository** option, and then enter a repository name and description.
4. By default, Keil Studio Cloud creates a private repository. If you want to create a public repository, clear the **Private repository** checkbox.
5. Click **Publish**.
   The **Source Control** view shows untracked changes (with a **U** status) in the **Changes** list. Untracked changes are changes that have not been staged yet. When first setting a repository, you are on the `main` branch by default.
6. Go to your GitHub account and check that the repository has been created as expected. The repository does not contain any commits yet.

5.3.2 Use source control and publish your changes

You can now update your project and publish the changes.

Before you begin

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

Procedure

1. In the **Source Control** view, create an initial commit on the `main` branch:
   a. Select all the unstaged files, move your cursor over one of the files, and then click **+** to stage all the files.

      The files are listed under **Staged changes**.

   b. In the **Message** box above the list of files, 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 `main` branch is updated.
2. Create a working branch:
   a. Click the **main** branch in the status bar at the bottom of the window.

      An input field opens 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`.
3. Go to the **Explorer** view, open the `Blinky` project folder, and look for the `Blinky.c` file.
4. Open that file. 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. Modified files are shown with an **M** status in the **Changes** list. Note that you can also make changes from the side-by-side comparison window.
6. Move your cursor 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.
9. Finally, to merge your changes into the `main` branch:
   a. Switch to the `main` branch by clicking `my-branch` in the status bar and selecting `main` in the list that opens at the top of the page.
   b. Click **...** > **Merge...** and select `my-branch` in the list to merge the changes done on `my-branch` into `main`.
10. Click **...** > **Push** to update the remote repository.

5.4 Debug a CMSIS Blinky example

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

Before you begin

- Use the Blinky example from Get started with a CMSIS Blinky example, or clone the Blinky example for the FRDM-K32L3A6 board from keil.arm.com.
- Set the Blinky example as the active project.
- Connect your board to your computer. If it is the first time you are connecting your board, follow these steps:
  1. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
  2. Click **Add Device**. Select the device firmware for your board in the dialog box that displays at the top of the window, and then click **Connect**.

     Your board displays in the **Device Manager**. After the first successful connection, Keil Studio Cloud automatically detects the board.
- Check that an appropriate build target displays in the **Build target** drop-down list. If not, select one.

Procedure

1. Click **Debug project** to start a debug session.

Keil Studio Cloud automatically builds and runs the project on your board. The debug session starts. Keil Studio Cloud 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()`.

2. While the debug session is still running: go to the **Explorer** view, open the `Blinky` project folder, and find the `Blinky.c` file.
3. Open that file. 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 **Continue**  .
   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 is highlighted.
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 documentation.
   a. Click **Continue**  .
   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`.
9. Click **Step over**  until you reach `osDelay(100U);`.
10. Click **Step into**  to check this function.
    It 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 Cloud, there is only one thread, **Main**. The call stack shows the execution flow of your code.
12. Click **Step into**  again several times to check the execution flow.
13. Click **Stop**  to stop the debugger and return to the editor.

5.5 Connect to AWS IoT and send MQTT messages

This tutorial explains how to connect your hardware to AWS IoT using the CMSIS AWS MQTT demo available for the STMicroelectronics B-U585I-IOT02A board. Learn how to manage your AWS IoT things, generate certificates, and configure connection settings in your project. When everything is set up, you will be able to send MQTT messages to AWS IoT.

5.5.1 Manage your AWS IoT things and certificates

Learn how to carry out the following tasks:

- Create a thing and a certificate (and an associated RSA key pair)
- Create a policy and attach it to the certificate
- Activate the certificate and attach it to the thing

Before you begin

- Install the AWS Toolkit extension and connect to AWS with your credentials. Follow the steps in Install the AWS Toolkit extension and Connect to AWS from Keil Studio Cloud. When you are connected to AWS, you can access the AWS Explorer from Keil® Studio Cloud.
- You need a board that can connect to the internet. For this tutorial, we assume that you are using the B-U585I-IOT02A board.
- Clone the AWS MQTT Demo example for the B-U585I-IOT02A board (in Keil Studio Cloud, go to **File** > **Clone...**).

5.5.1.1 Create a thing

Things are used to connect hardware to AWS IoT. A thing is a representation of a specific hardware component.

Procedure

1. Click the **AWS** icon in the activity bar to open the AWS Toolkit.
2. Click **Add regions to AWS Explorer...**.
   A dialog box opens.
3. Select a region or regions, and then click **OK**.
4. Open the **IoT** folder. Right-click the **Things** folder, and then select **Create Thing**.
   A pop-up opens and asks you for a new thing name.
5. Type a name and press **Enter**.
   Keil® Studio Cloud adds your thing to the **Things** folder.

5.5.1.2 Add a certificate and RSA key pair

You need a certificate to establish a secure connection between AWS IoT and your hardware. An X.509 certificate is a digital certificate that uses the international X.509 Public Key Infrastructure (PKI) standard to verify that a public key belongs to the hostname/domain, organization, or individual contained in the certificate.

Procedure

1. Right-click the **Certificates** folder and select **Create Certificate...**.
   A dialog box opens.

2. In the folder view, click **local**, and then click **Save certificate here**. (Create a **local** folder in your workspace if you do not already have one.)
   Keil® Studio Cloud saves an X.509 certificate and RSA key pair in the folder.

5.5.1.3 Create a policy and attach it to the certificate

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

Before you begin

The following policy allows any resource to access AWS IoT. In production, you must use a more secure policy. See the AWS documentation on Policies and permissions in IAM for more details.

Procedure

1. In the **Explorer** view, right-click **local**, and then select **New File**.
2. Name the file `myPolicy.json`.
3. Copy and paste the following code into the file:

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

4. Go back to the AWS Explorer and attach the policy to the certificate that you created earlier. The certificate is inactive for now and no policies are found.
5. Right-click the certificate and select **Attach Policy...**.
6. Select the policy in the pop-up that appears.
   Keil® Studio Cloud adds the policy under the certificate.

5.5.1.4 Activate the certificate and attach it to the thing

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

Procedure

1. Right-click the certificate and select **Activate...**.
2. Right-click the thing and select **Attach Certificate...**.
3. Select the certificate in the pop-up that appears.
   Keil® Studio Cloud adds the certificate under the thing.

5.5.2 Configure connection settings in your project

There are different elements to configure in the `aws_clientcredential.h` and `aws_clientcredential_keys.h` files contained in your project. These elements control how the example application connects and authenticates with AWS IoT.

5.5.2.1 Edit configuration

Update the application configuration to include your credentials, certificates, and endpoints. To include your certificates in the application code, you must first convert them to C strings.

Procedure

1. In the workspace tree view, find the certificate and private key files named <xxxxx>-certificate.pem.crt and <xxxxx>-private.pem.key. Right-click each one, and select **Duplicate**.
2. Open each duplicated file and perform the following steps:
   a. Press Ctrl+F followed by Ctrl+H. The **Find and Replace** dialog box opens.
   b. Make sure that the **Use Regular Expression** option is enabled by pressing Alt+R or clicking the **.*** icon.
   c. In the **Find** box, enter `^(.*)$`.
   d. In the **Replace** box, enter `"$1\\n" \\`.
   e. Press Ctrl+Alt+Enter to run the replace operation on all matches.
   f. Save the modified file. The contents of the file should now be a C string literal.
3. Open the modified certificate file (<xxxxx>-certificate.pem copy.crt). Copy the entire contents by pressing Ctrl+A and then Ctrl+C.
4. Open the `./aws_mqtt_mutualauth_demo/amazon-freertos/demos/include/aws_clientcredential_keys.h` project file. Scroll down until you find the `#define keyCLIENT_CERTIFICATE_PEM \` section that defines the certificate, and then insert the copied text below this line.
5. Open the modified private key file (<xxxxx>-private.pem copy.key) in the **local** folder. Copy the entire contents of the file by pressing Ctrl+A and then Ctrl+C, and then insert the copied text below the `#define keyCLIENT_PRIVATE_KEY_PEM \` line in the `./aws_mqtt_mutualauth_demo/amazon-freertos/demos/include/aws_clientcredential_keys.h` project file.
6. Open the `./aws_mqtt_mutualauth_demo/amazon-freertos/demos/include/aws_clientcredential.h` file. Scroll down to find the `#define clientcredentialMQTT_BROKER_ENDPOINT ""` section, and then insert the AWS IoT Core endpoint from your own account between the quotation marks.
7. In the `#define clientcredentialIOT_THING_NAME ""`, insert the name of the thing that you created earlier in between the quotation marks.

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

Configure your Wi-Fi connection.

Procedure

1. In the **aws_mqtt_mutualauth_demo** project folder, go to **Socket** > **WiFi** and open the `socket_startup.c` file.
2. Enter the Service Set Identifier (SSID) and password of your Wi-Fi network in the `SSID` and `PASSWORD` sections.
3. Enter the Wi-Fi access point security type in the `SECURITY_TYPE` section.
   You can now send MQTT messages to AWS IoT.

5.5.3 Send MQTT messages

Debug your project to start sending messages.

Before you begin

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

Procedure

1. Connect your board to your computer.
2. If it is the first time that you are connecting your board, follow these steps:
   a. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
   b. Click **Add Device** and select the device firmware for your board in the dialog box that displays at the top of the window, then click **Connect**.

   Your board displays in the **Device Manager**. After the first successful connection, Keil® Studio Cloud automatically detects the board.
3. Check that an appropriate build target displays in the **Build target** drop-down list. If not, select one.
4. Click **Debug**  .
   Keil Studio Cloud automatically builds and runs the project on the board. The **Debug** view displays and the debug session starts.


Note

If you are experiencing problems running the project on the board, set **Connect Mode** to **underReset** to be able to run and debug the project. See Advanced debugger settings for more details on the different connect modes.

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

```
AWS IoT Demo
```

```
Connecting to WiFi ...
WiFi network connection succeeded!
```

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

6. Work with CMSIS solutions

Explains what CMSIS (Common Microcontroller Software Interface Standard) solutions are and how to work with them in Keil® Studio Cloud.

6.1 CMSIS solutions

A solution is a container used to organize related projects that are part of a larger application and that can be built separately. See Project Setup for Related Projects for a solution example.

Solutions are defined in YAML format using `*.csolution.yaml` files. A `*.csolution.yaml` file defines the complete scope of an application and the build order of the projects that the application contains. Individual projects are defined using `*.cproject.yaml` files. A `*.cproject.yaml` file defines the content of an independent build.

You can edit the `*.csolution.yaml` and `*.cproject.yaml` files of a solution manually. Keil® Studio Cloud integrates the Red Hat YAML Language Support extension and uses YAML schemas to make the editing of these files easier. See the vscode-yaml repository for more information on the extension.

For more information, see the CMSIS version 5 documentation and the CMSIS version 6 documentation.

See the Build Overview of the CMSIS-Toolbox documentation and the Project Examples to understand how solutions and projects are structured. For more information on csolution project files, see CMSIS Solution Project File Format.

6.2 CMSIS-Packs

Keil® Studio Cloud includes support for CMSIS-Packs. CMSIS-Packs offer you a quick and easy way to create, build, and debug embedded software applications for Cortex®-M devices.

CMSIS-Packs are a delivery mechanism for software components, device parameters, and board support. A CMSIS-Pack is a file collection that might include the following elements:

- Source code, header files, and software libraries (for example real-time operating system (RTOS), digital signal processing (DSP), and generic middleware)
- Device parameters, such as the memory layout or debug settings, along with startup code and Flash programming algorithms
- Board support, such as drivers, board parameters, and descriptions for debug connections
- Documentation and source code templates
- Example projects that show you how to assemble components into complete working systems

CMSIS-Packs are developed by various silicon and software vendors, covering thousands of different boards and devices. You can also use them to enable life-cycle management of in-house software components.

See the Open-CMSIS-Pack documentation for more details.

You can download CMSIS-Packs from keil.arm.com.

6.3 Create, import, or clone a CMSIS solution

Describes how to start working with CMSIS solutions in Keil® Studio Cloud.

You can carry out the following tasks:

- Create a CMSIS solution from an example shipped with Keil Studio Cloud
- Import a CMSIS solution from keil.arm.com
- Import a Keil µVision® project from keil.arm.com and convert it to a solution
- Clone a CMSIS solution from a Git hosting service (such as GitHub)

6.3.1 Create a CMSIS solution from an example project

You can create a CMSIS solution from an example project.

Procedure

1. Open the **File** menu and select **New...** > **CMSIS Solutions Examples**.
   The **New Project** dialog box displays.
2. Click the **Example project** drop-down list and select an example from the list.
   Each example has a `README.md` file that explains the details of that example.
3. In the **Project name** field, edit the name of the new project or keep the name provided by default.
4. Check the default options:
   - Keil® Studio Cloud sets the newly created project as the active project. Build and run commands apply only to the active project. Clear the checkbox if you do not want to make the new project active.
   - Keil Studio Cloud 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 Cloud creates the project in your workspace.

6.3.2 Import a CMSIS solution from keil.arm.com

CMSIS solutions are available for some boards on keil.arm.com. You can import the examples to Keil® Studio Cloud and start working with solutions directly.

Procedure

1. Go to keil.arm.com.
2. Click the **Hardware** menu and select **Boards** to see the list of supported boards.
   You can search for a board by name or vendor, or filter the list by vendor or core. You can also display boards with example projects with the **Only include boards with example projects** checkbox.
3. When you have found your board, check the compatibility of the examples available for the board in the **Projects** tab. Look for the `Keil Studio` compatibility labels.
4. Move your cursor over the **Get Project** button for the project that you want to use and click **Open in Keil Studio Cloud**.
   Keil Studio Cloud opens so that you can import the CMSIS solution. Alternatively, you can download a zip file of the csolution project.

   The **Clone** dialog box opens.

   Keil Studio Cloud sets the solution as the active project by default. Build and run commands apply only to the active project.
5. Clear the checkbox if you do not want to make the solution active.
6. Click **Add project**.
   The project is imported.

6.3.3 Import a Keil µVision project from keil.arm.com

Keil® µVision® `*.uvprojx` examples are available for some boards on keil.arm.com. You can import the examples to Keil Studio Cloud and convert them to solutions (`*.csolution.yaml`). Note that the conversion does not work with Arm® Compiler 5 projects. You can download Arm Compiler 5 projects from the website, but you cannot use them in Keil Studio Cloud. Only Arm Compiler 6 projects can be converted. As a workaround, you can update Arm Compiler 5 projects to Arm Compiler 6 in Keil µVision, then convert the projects to csolutions in Keil Studio Cloud.

Procedure

1. Go to keil.arm.com.
2. Click the **Hardware** menu and select **Boards** to see the list of supported boards.
   You can search for a board by name or vendor, or filter the list by vendor or core. You can also display boards with example projects with the **Only include boards with example projects** checkbox.
3. When you have found your board, check the compatibility of the examples available for the board in the **Projects** tab. Look for the `µVision` compatibility labels.
4. Move your cursor over the **Get Project** button for the project that you want to use and click **Open in Keil Studio Cloud**.

Keil Studio Cloud opens so that you can import the µVision project. Alternatively, you can download a zip file of the µVision project.

The **Clone** dialog box opens.

Keil Studio Cloud sets the solution as the active project by default. Build and run commands apply only to the active project.

5. Clear the checkbox if you do not want to make the solution active.
6. Click **Add project**.
   The project is imported.

   A pop-up displays in the bottom right-hand corner with the following message "Convert µVision project [project-name] to csolution?".
7. Click **Convert**.
   The conversion starts immediately.

   Alternatively, right-click the `*.uvprojx` file in the folder of the project that you have just imported and select **Convert µVision project to csolution** from the **Explorer**.

   You can also run the **CMSIS: Convert µVision project to csolution** command from the Command Palette. In that case, select the `*.uvprojx` that you want to convert in the **Open File** dialog box and click **Select**.
8. Check the **Output** tab. Conversion messages are logged under the **µVision to Csolution Conversion** category.
   The `*.cproject.yml` and `*.csolution.yml` files are available in the folder where the `*.uvprojx` is stored.

6.3.4 Clone a CMSIS solution

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

Before you begin

For solutions on a Git hosting service, you can clone public repositories without setting up your Git credentials. If you want to clone a private repository, set up Git credentials.

For more information about what cloning a repository means, see git-clone on git-scm.com.

Procedure

1. Open the **File** menu and select **Clone...**.
   The **Clone** dialog box displays.
2. Paste the full HTTPS or SSH URL of the relevant web page on a Git hosting service and (optionally) edit the solution directory (**Project name** field).
   Keil® Studio Cloud sets the newly cloned solution as the active project by default. Build and run commands apply only to the active project.
3. Clear the checkbox if you do not want to make the new solution active.
4. Click **Add project**.

You can push changes back to the remote repository from Keil Studio Cloud. For more information, see Source control.

Next steps

You are now ready to work on your solution.

Next you can manage the software components in your solution.

6.4 Manage software components

Describes how to manage the software components of a CMSIS solution.

6.4.1 Software Components view

The **Software Components** view shows all the software components selected in the active project of a CMSIS solution.

From this view, you can see all the component details (called attributes in the Open-CMSIS-Pack documentation).

You can also carry out the following tasks:

- Modify the software components to include in the project and manage the dependencies between components for each target type defined in your solution, or for all the target types at once
- Build the solution using different combinations of pack and component versions, and different versions of a toolchain

6.4.2 Open the Software Components view

Describes how to open the **Software Components** view.

Procedure

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

Results

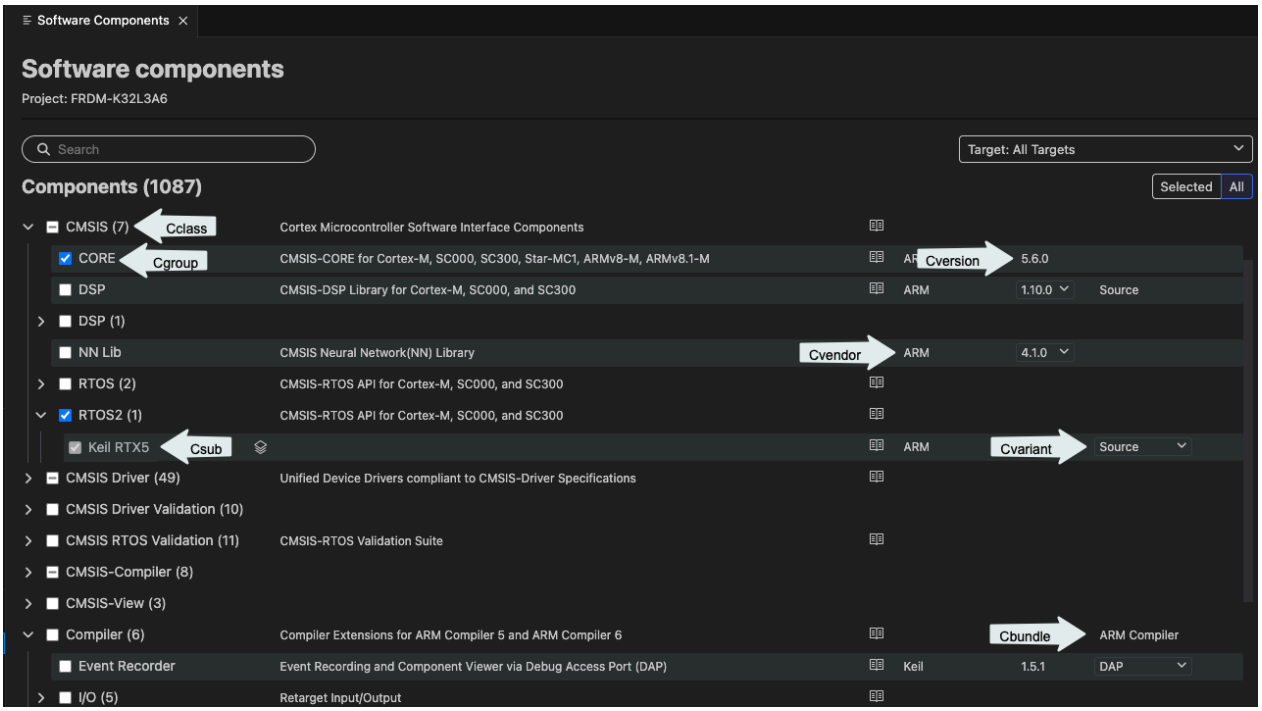
The **Software Components** view opens.

The default view displays the components included in the active project only (**Selected** toggle button). If you click the **All** toggle button, all the components available for use display.

You can use the **Search** field to search the list of components.

With the **Target** drop-down list, you can select components for the different target types that you have in your solution or for all the target types at once.

Figure 6-1: The 'Software Components' view showing all the components that are available for use



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**)
- 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 sub-group (Csub): A component sub-group that is used when multiple compatible implementations of a component are available (for example, **Keil RTX5** under **CMSIS > RTOS2**)
- Component variant (Cvariant): A variant of the software component is typically used when the same implementation has multiple top-level configurations, like **Source** for **Keil RTX5**
- Component vendor (Cvendor): The supplier of the software component (for example, **ARM**)
- 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).

Layer icons  indicate which components are used in layers. In the current version, layers are read-only, so you cannot select or clear them from the **Software Components** view. Click the layer icon  of a component to open the `*.clayer.yml` file or files associated.

Documentation links are available for some components at the class, group, or sub-group level. Click the book icon  of a component to open the related documentation.

6.4.3 Modify the software components in your project

You can add components from all the packs available, not just the packs that are already selected for a specific project.

Procedure

1. Click the **All** toggle button to display all the components available.
2. Select a specific target type in the **Target** drop-down list or, if you want to modify all the target types at once, select **All Targets**.
3. Use the checkboxes to select or clear components as required. For some components, you can also select a vendor, variant, or version.

   The `cproject.yaml` file is automatically updated.
4. Manage the dependencies between components and solve validation issues from the **Validation** panel.

   Issues are highlighted in red and have an exclamation mark  next to them. You can remove conflicting components from your selection or add missing component dependencies from a suggested list.
5. If there are validation issues, move your cursor over the issues in the **Validation** panel to get more details. You can click the proposed fixes to find the components in the list. In some cases, you might have to choose between different fix sets. Select a fix set in the drop-down list, make the required component choices, and then click **Apply**.

   If a pack is missing in the solution, a “Component’s pack is not included in your solution” message displays in the **Validation** panel. An error also displays in the **Problems** view. To install the missing pack, right-click the error in the **Problems** view and select the **Install missing pack** option. If there are several packs missing, use **Install all missing packs**. You can also install missing packs with the **CMSIS: Install required packs for active solution** command from the Command Palette.

   There can be other cases such as:

   - A component you selected is incompatible with the selected device and toolchain
   - A component you selected has dependencies which are incompatible with the selected device and toolchain

- A component you selected has unresolvable dependencies In such cases, you must remove the component. Click **Apply** from the **Validation** panel.

6.4.4 Undo changes

In the current version, you can undo changes from the **Source Control** view or by directly editing the `cproject.yaml` file.

6.5 Modify compiler and linker options

You can manually specify compiler and linker options in the YAML files of your solution as explained in the Open-CMSIS-Pack documentation.

You can add a `compiler:` node in your `*.csolution.yaml` or `*.cproject.yaml` files. You can also add a `linker:` node in your `*.cproject.yaml` or `*.clayer.yaml` files.

The CMSIS-Toolbox build process uses the YAML files and software packs, as well as the source code, configuration files, and linker scripts that you provided to generate a build output.

7. Work with Mbed projects

Keil® Studio Cloud supports Mbed™ OS 5.12 and newer, and Mbed OS 6.

Keil Studio Cloud also provides limited support for Mbed 2. You can build and run Mbed 2 projects, but you cannot debug them. 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.

7.1 Create, import, or clone an Mbed project or a standalone library

Describes how to work with Mbed™ projects and standalone libraries in Keil® Studio Cloud.

You can:

- Create a project from an example shipped with Keil Studio Cloud (examples are available for Mbed OS 5, Mbed OS 6, and Mbed 2). You can also create a 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 create a standalone library in a similar way.

You can also:

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

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

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

Procedure

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

   The **New Project** dialog box displays.
2. Click the **Example project** drop-down list and select an example from the list.
   Each example has a `README.md` file that explains the details of that example.

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

3. In the **Project name** field, edit the name of the new project or keep the default name.
4. Check the default options:
   - Keil® Studio Cloud sets the newly created project as the active project. Build and run commands apply only to the active project. Clear the checkbox if you do not want to make the new project active.
   - Keil Studio Cloud 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 Cloud creates the project in your workspace.

7.1.2 Create a blank Mbed project or a standalone library

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. You can create a standalone library the same way.

Procedure

1. To create an empty folder that is not associated with an existing project, right-click an empty area in the **Explorer** and select **New Folder**.
   The **New Folder** dialog box opens.
2. Type a name for the folder and click **OK**.
   An empty folder is created.
3. Right-click the folder and select **Set Active Project**.
4. Right-click the folder again and select **Add Mbed Library...**.
   The **Add Mbed Library** dialog box opens.
5. If you want to create a blank Mbed project, add a copy of Mbed OS to convert your folder to an Mbed project as follows:
   a. In the **URL** field, enter `https://github.com/ARMmbed/mbed-os` and click **Next** (the format `git@github.com:ARMmbed/mbed-os.git` also works).
   b. From the drop-down list, select the branch or tag of Mbed OS that you want to use. You can use any supported version, including patch versions.

      Release tags are listed on the Mbed OS releases page.
6. If you want to create a standalone library, add a copy of the library as follows:
   a. In the **URL** field, enter the URL for a library from os.mbed.com or from a Git hosting service and click **Next**.
   b. From the drop-down list, select the branch or tag that you want to use.
7. Click **Finish**. The selected branch or tag is added to the folder.

7.1.3 Import an Mbed OS project or a standalone library from your file system

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

Procedure

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

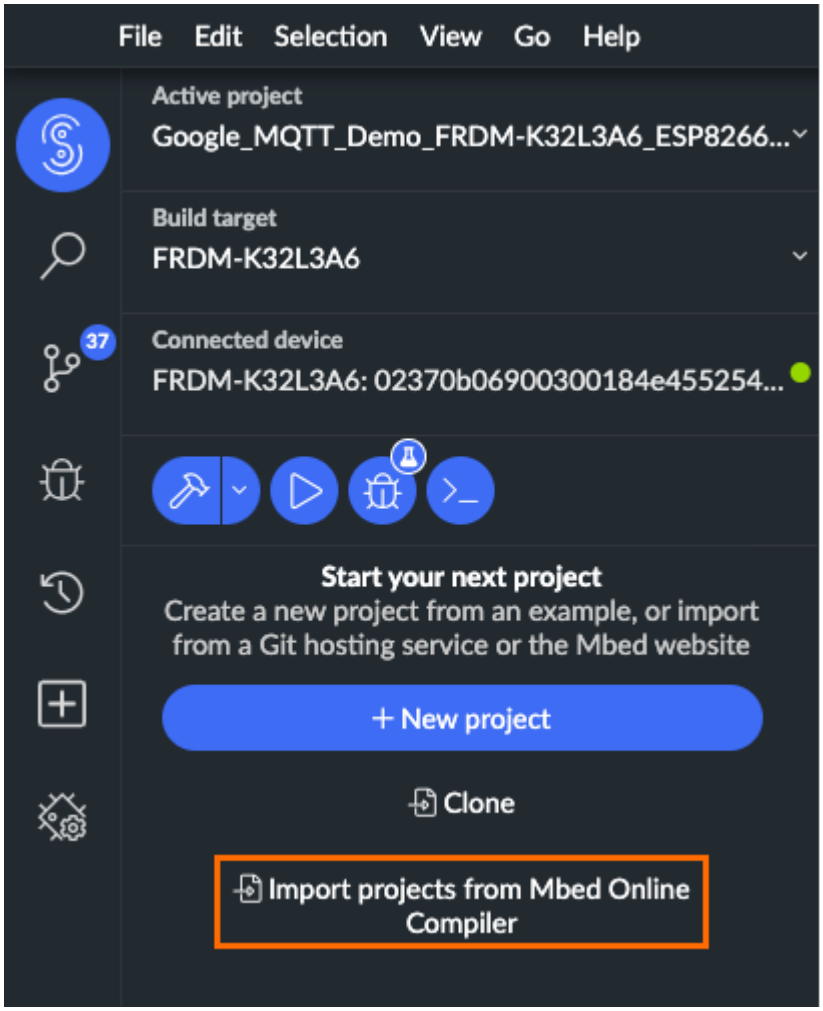
7.1.4 Import Mbed projects or standalone libraries from your Mbed Online Compiler workspace

The Mbed™ Online Compiler is deprecated but you can still import Mbed projects or libraries stored in your Mbed Online Compiler workspace to Keil® Studio Cloud. You can either import all your projects and libraries at once, or import them one by one. The import process creates copies.

Procedure

1. Open the **Copy Programs from Mbed Online Compiler** dialog box:
    - If you have already used Keil Studio Cloud and have projects in your workspace, go to **File** > **Import from Mbed Online Compiler...**.
    - Alternatively, if it is the first time that you have logged in, your workspace is empty and the **Explorer** view displays by default. Click **Import projects from Mbed Online Compiler** from the **Explorer** view.

Figure 7-1: Explorer view and Import projects from Mbed Online Compiler button



The **Copy Programs from Mbed Online Compiler** dialog box displays.

2. Select the first checkbox to import copies of all your projects and libraries, or select them one by one.
3. Click **Copy programs**.
   A progress bar indicates the progress of the import. A message and status icon display for each project or library. If a copy fails, move your cursor over the  icon to get more details on the import.
4. Click **Finish** and check your imported projects or libraries in the **Explorer** view.

Next steps

If you have imported Mbed 2 programs, 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.

Warning

Keil Studio Cloud lets you import non-valid Mbed projects. You must check and, if necessary, fix the structure of your Mbed projects in Keil Studio Cloud when the import is finished to be able to build the projects or run them on your hardware.

For a project to be valid, its **mbed** or **mbed-os** Mbed library folder must be directly under the top-level project folder (not inside a subfolder). For example:

Figure 7-2: Valid Mbed 2 project

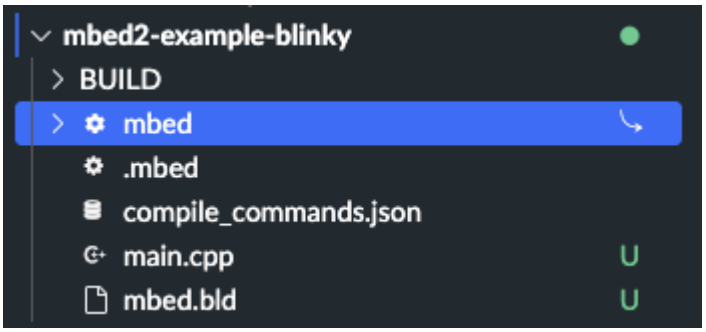
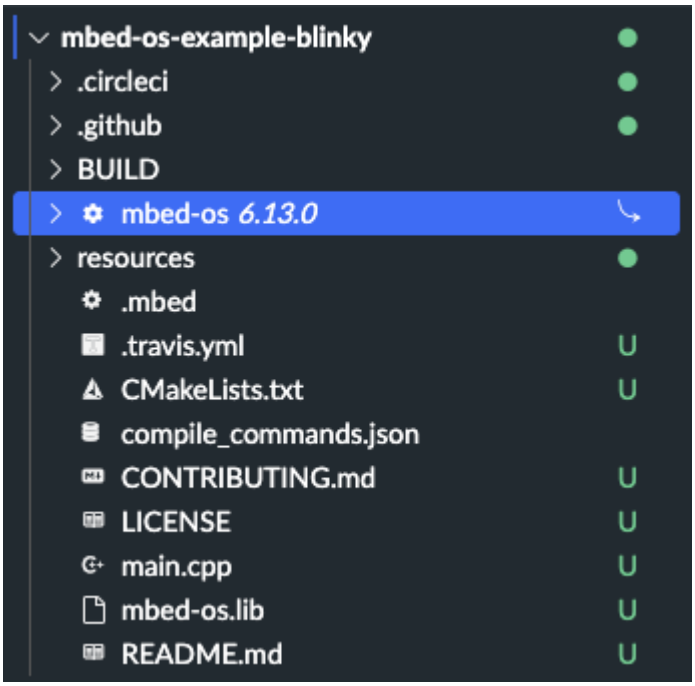


Figure 7-3: Valid Mbed OS 6 project



7.1.5 Clone an Mbed project or a standalone library

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

Before you begin

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

For more information about what cloning a repository means, see git-clone on git-scm.com.

Procedure

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

   Keil® Studio Cloud sets the newly cloned project or library as the active project by default.
3. Clear the checkbox if you do not want to make the new project active.
4. Click **Add project**.
   For Mbed projects, Keil Studio Cloud clones the project with the version of Mbed OS that it was originally created with.

   You can push changes back to the remote repository from Keil Studio Cloud. For more information, see Source control. For libraries with .lib files, see Source-control library updates.

7.1.6 Next steps

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

See Manage an Mbed project and its libraries or standalone libraries for more information.

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

7.1.7 Further resources about Mbed OS

You are now ready to work on your project.

You can find out more about Mbed OS APIs and programming on the Mbed OS documentation site.

When you are ready, build and run your project.

7.2 Manage an Mbed project and its libraries or standalone libraries

Describes how to manage Mbed™ projects and their libraries, or standalone libraries.

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

- When you create an example Mbed project, including an empty example, Keil® Studio Cloud adds the version of Mbed OS that you selected when creating the project.
- When you import or clone an existing Mbed project, Keil Studio Cloud 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 Add a 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 Cloud 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. See Change the Mbed version for your project. However, Arm® recommends that you upgrade to Mbed OS 5 or 6 to benefit from all the features. See Upgrade from Mbed 2 to Mbed OS 5 or 6 for more details.

7.2.1 Mbed OS 5 or 6 project

Describes how to manage the libraries in an Mbed™ OS 5 or 6 project.

7.2.1.1 Open the Mbed Libraries view

Libraries are stored inside folders and identified by a cogwheel icon in the **Explorer**. The **Mbed Libraries** view lists all the libraries in an active project. From the **Mbed Libraries** view, you can see library details, such as the library version and where it was imported from. You can also add or remove libraries, and change the versions that you have.

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 has dependencies with another library, it is indented under the main library.

7.2.1.2 Add a library

Describes how to add libraries to an existing project.

About this task

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

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

Procedure

1. Set the project that you want to work on as the active project.
2. To add a library, you can:
   - Select **View** > **Mbed Libraries** to open the **Mbed Libraries** view and click the **Add new library** button +.
   - Right-click the project from the **Explorer** view and select **Add Mbed Library...**.

   The **Add Mbed Library** dialog box opens.
3. Enter the URL for a library from os.mbed.com or from a Git hosting service.
4. Modify the library name as required and click **Next**.
5. Select the release, branch, or tag that you want to import in the drop-down list.

- For Mbed™ OS libraries, 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.

6. Click **Finish**.
   You can monitor the progress of the import with the progress indicator in the bottom right-hand corner of the screen. When the import is complete, the library displays in the **Mbed Libraries** view and in the **Explorer** view.

7.2.1.3 Remove a library

Describes how to remove a library from a project.

Procedure

1. Set the project that you want to work on as the active project.
2. Select **View** > **Mbed libraries** to open the **Mbed Libraries** view.
3. Move your cursor over the library that you want to remove and click the **Remove library** button  that displays in the right-hand corner.
   The **Remove library** dialog box opens.
4. Click **Remove**.

7.2.1.4 Change the version of a library

Describes how to change the version of a library.

About this task

When you import a library to Keil® Studio Cloud, 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 for each 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.

If you update a top-level library with dependencies, then all the libraries referenced by the main library are updated at the same time.

Procedure

1. Set the project that you want to work on as the active project.
2. To change the version of a library:
   - Select **View** > **Mbed libraries** to open the **Mbed Libraries** view and click the blue tag with the library version (**Change library version**) for the library that you want to update.

     Figure 7-4: Tag example

   - In the **Explorer** view, right-click the library that you want to update, and select **Change Library Version**.

The **Update <selected-library> to Version** dialog box opens.

3. Click the drop-down menu and select the version of the library that you want to use.

- For Mbed™ OS libraries, 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.

4. Click **Update library**.
The selected version of the library starts to check out. You can monitor the progress of the checkout with the progress indicator in the bottom-right corner of the screen.

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

7.2.1.5 Fix library problems

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

To view details about the required fixes, click the **Fix all problems** button.

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

Procedure

1. In the top right-hand corner of the **Mbed Libraries** view, click **Fix all problems**  .

   The **Fix all libraries** dialog box opens with details about all the required fixes.
2. Click **Fix all**. Alternatively, to fix a specific library, click **Fix problems** for the library that you want to fix.

7.2.1.6 Source control library updates

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

The `.lib` file only identifies the commit that you are using; the file does not include branch information. In Keil® Studio Cloud, 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.

7.2.2 Mbed 2 project

Describes how to manually change the version of Mbed™ for your Mbed 2 project. Also describes how to upgrade to Mbed OS 5 or 6, which is the recommended approach.

7.2.2.1 Change the Mbed version for your project

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

Procedure

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

7.2.2.2 Upgrade from Mbed 2 to Mbed OS 5 or 6

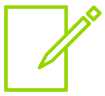
Arm® recommends that you upgrade your Mbed™ 2 projects to Mbed OS 5 or 6 to benefit from all the features.

About this task

The main steps are as follows:

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

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



Optionally, you can enable the bare-metal profile for your project to use Mbed without an RTOS. See Using the bare-metal profile for more information.

Procedure

1. Select **File** > **New...** > **Mbed Project**.
   The **New Project** dialog box opens.
2. Click the **Example project** drop-down list, and then look for the Mbed 2 `mbed2-example-blinky` example and select it.
3. Click **Add project**. Keil Studio Cloud 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** > **Preferences** > **Open Settings (UI)** 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 **URL** field, paste the following URL: https://github.com/ARMmbed/mbed-os (it 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 Cloud imports the `mbed-os.lib` library.
10. Check that an appropriate build target displays in the **Build target** drop-down list. If not, select one.
11. Click **Build project** (hammer)  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.

7.3 Configure compile-time customizations

Describes how to configure compile-time customizations in Keil® Studio Cloud.

The Arm® Mbed™ OS configuration system, a part of the Arm Mbed OS build tools that underpin Keil Studio Cloud, customizes compile-time configuration parameters.

Each library can define various configuration parameters in its `mbed_lib.json`. The project-level `mbed_app.json` can 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 are:

- 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.

8. Build and run a project

Build a project or run a project on your hardware with Keil® Studio Cloud.

8.1 Connect your hardware

Describes how to connect your hardware for the first time.

Procedure

1. To open the **Device Manager**, click the **Connected device** area under the **Build target** drop-down list, or click **Device Manager** in the activity bar.
2. Connect your hardware to your computer.
3. Click **Add Device**. Select the device firmware for your hardware in the dialog box that displays at the top of the window, and then click **Connect**.

   Your hardware displays in the **Device Manager**.

   You can now use your hardware to run and debug a project.

   After the first successful connection, Keil® Studio Cloud automatically detects the hardware.

Next steps

If you need to add more hardware, click **Add Device** in the top right-hand corner.

8.2 Edit your hardware

If your board cannot be detected or if you are using an external debug probe, you can edit the hardware entry from the **Device Manager**. You can specify a Device Family Pack (DFP) and a device name retrieved from the pack to be able to work with your hardware. DFPs handle device support.

Procedure

1. Move your cursor over the hardware that you want to edit and click **Edit Device**.
2. Edit the hardware name in the field that displays at the top of the window if needed, and then press **Enter**. This is the name that displays in the **Device Manager**.
3. Select a Device Family Pack (DFP) CMSIS-Pack for your hardware from the drop-down list.
4. Select a device name to use from the CMSIS-Pack in the field and press **Enter**.

8.3 Remove hardware

You can remove hardware from the **Device Manager**.

Procedure

Move your cursor over the hardware that you want to remove, and then click **Forget Device**  .

8.4 Build and run a project on your hardware

Describes how to build and run a project.

Procedure

1. Ensure that the project that you want to build and run is set to active.
   To make the project active, right-click the project name in the **Explorer** view and select **Set Active Project**.
2. Check that an appropriate build target displays in the **Build target** drop-down list. If not, select one.
   The build target tells Keil® Studio Cloud how to build the real-time operating system (RTOS) so that it matches your hardware.
   - For Mbed™ projects, you must manually select an appropriate build target for a particular project the first time you work with that project.
   - For CMSIS solutions, you must manually select a build target.
3. Select one of these options:
   - **Build project**: Builds the project, without running it on your hardware.
     - **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 build target, 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 list next to the hammer): Ignores all previous builds. A clean build is slower than reusing a previous build, but guarantees that your build has no old artifacts.
   - **Run project**  : If a binary is available, Keil Studio Cloud runs the project on the hardware. If there is no binary, Keil Studio Cloud builds the project and then runs it on the hardware.


Note

For boards with multiple cores, you must select the appropriate processor for your project in the **Select a processor** drop-down list that displays at the top of the window when you click **Run project**. Note that the **Select a processor** drop-down list does not display when the **daplink** option is selected for the **Flash Mode** setting.

Keil Studio Cloud uses the release build profile for deploying projects (build and run) and the debug build profile for debugging. Note that you cannot change the build profiles from the UI.

To stop a build at any time, click **Cancel** in the notification that displays in the bottom right-hand corner of the screen.

For Mbed projects only: After your project is built, you can use the **Output** view to check how much flash memory, RAM (**Memory**), and ROM your program uses, and how much flash memory and RAM is still available. See Output view for more details.

8.5 Output view

The output of a build displays in the **Output** view.

To open the **Output** view, select **View** > **Output**.

To turn auto scrolling off, click  . When 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 turn auto scrolling on, click the icon again.

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 **Ctrl+mouse click** (Windows) or **Cmd+mouse click** (macOS) to follow the error or warning to the relevant part of the code.

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

9. IntelliSense code editing

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.

9.1 Code editing

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

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

A build target must display in the **Build target** drop-down list to get all IntelliSense features.

9.1.1 Navigate your code

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 Cloud.

Quick file navigation

To follow a link included in your code or to find a file where a code element has been defined, press and hold down **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

When you move your cursor over elements in your code, a pop-up message displays additional information about those elements.

Go to Declaration and Peek Declaration

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

Move your cursor 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.

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.

Move your cursor 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

Go to Implementations and Peek Implementations

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

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

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. Move your cursor over the code element, right-click, and select one of the options.

Go to Symbol

Describes how to use the **Go to Symbol...** option in Keil® Studio Cloud 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...**. Alternatively, select **Go** > **Go to Symbol in Editor...**.

   The search bar displays the available symbols in the file you are currently working on.
2. Select a symbol to go to that symbol.

Note

You can also search for symbols in the workspace with the **Go** > **Go to Symbol in Workspace** 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  in the right-hand side 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.

Show AST

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

The abstract syntax tree opens in the Side bar below the workspace tree view. It shows how the clangd language server parses the code.

9.1.2 Edit and refactor your code

Describes the auto-completion, renaming, format-correcting, and refactoring functionality that is available 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 Cloud 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.

To trigger auto-complete and help for existing code, press **Ctrl+Space**.

Rename an element

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 that you are currently working on.

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

Code formatting

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

- **Format Document**: The indentation corrects in the whole file.
- **Format Selection**: Only the lines of code that you selected indent correctly.

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, and 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:

- To display corrections, click an element underlined with a squiggly line.

  A light bulb icon appears next to the problematic line. Click the light bulb and check the correction suggested. If you are happy with the suggestion, click the pop-up message.
- Use the **Peek Problem** and **Quick Fix...** options to correct your code. Move your cursor 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 pop-up message to apply a correction.

Problems view

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

To use the **Problems** view:

1. Go to the **View** menu and select **Problems**, or click **Problems** in the status bar.
2. To collapse the list of problems in the **Problems** view, click **Collapse All**  .

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

Refactoring

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

To refactor your code:

1. Select the source code that you want to extract, right-click the code, and select **Refactor...**:
   - If you are extracting a function, an **Extract to function** pop-up message appears. Source code fragments can be extracted into a new function.

- If you are extracting a variable, an **Extract subexpression to variable** pop-up message appears. You can create a new local variable for the currently selected expression.

2. To refactor the code, click the pop-up message for your function or variable extraction.

Alternatively, click the light bulb next to the code that you want to refactor.

9.2 Code linting

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

Keil® Studio Cloud 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 light bulbs to suggest fixes in the editor.

9.2.1 Enable clang-tidy

Clang-tidy is not enabled by default.

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 the active project and select **New File**.
2. In the **New File** dialog box, type `.clang-tidy` and click **OK**.
   Keil® Studio Cloud 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: '*'`.

9.2.2 Configure clang-tidy checks

For each of your projects, you can use a `.clang-tidy` file to configure the checks that clang-tidy must carry out.

9.2.2.1 Enable or disable checks

To 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.

9.2.2.2 Configure advanced custom checks

This section 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.

10. Manage files

Describes how to search the content of files, compare, upload, download, and move files in Keil® Studio Cloud.

10.1 Find a file

Describes how to find a file in the workspace.

Procedure

1. Go to the **Go** menu and select **Go to File...**.
   A drop-down list displays and shows the recently opened files by default.
2. Type the name of the file that you are looking for.
   The list of files is updated as you type.
3. When you have found your file in the list, click the file to open it in the editor.

10.2 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 Cloud searches through both saved and unsaved changes.

10.2.1 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 **Search** 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 **Collapse All**  .

If the **Search** view is hiding the **Explorer** view, click the Keil® Studio Cloud icon, or press **Ctrl+Shift+E** (Windows) or **Cmd+Shift+E** (macOS) to return to it.

Example 10-1: Search examples

- To specify which files, folders, or path segments to include in or exclude from 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. Note that with the **Edit** > **Replace in Files** option, you can directly open the **Search** view with the **Replace** field visible.

  To replace all instances of the text in all files, click **Replace All**  .

  To replace all instances of the text in a specific file, move the cursor over the file name in the search results, and click **Replace All** near the file name.

10.2.2 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 search one part of the file only, enter text in the search box, then select the lines that you want to check and click **Find in selection**  .
3. To replace the searched-for text with new text, click **Toggle Replace**  and enter your new text. The replace option is also available from **Edit** > **Replace**.
4. To replace a single instance or all instances of the text in the file, click **Replace**  or **Replace All**  .

10.2.3 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.

Wildcard	Description
`[`	Matches zero or more path segments. For example, `tools]{.ui}` 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`.

10.3 Compare files

Keil® Studio Cloud 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 Cloud displays both files in a single tab, with highlighted differences.

Related information

Compare local and remote versions in Git source control on page 70
Compare local and remote versions in Mercurial source control on page 81

10.4 Upload and download files or projects

Describes how to upload or download a file or project in Keil® Studio Cloud.

Procedure

- Either:
  - **Upload**: You can use the **File** > **Upload Files...** option to upload files to the root directory of a project. The option displays in the menu only after you have clicked a project in the **Explorer**. 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.

10.5 Change file locations

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

Before you begin

For Mbed™ projects only - You cannot move the `mbed-os` library for a project. The `mbed-os` library must remain in the root folder of the project.

To create a folder before adding files or folders to it, do one of the following:

- Right-click a project and select **New Folder**.
- Select a project in the **Explorer** and go to **File** > **New Folder**.

Procedure

1. To move a file or a folder, select it, then right-click and select **Move to...**.
   The **Move to...** dialog box opens.
2. Select where you want to put the file or folder. To put the file or folder in the workspace root directory, select the **workspace** entry.
3. Click **Open**.

You can also drag and drop files to move them around.

10.6 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**.

11. Source control

Describes how to use Keil® Studio Cloud with the Git and Mercurial source control technologies. Also describes how to use the **History** view to view your commit history.

11.1 Work with Git

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

The flow of Keil Studio Cloud 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.

If you are not yet familiar with Git, check Get started with Git.

11.1.1 Get started with Git

If you are new to Git, read the Work with Git source control tutorial tutorial. The official git-scm.com website is also a good place to start. You can check the Books and videos available.

You can find other useful videos and content in the Visual Studio Code documentation.

11.1.2 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 Cloud and link your Arm® account and your GitHub account.

Procedure

1. Go to the **User Profile** view by clicking **Accounts** in the activity bar and selecting **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 Cloud application to connect with your GitHub account. Read the authentication terms, and if you accept them, 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 Cloud application to be used with your GitHub account. Read the authentication terms, and if you accept them, click **Authorize <github-account-name>**.

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

Related information

Manage accounts on page 12
User Profile view on page 12

11.1.3 Interface and features reference

Source control in Keil® Studio Cloud 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:

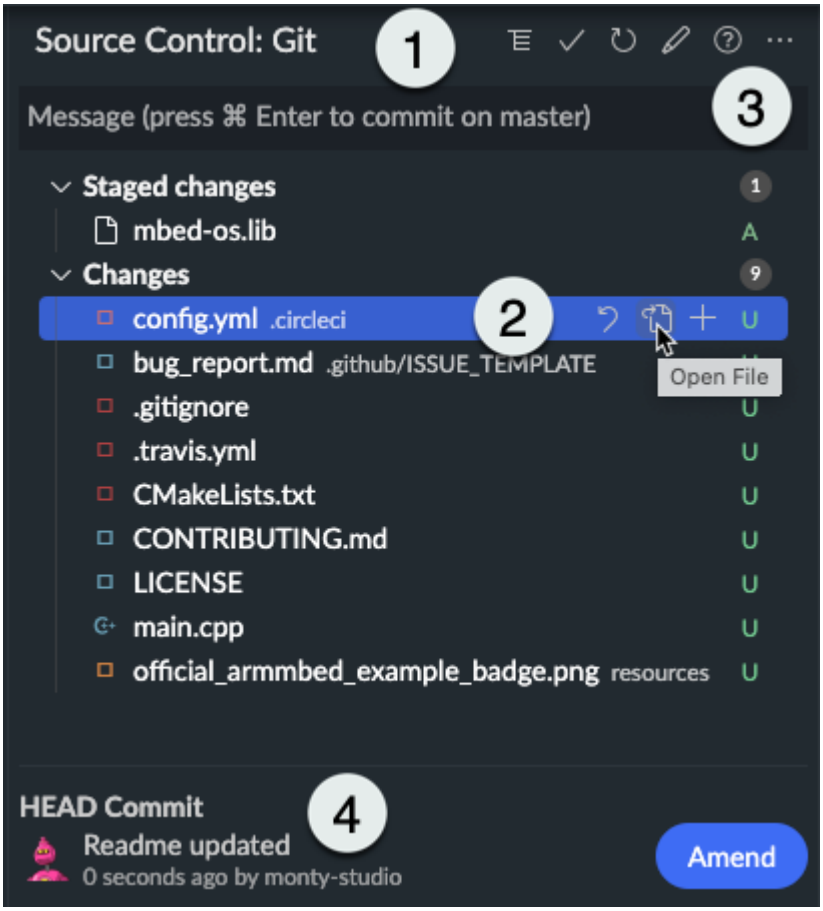
1. The actions menu.

   The following buttons allow you to display your changes as a list or as a tree view:

   - **Toggle to List View** 
   - **Toggle to Tree view** 

   Click the **...** button for more actions.
2. The following options are available for each file when you move your cursor over a file name: Discard Changes, Open File, Stage Changes (**+**), and Unstage Changes (**-**).
3. Nine 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 to their starting state if they have never been pushed).
Local changes	Stage changes and Stage all changes	Use the **Stage** option for files that 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 changes and Unstage all	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, it 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 check out an existing branch. The default branch for a newly set repository is `main`.

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. It 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.

11.1.4 Configure a project for source control and collaboration

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

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. 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 Cloud

For cloned projects:

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

11.1.4.1 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 Cloud. To learn how to connect your GitHub account with Keil Studio Cloud, 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 initialize the project for Git (turn your project into a Git repository) without publishing it yet. Click **...** at the top of the **Source Control** view and select **Initialize the Active Project**.
     b. You can also publish the project (the publishing process initializes the project for Git and creates a remote repository on GitHub to publish your changes). Click **Publish Project** or click **...** at the top of the **Source Control** view, and then select **Publish Project**.
   - If the project is already initialized for Git, click **...** 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 Cloud 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 possible only once for each new project. Setting a remote repository is not possible for a project that you clone from Git. The remote repository is automatically set to the repository you cloned it from.

11.1.4.2 Fork a Git project

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

Procedure

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

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

11.1.5 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 Cloud allows creating branches, and synchronizing 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 `main` 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 the branch.

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

11.1.6 Manage local files

Describes the different options to manage your local files and explains how to update the remote.

11.1.6.1 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

11.1.6.2 View changes

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

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 and what has changed.

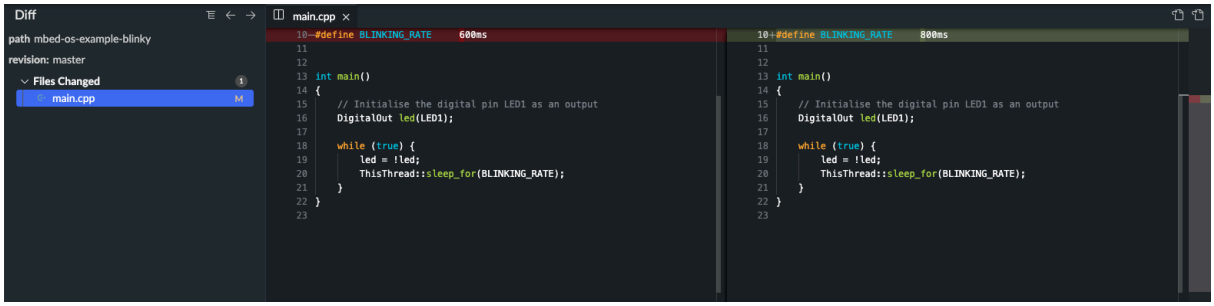
Figure 11-2: Double-click a file to view its side-by-side comparison

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

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

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

  Figure 11-3: List of files changed in the folder



Next steps

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

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

You can open several files at once for editing: press and 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.

11.1.6.3 Stage, commit, and push changes

Describes how to stage, commit, and push changes in Keil® Studio Cloud.

About this task

Use **staging** to manually control which of your edited files are added to each commit. This process 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. Move your cursor over the file that you want to commit and click **+** to stage your changes. The file is listed under **Staged changes**. To stage several files at once, press and hold **Shift** or else **Cmd** (macOS) or **Ctrl** (Windows and Linux) and click the files that 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**.

- 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 will 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 for which you do not have permission and a fork already exists, you will 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.

11.1.6.4 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.

11.1.6.4.1 Change the last commit

Describes how to change your last commit in Keil® Studio Cloud.

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. Alternatively, to reuse the last commit message and only update the content of the commit, press **Enter**.

11.1.6.4.2 Change multiple commits

Describes how to change multiple commits in Keil® Studio Cloud.

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 **Amend** 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 **Unamend** if you change your mind and prefer to keep a commit untouched. This option resets the head commit to the commit you have just "unamended". If there are several commits to "unamend", click **Unamend all commits** (**-**).
2. When you are happy with the changes, enter a commit message for the staged changes in the **Message** box.
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 **Clear amending commits** (**x**). The files staged while amending the commits remain in the **Staged changes** list.

11.1.6.5 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 Cloud does not list any Mbed OS files.

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

For more information about how `.gitignore` files work and how to use them, see Ignoring Files on git-scm.com.

11.1.7 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 the following information:

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

You can click the **Synchronize Changes** indicators to display the available synchronization options.

Synchronization options

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

To display the available options, click **Synchronize Changes**:

- **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. When you are happy with the corrections on a particular file, click **+** to stage your changes.

For more information about merge conflicts, see About merge conflicts in the GitHub Help.

11.2 Work with Mercurial

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

The Mercurial flow in Keil Studio Cloud:

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

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 Cloud. You can work with all public repositories, and any private repository to which you have access through os.mbed.com.

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

11.2.2 Interface and features reference

Source control in Keil® Studio Cloud 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:

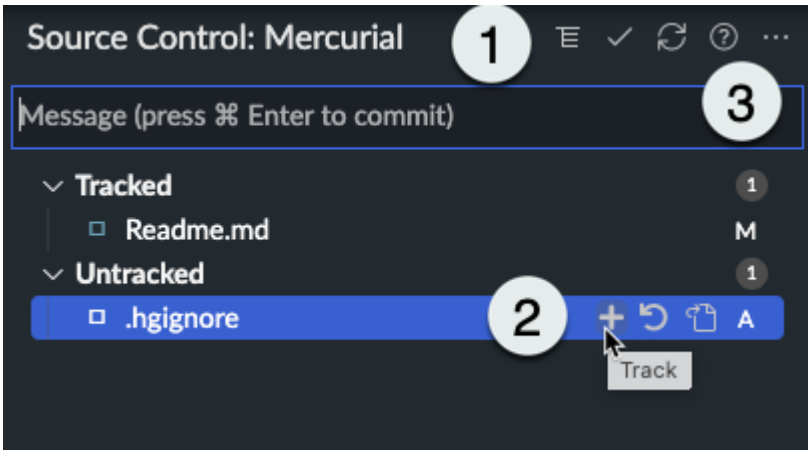
1. The actions menu.

   The following buttons allow you to display your changes as a list or as a tree view:

   - **Toggle to List View**:
   - **Toggle to Tree view**:

   Click **...** for more actions.
2. The buttons available on each file when you move your cursor 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 it was committed. One untracked file - an **Added** (A) file, which has not been committed yet.

Figure 11-4: The 'Source Control' view for Mercurial



Group	Features	Comments
Local changes	Track and Track all files	Use the **Track** option to monitor the files for changes. A tracked file with changes is automatically added to a commit. You can move files back to the **Untracked** list only if you have never committed them (these files are marked with "A", for "Added"). After 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.

Group	Features	Comments
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.
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.
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 particular project (that is to say, point to an existing remote repository without any commits). You must create remote repositories directly through os.mbed.com.

For cloned projects:

A project cloned from Mercurial automatically points to the remote repository you cloned the project from. With Keil® Studio Cloud, you can convert a project cloned 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 **Set remote URL**.

      The **Set Remote URL** dialog box opens.

c. Enter a URL for your remote repository.

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 possible only once for each new project.

11.2.3.2 Convert a Mercurial project to Git

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

Procedure

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

About this task

Mercurial uses branches to isolate your work from the work of other people or from code that must remain stable. Keil Studio Cloud allows creating branches, and synchronizing 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.

11.2.5 Manage local files

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 11-5: Double-click a file to view its side-by-side comparison

- Open the file for editing by clicking **Open File** in the top right-hand corner of the screen. To go back to the changes, click **Open Changes**.

  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 Cloud.

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. Move your cursor over the file that 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**.
   The **History** view shows the committed changes.
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 Cloud does not list any Mbed™ OS files.

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

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. 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 must decide which part you want to keep or merge the contents yourself, and then remove all the merge conflict markers. When you are happy with the corrections on a 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 particular commit, click the eye icon  . The commit information includes the commit message, date, and a list of changed files.

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

12. Monitor and debug

Describes the **Serial Monitor** view, and how to debug supported development boards in Keil® Studio Cloud.

12.1 Use the Serial Monitor view

The **Serial Monitor** view displays the output of your board.

If you are connecting your hardware for the first time, follow these steps:

1. Connect your hardware over USB.
2. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
3. Click the **Add Device** button and select the device firmware for your hardware in the dialog box that displays at the top of the window, and then click **Connect**.

   Your hardware displays in the **Device Manager**. After the first successful connection, Keil® Studio Cloud automatically detects the hardware.

If you are having trouble getting serial output, try disconnecting and reconnecting the board.

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 Cloud.

12.1.1 Access the Serial Monitor view after a first successful connection of your board

To open or reopen the **Serial Monitor** view, choose one of the following options.

Procedure

1. Choose one of the following options:
   - From the **Explorer** view, click **Open serial monitor**  .
   - From the **Device Manager**, move your cursor over the hardware for which you want to open a serial monitor and click **Open Serial**  .
2. Select a serial port in the dialog box that displays at the top of the window, and then click **Connect**.
   A drop-down list displays at the top of the window where you can select a baud rate. The baud rate is the data rate in bits per second between your computer and your hardware. To view the output of your hardware correctly, you must select an appropriate baud rate. The baud rate you select must be the same as the baud rate of your active project.

3. Select a baud rate.
   The **Serial Monitor** view opens with the baud rate selected.

Next steps

To modify the baud rate, follow these steps:

1. Click **Change active device Serial baud rate** in the status bar and select a baud rate from the drop-down list.

   Figure 12-1: Change active device Serial baud rate icon

2. Select a serial port in the dialog box that displays at the top of the window, and then click **Connect**.

12.2 Debug a project with Keil Studio Cloud

Describes the debugger mode and debugging tools available in Keil® Studio Cloud.

12.2.1 Introduction

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

For more information about supported hardware, see Supported development boards and MCUs and Supported debug probes.

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.

12.2.2 Start a debug session

Explains how to start a debug session.

Before you begin

If you are having problems running an Mbed™ project on an ST board, set the **Connect Mode** preference to **underReset** to be able to run the project and debug it. See Advanced debugger settings for more details on the different connect modes.

Procedure

1. Set the project that you want to debug as the active project.
2. Connect your hardware over USB.
3. If it is the first time that you are connecting your hardware, follow these steps:
   a. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
   b. Click **Add Device** and select the device firmware for your hardware in the dialog box that displays at the top of the window, and then click **Connect**.

   Your hardware displays in the **Device Manager**. After the first successful connection, Keil® Studio Cloud automatically detects the hardware.
4. Check that an appropriate build target displays in the **Build target** drop-down list. If not, select one.
5. Click **Debug**  .
   Keil Studio Cloud automatically builds and runs your project on the connected board. The **Debug** view displays and the debug session starts. The debugger stops at the `main()` function of your application.

12.2.3 Restart or stop the debugger

Explains how to restart or stop the debugger.

Procedure

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

12.2.4 Navigate your code using the step buttons

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

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

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

12.2.5 Set breakpoints

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

Before you begin

By default, Keil® Studio Cloud stops on the first line of `main()`. To change this behavior, select **File** > **Preferences** > **Open Settings (UI)** and search for **Run To Main** in the **Debug** category. To keep the program running until the first user breakpoint, clear the **Run until the start of the main function on connect** checkbox.

Procedure

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

Next steps

- After 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 already disabled). You can also remove a breakpoint by clicking that breakpoint in the margin.
- You can also remove, disable, and enable breakpoints by using the options in the **Breakpoints** panel:
  - **Disable/Enable Breakpoints**  : Disable or enable all breakpoints at once. Note that if you have disabled all your breakpoints, you cannot then activate individual breakpoints by clicking them.
  - To disable or enable an individual breakpoint, clear or select the checkbox next to the breakpoint in the list
  - **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**

12.2.6 Set function breakpoints

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

Procedure

1. Click the **Breakpoints** panel header, and then click **Add Function Breakpoint**  .
   The **Add Function Breakpoint** dialog box opens.
2. Type the name of the function that you are looking for (function names are case-sensitive) and click **OK**.

12.2.7 Examine threads and call stacks

You can examine and work with threads in the code that you are debugging. Working with threads is useful for debugging multithreaded applications. Each thread has a call stack and appears on a separate row in the **Threads** list. In the current version of Keil® Studio Cloud, only one thread displays in the **Threads** list (**Main**).

Procedure

- When you select the **Main** thread, the call stack displays in the **Call Stack** list. The call stack shows the order in which methods and functions are called. The call stack list is a good way to examine and understand the execution flow of your code.
- When you 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. This option allows you to copy a call stack to the clipboard to further investigate a problem.

12.2.8 Inspect variables

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

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

To see the current value of a property when the debugger is paused, move your cursor over the object. You can also view variable values in the **Variables** list.

The **<not in scope>** value 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}`. To see the global variables defined for a module, click that module.

Only modules with global variables are visible.

You can display variable values inline in the editor with the **Inline Values** setting. Go to **File** > **Preferences** > **Open Settings (UI)** and search for **Inline Values** in the **Debug** category.

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.

12.2.9 Use watch expressions

Watch expressions are expressions that you want to evaluate and monitor while debugging, such as the value of a variable or a condition.

To add a watch expression:

1. Click the **Watch** panel header, and then click **Add Expression**  .

   The **Edit Watch Expression** dialog box opens.
2. You can type any valid expression, such as a variable name or a condition.
3. Click **OK**.
4. Start a debug session.

   Your watch expression is evaluated and updated every time the execution pauses or steps.

More options are available from the **Watch** list:

- **Collapse All**  : Collapse all expressions
- **Remove All Expressions**  : Remove all expressions at once

- When you right-click a watch expression from the list, the following options are available: **Edit Expression**, **Copy Expression Value**, **Remove Expression**, **Remove All Expressions**

12.2.10 Inspect registers

The **Registers** list displays register contents for the detected CPU.

The following tables provide register details for the Arm®v6, Arm®v7, and Arm®v8 architectures.

Armv6

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

Armv7

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

Armv8

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

Register category	Description
FPU	FPU (Floating-Point Unit) is shown only if the Floating Point (FP) extension is enabled. Shows FP extension registers: thirty-two 32-bit single-precision registers (S0 to S31) and sixteen 64-bit double-precision registers (D0 to D15). Also shows selected bits of the Floating-Point Status and Control Register (FPSCR). See the Armv8-M Architecture Reference Manual for more details on FPSCR. Values are shown in both binary and numeric format.
FPU-MVE	This category is shown only if the MVE (M-Profile Vector Extension) is enabled (not the FP extension). MVE uses registers of the FP extension.
MVE	This category is shown when both the FPU and MVE extensions are enabled. The MVE extension provides operations on various SIMD data types.
Secure	This category is shown only when the Security (TrustZone®) extension is enabled. When the core is in Secure state, it can access both Secure and Non-secure memories. If the Security extension is enabled, the processor starts up in Secure state.
Non-Secure	This category is shown only when the Security (TrustZone) extension is enabled. When the core is in Non-secure state, it can access Non-secure memories only.
Internal	Shows CPU state details. **Mode** can be "Thread" or "Handler". **Privilege** shows the CPU code execution and can be "Privileged" or "Unprivileged". **Stack** shows the currently selected stack pointer (MSP or PSP).

12.2.11 Check peripherals

Silicon vendors provide System View Description (CMSIS-SVD) files, which describe the peripheral registers of devices. Silicon vendors distribute their descriptions as part of CMSIS Device Family Packs (DFP). Keil® Studio Cloud uses CMSIS-SVD files in the background to display peripheral details. For more information on SVD, see the official CMSIS-SVD documentation.

The **Peripherals** list shows information about the peripheral registers of your hardware during a debug session.

From this list, you can:

- View peripheral register property values
- View additional information about a specific property when you move your cursor over it
- Copy property values with **Copy Value** 
- Change property values at runtime with **Update Value** 

There are also options to pin a peripheral register to the top of the list  or refresh property values  .

12.2.12 Debug with an Arm Mbed LPC1768 board

If you are using an LPC1768 board, you must update the firmware of your board to the latest version and add some configuration in your project before being able to debug it.

Procedure

1. Connect your board to your computer.

2. If it is the first time you are connecting your board, follow these steps:
   a. Click the **Connected device** area under the **Build target** drop-down list to open the **Device Manager**.
   b. Click the **Add Device** button and select the device firmware for your board in the dialog box that displays at the top of the window, then click **Connect**.

      Your board displays in the **Device Manager**. After the first successful connection, Keil® Studio Cloud automatically detects the board.
3. Set the project that you are working on as active and make sure that the correct build target is selected in the **Build target** drop-down list.
4. Download the latest firmware from the Firmware LPC1768 LPC11U24 page and follow the instructions on that page to update the firmware of your board (power-cycle the board).
5. Create an `mbed_app.json` file at the root of the project.
6. Add the following configuration in the `mbed_app.json` file:

```
{
    "target_overrides": {
        "LPC1768": { "target.device_has_remove": ["SEMIHOST",
 "LOCALFILESYSTEM"] }
    }
}
```

7. Save the file.
8. Click **Debug**  .
   Keil Studio Cloud automatically builds and runs your project on the LPC1768 board. The **Debug** view displays and the debug session starts. The debugger stops at the `main()` function of your application.

12.2.13 Advanced debugger settings

The following tables describe advanced settings for the debugger. Go to **File** > **Preferences** > **Open Settings (UI)**. The **Debug** settings are listed under the **Features** heading, and the **Run** settings are listed under the **Extensions** heading.

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

For flash download, you can either:

- Use the DAPLink (**daplink** option for the **Flash Mode** setting 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 (**cmsis** option). 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.


Note

There are also options to change how panels display 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: • auto (default): The debugger decides which connect mode to use based on the connected hardware. For ST boards, when auto is selected, underReset is used. For other boards, haltOnConnect is used. If you were already using Keil Studio Cloud, you have to manually select auto for this setting. • haltOnConnect: Stops the CPU of the board for a reset before the flash download or the debug session. • underReset: Asserts the hardware reset during the connection. • preReset: Triggers a hardware reset pulse before the connection. • running: Connects to the CPU without stopping the program execution during the connection.
Reset After	Performs a reset operation as defined in Reset Mode after connecting to the Connect board.
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: Uses the ResetSystem sequence from the CMSIS-Pack. • hardware: Uses the ResetHardware sequence from the CMSIS-Pack. • processor: Uses the ResetProcessor sequence from the CMSIS-Pack.
Run First	If selected, a flash download is triggered when you start a debug sesssion. To save connection time, the flash download happens only if the code has changed, if the active project has changed, or if the board has been disconnected and reconnected.
Verify Application	Compares the content of the target memory with the program loaded in the debugger. Enable this option to make sure 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 Cloud.

Flash download options in the **Run** category:

Setting	Description
Erase Mode	The erase modes available if you are not using DAPLink. The options are: • sectors (default): Erase only the sectors to be programmed. • full: Erase full chip. • none: Skip flash erase step.
Flash Mode	The flash modes available. The options are: • auto (default): The debugger decides which flash mode to use based on the connected hardware. With auto, DAPLink is used by default if the connected hardware uses a DAPLink firmware. Otherwise, cmsis is used. • cmsis: If selected, flash algorithms contained in the CMSIS-Pack that is used in your project are deployed in the background to do the flash download. • daplink: If selected, the DAPLink firmware does the flash download of programs to your board.
Program flash	The Flash program option if you are not using DAPLink. Writes code into the flash memory.
Reset Run	The reset option if you are not using DAPLink. Triggers a hardware reset after the flash download.

Setting	Description
Verify flash	The flash verify option if you are not using DAPLink. Verifies the content downloaded to the flash memory during the flash download.

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

Setting	Description
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 only the first time that you start a debug session.
Open Debug	Controls when the **Debug** view displays. The options are: • neverOpen: The **Debug** view never displays. • openOnSessionStart (default): The **Debug** view displays when ever a debug session starts. • openOnFirstSessionStart: The **Debug** view displays only the first time that you start a debug session.

12.2.14 Switch to RAM debugging

Explains how to switch from flash debugging to RAM debugging.

Before you begin

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

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

No specific settings are needed to do RAM debugging in Keil Studio Cloud, but you must configure your program in a certain way.

The following example shows how to configure your project for RAM debugging.

Some settings used for flash debugging might affect how your RAM debugging program runs. Check the advanced settings available for the debugger under **File** > **Preferences** > **Open Settings (UI)**:

- Under **Features** > **Debug**, make sure that **Run First** is selected.
- Make sure that you select either `auto`, `system`, or `processor` from the **Reset Mode** drop-down list.
- Under **Extensions** > **Run**, make sure that **Flash Mode** is set to **auto**.

Procedure

1. Clone the Blinky example for the FRDM-K32L3A6 board from keil.arm.com. Under **Get Project**, click **Open in Keil Studio Cloud** to clone the project.
2. In Keil Studio Cloud, set the example as the active project. Right-click the project and select **Set Active Project**.
3. Open the `FRDM-K32L3A6.cproject.yml` file included in the project.
4. In the `files` list, go to this entry:

```
- file: ./RTE/Device/K32L3A60VPJ1A_cm4/K32L3A60xxx_cm4_flash.scf
```

5. Replace it with this entry to point at the `FRDM-K32L3A6_cm4_ram.scf` file in the project (this file is provided by default in the **RTE** > **Device** folder):

```
- file: ./RTE/Device/K32L3A60VPJ1A_cm4/K32L3A60xxx_cm4_ram.scf
```

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

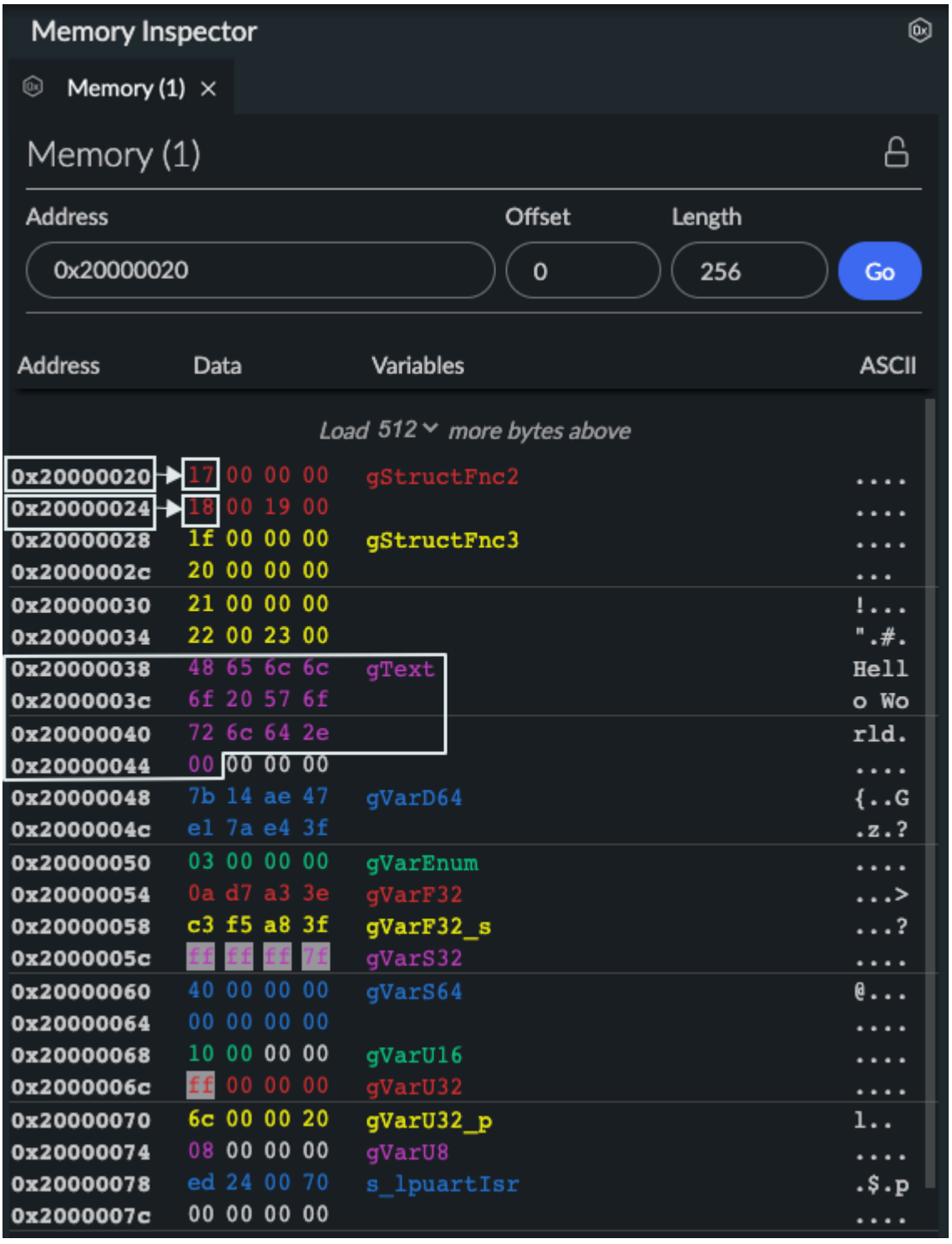
12.3 Use the Memory Inspector view

Describes the **Memory Inspector** 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 Inspector** 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

Figure 12-2: The Memory Inspector view



- The **Address** column shows the addresses of memory locations (or cells in memory) in hexadecimal format. The addresses 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 move your cursor 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 to 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

To monitor the memory usage of your board, you must first start a debug session and set breakpoints.

To start a debug session and check the **Memory Inspector** view:

1. Set the project that you want to debug as the active project and start a debug session.

   The debugger stops at the `main` function of your application.

2. Click **Memory Inspector** in the top right-hand corner of the screen to display the **Memory Inspector** view.

   The **Memory Inspector** view opens with an empty **Memory (1)** tab. You can rename that tab.

3. Set breakpoints on the parts of code that you want to examine.

4. Click **Continue**.

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

5. In the **Variables** list, find the variable that you want to check.

6. Right-click the variable and select **Show variable in memory**.

Note

You can also directly type the address of a variable in the **Address** field and click **Go**. You can use addresses of pointers too.

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

7. 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 Inspector** view displays a memory region that starts at `0x2000006e`. If you type 64 (bytes) in the **Length** field, the **Memory Inspector** view displays 64 bytes from the current memory location, so 4-byte values per line * 16 lines.


Note

You can also display more lines using the **Load more bytes above** and **Load more bytes below** drop-down lists.

8. 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.

   To write a new value to any memory location displayed, type over the value shown in the **Data** column and click **Apply Changes**. 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 must check that 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 that you want to check.
2. Click **Create new memory inspector** 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. Click **Freeze memory view** 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 **Toggle Comparison Widget Visibility**.

   A comparison panel opens at the bottom of the screen.
2. If you have more than two tabs open, select the tabs that you want to compare in the drop-down lists and click **Go**.


Note

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).

13. Supported hardware and Arm Virtual Hardware

Describes the hardware that Keil® Studio Cloud supports and how to run projects on Arm® Virtual Hardware.

13.1 Supported development boards and MCUs

For CMSIS projects, Keil® Studio Cloud supports the following development boards and MCUs.

For Mbed™ projects, Keil Studio Cloud supports the following development boards.

13.2 Supported debug probes

Describes the debug probes that Keil® Studio Cloud supports.

WebUSB-enabled CMSIS-DAP debug probes

Keil Studio Cloud supports debug probes that implement the CMSIS-DAP protocol. See the CMSIS-DAP documentation for general information. There is also an update for CMSIS version 6.

Such implementations include:

- The DAPLink implementation: see the ARMmbed/DAPLink repository
- The Nu-Link2 implementation: see the Nuvoton repository
- The ULINKplus™ (firmware version 2) implementation: see the ULINKplus documentation

ST-LINK debug probes

Keil Studio Cloud 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 Cloud notifies you when older firmware versions are detected but does not stop you from using them. To stay up to date with the latest firmware versions, visit the STSW-LINK007 page.

13.3 Arm Virtual Hardware

Keil® Studio Cloud supports Arm® Virtual Hardware (AVH).

Arm Virtual Hardware provides simulation models that allow you to test and validate a software project without having to connect to real hardware.

For more details on Arm Virtual Hardware, check the product overview.

Arm Virtual Hardware is available for CMSIS projects only. Note that this feature is still under development.

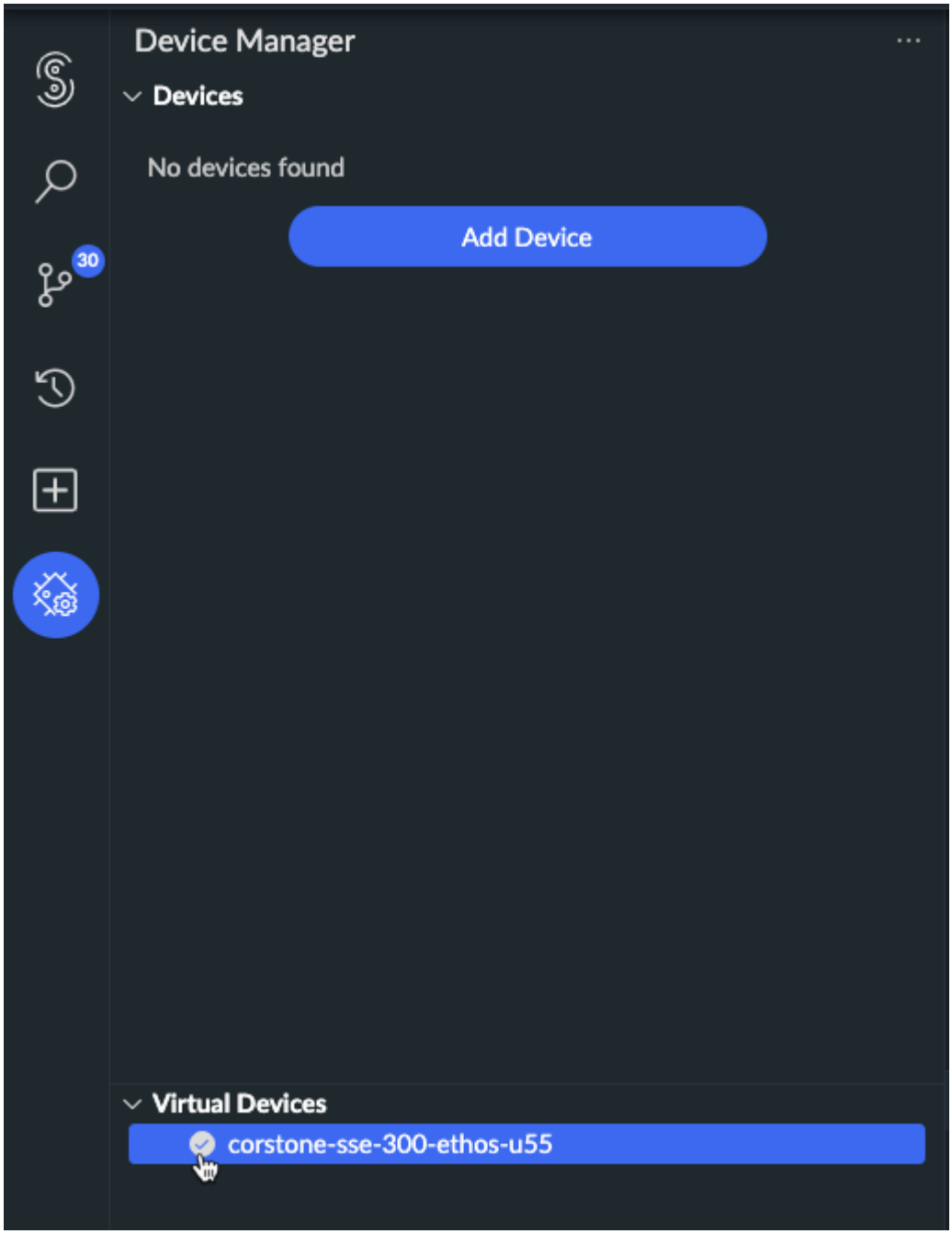
13.3.1 Run a project on Arm Virtual Hardware

Describes how to run a project on Arm® Virtual Hardware in Keil® Studio Cloud.

Procedure

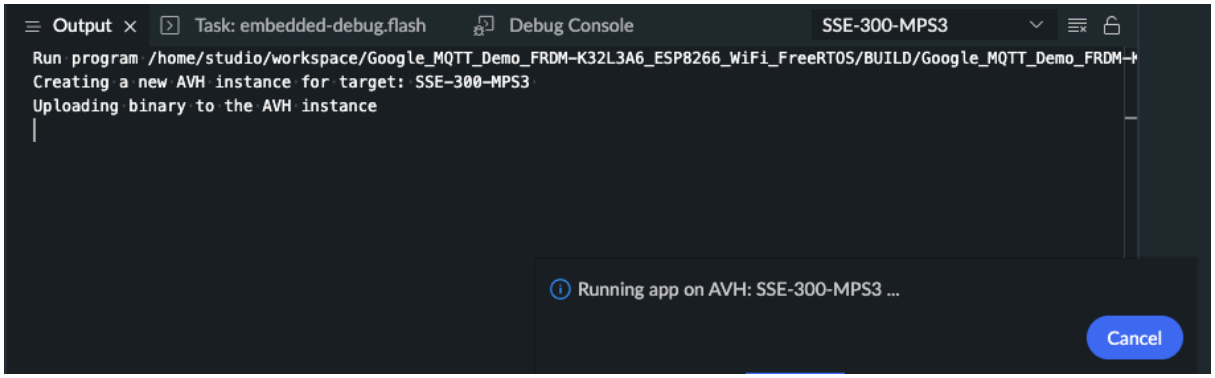
1. Check that your project is set as the active project. Right-click the project and select **Set Active Project**.
2. Click **Device Manager**  in the Keil Studio Cloud activity bar.
3. Click the **Virtual Devices** header below **Devices** to open the panel and select a virtual device.

Figure 13-1: Select a virtual device



4. From the **Explorer**, click **Run project** to build the example and run it on the virtual device. The output of the build displays in the **Output** view.

Figure 13-2: Output



5. Click **Cancel** in the pop-up message that displays in the bottom right-hand corner of the screen to stop the virtual device.

14. Extensions

Keil® Studio Cloud supports a fixed list of Visual Studio Code compatible extensions. Extensions can add support for programming languages, integrate with cloud service providers, or help configure your microcontroller unit (MCU).

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

14.1 Install the AWS Toolkit extension

The **AWS Toolkit** extension is available from the **Extensions** view.

Procedure

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

   After the **AWS Toolkit** extension has been installed, an **AWS** icon displays in the activity bar.
4. Click the **AWS** icon.
   This action opens the AWS Explorer and CDK Explorer. Another pop-up message appears to indicate which version of the Toolkit has been installed. Click **View Quick Start** in the pop-up message to check the `README.md` file and learn about the **AWS Toolkit**. Next, follow the steps to Connect to AWS from Keil Studio Cloud.

Note

- The `README.md` file is also available when you click the extension from the **Extensions** view.
- To uninstall the **AWS Toolkit** extension, switch off the toggle button from the **Extensions** view.

14.2 Connect to AWS from Keil Studio Cloud

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

Before you begin

To create an account, see How do I create and activate a new AWS account?.

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

Procedure

1. Open the **AWS Toolkit** extension in Keil Studio Cloud and go to the AWS Explorer.
   A "Connect to AWS..." message displays.
2. Click the message.
   A pop-up message appears to indicate that no credentials were found and invite you to create credentials.
3. Click **Yes** to create credentials.
   A pop-up message appears and asks you for a credential profile name.
4. Type a profile name and press **Enter**.
   A pop-up message appears and asks you for an AWS Access Key (this is the access key ID and looks like this: `AKIAIOSFODNN7EXAMPLE`).
5. Provide your access key and press **Enter**.
   A pop-up message appears and asks you for an AWS Secret Key (this is the secret access key and looks like this: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`).
6. Provide your secret key and press **Enter**.
   A pop-up message appears and asks you if you want to show the default region your profile is associated with in the AWS Explorer.
7. Click **Yes**.
   The region is added to the AWS Explorer. You can now start using the AWS services and resources available.

15. Known issues and troubleshooting

Describes Keil® Studio Cloud known issues, how to troubleshoot some common issues, and how to contact Arm® to report issues or to suggest enhancements.

15.1 Known issues

Keil® Studio Cloud has the following known issues:

- Some users have reported licensing issues for Arm® Compiler 6. These issues might be related to the system clock and region format.
- On Linux, there is a limit on the number of files that can be watched in the workspace. To increase the limit, follow the instructions available in the Visual Studio Code documentation.
- On Linux, source control management requires the `libcurl4` package to work correctly.

15.2 Troubleshooting

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

15.2.1 Keil Studio Cloud does not load

Keil® Studio Cloud hangs on the loading screen.

Solution

Keil Studio Cloud takes longer to load when you have many tabs opened in your browser and change tabs while Keil Studio Cloud is loading. Stay on the tab where Keil Studio Cloud is loading to solve the issue. You can also try to clear your browser cache to solve the problem.

15.2.2 Cannot log into Keil Studio Cloud

You are unable to log into Keil® Studio Cloud.

Solution

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

15.2.3 Connected development board or debug probe not found

You have connected your development board or debug probe, but the Device Manager cannot detect the hardware.

Solution

- Run **Device Manager** (Windows), **System Information** (Mac), or a Linux system utility tool like **hardinfo** (Linux), and then check for warnings beside your hardware. Warnings can indicate that hardware drivers are not installed. If necessary, obtain and install the appropriate drivers for your hardware.
- On Windows: ST development boards and probes require extra drivers. You can download them from the ST site.
- On Windows: Check if you have an Mbed™ serial port driver installed on your machine. The Mbed serial port driver is required with Windows 7 only. Serial ports work out of the box with Windows 8.1 or newer. The Mbed serial port driver breaks native Windows functionality for updating drivers as it claims all the boards with a DAPLink firmware by default. Arm recommends that you uninstall the driver if you do not need it. Alternatively, you can disable it.

  You can either:

  - Uninstall the Mbed serial port driver (recommended): Open a command prompt as an administrator. Find and delete the `mbedserial_x64.inf` and `mbedcomposite_x64.inf` drivers.

    ```
    pnputil /enum-drivers

    pnputil /delete-driver {oemnumber.inf} /force
    ```

    Then, connect your hardware using a USB cable and open the Windows Device Manager. In `Ports (COM & LPT)` and `Universal Serial Bus controllers`, find the `mbed` entries and uninstall both by right-clicking them. Finally, disconnect and reconnect your hardware.
  - Disable the Mbed serial port driver: Open the Windows Device Manager. In `Ports (COM & LPT)`, find the Mbed Serial Port. Right-click it and select **Properties**. Select the **Driver** tab and click **Update Driver**. Click **Browse my computer for drivers** and then click **Let me pick from a list of available drivers on my computer**. Select `USB Serial Device` instead of `mbed Serial Port`.
- On Linux: udev rules grant permission to access USB boards and devices. You must install udev rules to be able to build a project and run it on your hardware or debug a project.

  Clone the pyOCD repository, then copy the rules files which are available in the udev folder to `/etc/udev/rules.d/` as explained in the README.md file. Follow the instructions in the `README` file.

  After installing the udev rules, your connected hardware is detectable in the Device Manager. You might still encounter a permission issue when accessing the serial output. If this is the case, run `sudo adduser "$USER" dialout`, and then restart your machine.
- Check that the firmware version of your board or debug probe is supported and update the firmware to the latest version. See Out-of-date firmware for more details.

- Your board or device might be claimed by other processes or tools (for example, if you are trying to access a board or device with several instances of Keil Studio Cloud, or with Keil Studio Cloud and another IDE).
- Activate the **Manage All Devices** setting. This setting allows you to select any USB hardware connected to your computer. By default, the Device Manager gives you access only to hardware from known vendors.
  1. Go to **File** > **Preferences** > **Open Settings (UI)**.
  2. Find the **Device-manager: Manage All Devices** setting and select its checkbox.

15.2.4 Out-of-date firmware

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

Solution

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

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

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

For more information on firmware updates, see also the Debug Probe Firmware Update Information Application Note.

15.2.5 Connection fails when clicking Run project or Debug project

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

Solution: check your hardware setup

Cables:

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

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

Jumpers and switches:

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

For example:

- Jumpers to select or connect to the power supply (for example, there can be different USB connectors or external power supply units)
- Multiple jumpers to enable, connect, or configure different subsystems (core, peripherals, GPIO pad) or to 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 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 Out-of-date firmware for more details.

External debug probe:

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

Solution: check the connect mode or target device boot mode

Problems can happen with code that was flashed to your device using a specific build target. The code can cause some debug connectivity issues.

Problems can occur when:

- One or more device-specific low-power modes are used which affect the debug connectivity. For example, this is 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 or the CPU or both in reset mode during the debug connection.
- If the target device does not support **underReset** connections, then change the connect mode to **preReset**. Depending on the project that is running, **preReset** can delay the execution of problematic parts of code long enough to establish a debug connection. However, the success of this depends 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 of connecting than a slow debug unit.
- Boot the target device in a different boot mode. This process can require changing on-board switches or jumpers or both and disconnecting and reconnecting the board. For some boards (for example, the NXP i.MX RT boards), you can change the boot mode to a system boot mode.

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

- If you are using a DAPLink debug probe, select the **daplink** option for the **Flash Mode** setting 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 Cloud to your computer and use this alternative way of programming.

Solution: check that your device supports the SWD protocol

You can use an on-board debugger or an external debug unit if your board provides a suitable debug connector or provisions for it. Note that Keil Studio Cloud currently supports only the SWD protocol, not JTAG.

15.2.6 Development board not showing serial data

Serial data for a development board is not being shown in Keil® Studio Cloud.

Solution

- Disconnect and reconnect the board.
- Check that no other project is accessing the serial data on the board; the device can connect to only one serial monitor at a time.

- On Linux, run `sudo adduser "$USER" dialout` to check that the current user belongs to the `dialout` group.

15.2.7 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

Remove spaces from the project name and compile again.

15.3 Submit feedback

If you have suggestions or you have discovered an issue with Keil® Studio Cloud, please contact us. Go to the keil.arm.com support page and use the links provided in the **Keil Studio Cloud** category.

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

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

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

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

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

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
17-11	25 July 2024	Non-Confidential	Various updates
17-10	20 June 2024	Non-Confidential	Various updates
17-09	29 February 2024	Non-Confidential	Introduction updated
17-08	31 January 2024	Non-Confidential	1.7.15 updates
17-07	20 December 2023	Non-Confidential	Mbed Online Compiler chapter removed.
17-06	5 December 2023	Non-Confidential	1.7.13 updates
17-05	31 October 2023	Non-Confidential	1.7.12 updates
17-04	19 October 2023	Non-Confidential	1.7.11 updates
17-03	28 September 2023	Non-Confidential	1.7.9 updates
17-02	6 September 2023	Non-Confidential	1.7.7 updates

Issue	Date	Confidentiality	Change
17-01	20 July 2023	Non-Confidential	1.7.0 updates

Change history

For information about the technical changes to Arm Keil Studio Cloud, see the Release Notes. Go to **Help** > **Release Notes**.

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	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	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	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or harming yourself.

This information is important and needs your attention.

A useful tip that might make it easier, better or faster to perform a task.

A reminder of something important that relates to the information you are reading.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm Keil Microcontroller Development Kit (MDK) Getting Started Guide	109350	Non-Confidential
Arm Keil Studio Visual Studio Code Extensions User Guide	108029	Non-Confidential
µVision User Guide	101407	Non-Confidential