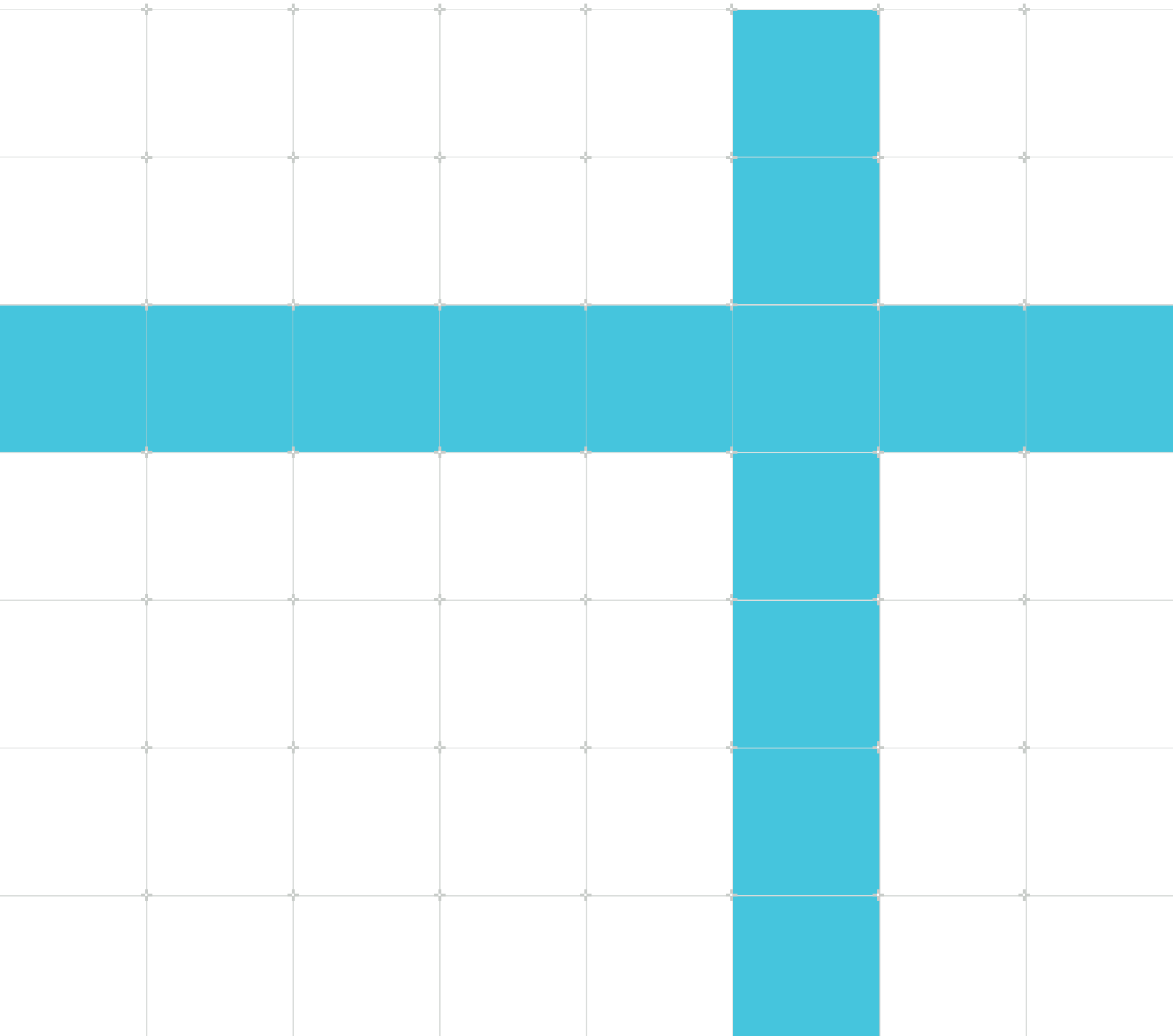
Arm® Keil® Studio Cloud CMSIS environment

Version 2.0.0

User Guide

Issue 01
109811_2.0.0_01_en



Arm® Keil® Studio Cloud CMSIS environment User Guide

This document (109811_2.0.0_01_en) was issued on 2024-11-28. There might be a later issue at https://developer.arm.com/documentation/109811

The product version is 2.0.0.

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.

We believe that this document contains no offensive language. 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. Log in to Keil® Studio Cloud

You can access Keil® Studio Cloud using an Arm® or Mbed™ account.

Procedure

1. In your web browser, go to https://studio.keil.arm.com/auth/login.
2. Select the environment that you want to use. To use the Keil® Studio Cloud CMSIS environment for microcontroller development using the Common Microcontroller Software Interface Standard libraries, components, and tools, select **CMSIS**.
3. Enter the email associated with your Arm or Mbed account.
4. Click **Log in**.

Note

For details of the Keil Studio Cloud Classic environment (Mbed and CMSIS), see the Keil Studio Cloud User Guide.

2. Get started with an example project

Start working with an example.


Note

This section describes working with example solution or µVision projects that you can get from keil.arm.com. If you open a µVision project, Keil® Studio Cloud converts it automatically to csolution format and installs any missing packs. Keil® Studio Cloud also initializes Git. Any projects that have the `AC5` compatibility label only use Arm Compiler 5, which does not support automatic conversion. As a workaround, update Arm Compiler 5 projects to Arm Compiler 6 in Keil µVision, then convert the projects to csolutions in Visual Studio Code. See Download a Keil µVision example for more information.

You can also create solutions from scratch, or convert your existing µVision projects to solutions. For more information, see Create a solution and Convert a Keil µVision project to a solution.

- Use an example solution project from keil.arm.com (recommended).
- Download a Keil µVision `*.uvprojx` project from keil.arm.com and convert it to a solution (alternative).

When you are ready:

- Build the example project.
- Explore what you can do with Keil® Studio Cloud:
  - Choose a context for your solution.
  - Look at the Solution outline.
- Connect your board and run the example on the board.
- Start a debug session.
- Check the serial output.
- Work with Git source control.

2.1 Import a solution example

Import a solution example or download a zip file that contains the solution.

Procedure

1. Go to keil.arm.com.
2. Click the **Hardware** menu and select **Boards**.
3. Search for your board and select it in the **Suggested Boards** list.
4. Find a project in the **Projects** tab.

The `Keil Studio` compatibility label indicates that the example is compatible with Keil® Studio Cloud.

5. Move your cursor over **Get Project**, and then click **Open in Keil Studio Cloud** to import the solution example.

2.2 Download a Keil µVision example

When you download and open a Keil® µVision® `*.uvprojx` project, Keil® Studio Cloud automatically converts it to a solution. Note that conversion does not work with Arm® Compiler 5 projects. You can download Arm Compiler 5 projects from the website, but you cannot use them with Keil® Studio Cloud. Only Arm Compiler 6 projects can be converted. As a workaround, update Arm Compiler 5 projects to Arm Compiler 6 in Keil µVision, then convert the projects to solutions in Keil® Studio Cloud. For more information, see the Migrate Arm Compiler 5 to Arm Compiler 6 application note and the Arm Compiler for Embedded Migration and Compatibility Guide.

Procedure

1. Go to keil.arm.com.
2. Connect your board over USB and click **Detect Connected Hardware** in the bottom right-hand corner.
3. Select the device firmware for your board in the dialog box that displays at the top of the window, and then click **Connect**.
4. Click the **Board** link in the pop-up message that displays in the bottom right-hand corner.

   The page for the board opens. Example projects are available in the **Projects** tab.
5. Move your cursor over the **Get Project** button for the project that you want to use and click **Download .zip** to download the Keil µVision `*.uvprojx` example.
6. Unzip the example and drag and drop the folder in the **Explorer** of Keil® Studio Cloud.

   Keil® Studio Cloud converts the project automatically.

The `*.cproject.yml` and `*.csolution.yml` files are available next to the `*.uvprojx` in the **Explorer**  .

2.3 Build the example project

Check that your example project builds. You can build your project from the **Solution outline**, or from the Command Palette.

Procedure

1. Build the project:
   - From the **Solution outline**:

a. Click **CMSIS**  in the Activity Bar.

   The **Solution outline** opens. A **Build solution** icon is available in the **Solution outline** header.

b. Click **Build solution**  .

   The **Rebuild solution** option is also available with **Views and More Actions**  .

- From the Command Palette: You can also trigger **Build solution** and **Rebuild solution** with the **CMSIS: Build** and **CMSIS: Rebuild solution** commands.

2. Check the **Terminal** tab to find where the ELF file (`.axf`) was generated.

2.4 Work with Git source control

This section explains the basics of Git source control in Keil® Studio Cloud using the CMSIS Blinky example available for the NXP FRDM-K32L3A6 board.

Start by setting up 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

2.4.1 Set up your project for source control

Learn how to work with Git.

Before you begin

Learn how to publish your changes to a new GitHub repository.

You need:

- A GitHub account. Create an account on github.com or, if you already have an account, sign in.

  Click **Accounts** in the Activity Bar to check the connection. If you are not already signed in to your GitHub account, Keil® Studio Cloud prompts you to connect when you use the source control features. Alternatively, open the Command Palette and select **KSC: Connect GitHub Account**.

- The Blinky example project for the NXP FRDM-K32L3A6 board.


Note

If you do not have the example yet, you can download it from keil.arm.com, or from https://github.com/Arm-Examples.

Procedure

1. Go to the **Source Control** view.
2. Initialize the project for Git from the Command Palette with the **Git: Initialize Git Repository** Command.
3. Click **Publish to GitHub**.
4. If you are not already signed in to your GitHub account, click **Allow** when prompted to sign in using GitHub.
5. A drop-down window opens at the top of the page, allowing you to choose whether to create a public repository or a private repository. Select **Publish to GitHub private repository**.
   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.

2.4.2 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.
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.
   Git updates `my-branch` with your changes.
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 **...** > **Branch** > **Merge...** next to the example name. Select `my-branch` in the list to merge the changes done on `my-branch` into main.
10. Click **...** > **Push** to update the remote repository.

2.5 Choose a context for your solution

A context is the combination of a target type, known as a build target, and a build type, known as a build configuration. This context is specific to a particular project in your solution.

Read Set a context for your solution for more details.

2.6 Look at the Solution outline

The **Solution outline** presents the content of your solution in a tree view.

Read Use the Solution outline for more details.

2.7 Connect your board

Connect your board. See Supported hardware for more details on the development boards, MCUs, and debug probes supported.

Procedure

1. Click **Device Manager**  in the Activity Bar to open the Device Manager .

2. Connect your board to your computer over USB.

   The Device Manager detects the board and displays a pop-up message.
3. Click **OK** in the pop-up message to use the hardware.

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

2.8 Run the solution on your board

Run the solution project on your board.

Procedure

1. Click **CMSIS** in the Activity Bar.
   The **Solution outline** opens. A **Run** icon is available in the **Solution outline** header.
2. Click **Run**.
   The project runs on the board.
3. Check the **Terminal** tab.

2.9 Start a debug session

Start a debug session.

Procedure

1. Click **CMSIS** in the Activity Bar.
   The **Solution outline** opens. A **Debug** icon is available in the **Solution outline** header.
2. Click **Debug**.
   The **Run and Debug** view displays and the debug session starts. The debugger stops at the `main()` function of your project.
3. To see the debugging output, check the **Debug Console** tab.

Next steps

Look at the Visual Studio Code documentation to learn more about the debugging features available in Visual Studio Code.

3. 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

3.1 Get started with Git

If you are new to Git, read Work with Git source control. 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.

3.2 Set credentials for GitHub

Working with Git-based hosting services requires authentication against those services.

Click **Accounts** in the Activity Bar to check the connection.

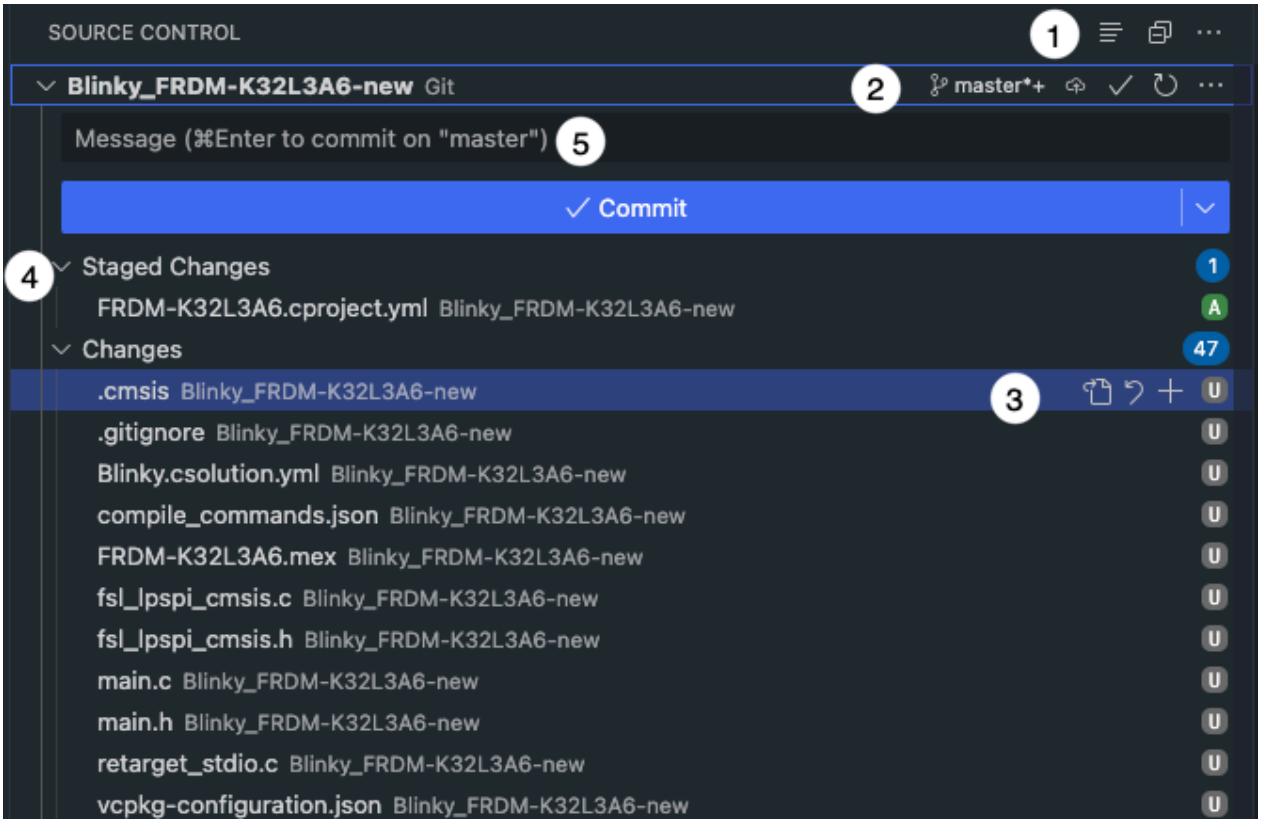
If you are not already signed in to your GitHub account, Keil® Studio Cloud prompts you to connect when you use the source control features.

Alternatively, click **View** > **Command Palette** and select **KSC: Connect GitHub Account**.

3.3 Interface and features reference

Handling current work in source control in Keil® Studio Cloud is managed in the **Source Control** view. The **Source Control** view shows the projects that have been initialized for Git.

Figure 3-1: The 'Source Control' view for Git.



The image highlights:

1. The actions available from the **Source Control** header:
   - **View as Tree** or **View as List**: Display your changes in a tree view or in a list view. These options are also available from the **View & Sort** menu under **Views and More Actions**.
   - **Collapse All Repositories** or **Expand All Repositories**.
   - **Views and More Actions**: Choose different display options for the **Source Control** view.
2. The actions available at the project level:
   - Branch currently checked out.
   - **Publish to GitHub**: Publish to a new GitHub repository for the first time.
   - **Commit**: Commit your changes on the branch that you currently have checked out.
   - **Refresh**: Update the list of local changes.
   - **More Actions**: View more actions (see Features available from the More Actions menu).

3. The options available when you move your cursor over a file name: Open File, Discard Changes, Stage Changes (**+**), or Unstage Changes (**-**).
4. Staged files and changed files.
5. The **Message** box where you enter commit messages.

3.3.1 Features available from the More Actions menu

This table describes the menu items and available from the **More Actions** menu in more detail.

Menu command or group	Features	Comments
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, push your changes to the remote first before pulling.
Push		Send new local commits to the remote.
Clone		Make a full copy of a remote repository on your local machine.
Checkout to...		Check out a specific branch.
Fetch		Fetch new content from a remote repository.
Commit	Commit, Commit Staged, Commit All	The commit is the record you can push to the remote repository. The Commit command commits staged changes and prompts you to stage all changes before committing if you have not already done so. The Commit Staged command commits only the changes that you have staged. The Commit All command commits all changes in your working directory, including both staged and unstaged changes.
Commit	Undo Last Commit	Undo your last local commit.
Commit	Abort Rebase	Undo a rebase and return your branch to its previous state.
Commit	Commit (Amend), Commit Staged (Amend), Commit All (Amend)	Change your last commit. The Commit (Amend) command amends the most recent commit with any changes that are currently staged. If no changes are staged, you can modify the commit message of the last commit. The Commit Staged (Amend) command specifically amends the most recent commit with only the changes that are currently staged. The Commit All (Amend) command amends the most recent commit with all changes, both staged and unstaged.
Commit	Commit (Signed Off), Commit Staged (Signed Off), Commit All (Signed Off)	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. The Commit (Signed Off) command commits all changes in your working directory, whether they are staged or not, and signs off the commit with your GPG (GNU Privacy Guard) key. The Commit Staged (Signed Off) command commits only the changes that have been staged and signs off the commit with your GPG key. The Commit All (Signed Off) command stages all changes in your working directory and then commits them in one step, signing off the commit with your GPG key.
Changes	Stage All Changes	Staged changes are included in your next commit.
Changes	Unstage All Changes	Unstaged changes are in Git but not marked for commit.

Menu command or group	Features	Comments
Changes	Discard All Changes	Undo all of your current local changes.
Pull, Push	Sync	Synchronize your local and remote repositories.
Pull, Push	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, push your changes to the remote first before pulling.
Pull, Push	Pull (Rebase)	Apply the changes from a remote repository and integrate them with the local branch using rebase instead of merge.
Pull, Push	Pull from...	Pull from a specific branch.
Pull, Push	Push	Push your local changes to a remote repository.
Pull, Push	Push to...	Push to a specific remote, instead of the current one.
Pull, Push	Fetch	Fetch changes from the remote, but do not merge them into the current local branch.
Pull, Push	Fetch (Prune)	Fetch changes from the remote, but remove stale references to remote branches that no longer exist on the remote repository.
Pull, Push	Fetch From All Remotes	Automatically fetch all changes from all remotes.
Branch	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	Rebase Branch...	Take all the changes that were committed on one branch and replay them on a different branch.
Branch	Create Branch...	Create a new local branch or check out an existing branch. The default branch for a newly set repository is `main`.
Branch	Create Branch From...	Create a branch from another branch.
Branch	Rename Branch...	Change a local branch name.
Branch	Delete Branch...	Delete a branch.
Branch	Publish Branch...	Push a local branch to a remote repository.
Remote	Add Remote...	Add a remote repository.
Remote	Remove Remote	Remove a remote repository.
Stash	Stash	Set aside your changed files. This command 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	Stash (Include Untracked)	Include untracked files in your stashed files list.
Stash	Apply Latest Stash	Add 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.

Menu command or group	Features	Comments
Stash	Pop Latest Stash	Apply the latest stash and clear the stash.
Stash	Pop Stash...	Apply a specified stash and clear the stash.
Stash	Drop Stash...	Discard all changes in the stash.
Stash	Drop All Stashes...	Remove all the stashed states.
Stash	View Stash...	Show the changes that are recorded in the stash without applying them.
Tags	Create Tag	Create a tag for your commit.
Tags	Delete Tag...	Delete a local git tag.
Tags	Delete Remote Tag...	Delete a remote git tag.
Show Git Output		View the output of git commands.

3.4 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 connect your GitHub account to Keil® Studio Cloud, open the Command Palette and select **KSC: Connect GitHub Account**.

Procedure

Go to the **Source Control** view.

- You can initialize the project for Git without publishing it yet.
  a. Open the Command Palette, search for `Git: Initialize Git repository` and then select it.
  b. Choose the project to initialize from the drop-down list at the top of the page.
  c. Keil® Studio Cloud adds the project to the list in the **Source Control** view.
- 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 to GitHub** in the project header. You can then choose a name for the repository, and whether to make it public or private with the **Publish to GitHub private repository** or **Publish to GitHub public repository** options.

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 with the **Git: Clone** command. The remote repository is automatically set to the repository you cloned it from.

3.5 Create or switch branches

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

About this task

Git uses branches to isolate your work from the work of other people or from code that must remain stable. Keil® Studio 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.

3.6 Manage local files

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

3.6.1 File statuses

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

- **U**: Untracked - a new file in the **Changes** list that is 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

3.6.2 View changes

This section 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, 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.

Next steps

To open the file for editing, click the **Open File** icon next to the tab displaying the changes.

You can open several files at once for editing from the **Source Control** view. Press and hold **Shift** or **Cmd** (macOS) or **Ctrl** (Windows and Linux). Select the files, then right-click and select **Open File**.

3.6.3 Stage, commit, and push changes

This section 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, instead of 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 **Cmd** (macOS) or **Ctrl** (Windows and Linux). 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**.
5. To push the commit to the remote branch, click **More Actions...** > **Push**.

3.6.4 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

Note

If you have pushed your changes to the remote and notice that something is wrong, 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.

3.6.4.1 Change the last commit

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

About this task

Use the **Commit (Amend)** option to 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 so that the file appears as deleted **D** in the **Changes** list, then stage the deleted file.

2. Click **...** > **Commit** > **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**.

3.7 Synchronize

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

To synchronize your changes, select **Git: Sync** from the Command Palette or click **More Actions...** > **Pull, Push** > **Sync**.

This action applies the latest changes from the remote repository to your local repository by doing a pull (fetch and merge), then pushes your changes to the remote repository. This option might generate merge conflicts.

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 2 branches being merged, Git does the following:

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 (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, 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.

4. Manage your hardware

Look at the hardware supported with Keil® Studio Cloud.

Then, manage your hardware with the Device Manager:

- Connect your hardware
- Edit your hardware
- Open a serial monitor

4.1 Supported hardware

This section describes the hardware that the Device Manager supports.

4.1.1 Supported development boards and MCUs

Keil® Studio Cloud supports the development boards and MCUs available on keil.arm.com.

4.1.2 Supported debug probes

The following debug probes are supported.

4.1.2.1 WebUSB-enabled CMSIS-DAP debug probes

Keil® Studio Cloud supports debug probes that implement the CMSIS-DAP protocol, for example:

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

See the CMSIS-DAP documentation for general information.

4.1.2.2 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.

4.2 Connect your hardware

This section describes how to connect your hardware for the first time.

Procedure

1. Click **Device Manager** in the Activity Bar.
2. Connect your hardware to your computer over a USB connection.
   The Device Manager detects the hardware and a pop-up message displays in the bottom right-hand corner.
3. Click **OK** to use the hardware.

   Alternatively, click **Add Device** 

and select your hardware in the drop-down list that displays at the top of the window.

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

Next steps

To add more hardware, click **Add Device** in the top right-hand corner.

4.3 Edit your hardware

If the Device Manager cannot detect your hardware 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 press **Enter**. The hardware name is the name that displays in the Device Manager.
3. Select a Device Family Pack (DFP) CMSIS-Pack for your hardware in the drop-down list.
4. Select a device name to use from the CMSIS-Pack in the field and press **Enter**.

4.4 Open a serial monitor

Open a serial monitor. The serial output shows the output of your board. You can use the serial output as a debugging tool or to communicate directly with your board.

Procedure

1. Move your cursor over the hardware for which you want to open a serial monitor and click **Open Serial**  .
   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 that you select must be the same as the baud rate of your active project.
2. Select a baud rate.
   A **Terminal** tab opens with the baud rate selected.

5. Work with CMSIS solutions

Keil® Studio Cloud provides support for working with CMSIS solutions, otherwise known as csolution projects. Keil Studio Cloud manages the information needed to create your csolution projects.

You can carry out the following tasks:

- Select a solution from the workspace
- Set a context for your solution
- Use the Solution outline
- Manage software components
- Use the Configuration Wizard to customize startup code and other configuration files

You can also:

- Create a solution from scratch
- Configure a solution
- Convert a Keil µVision project to a solution

5.1 CMSIS solutions

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

Solutions are defined in YAML format using `*.csolution.yml` files. The `*.csolution.yml` 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.yml` files. The `*.cproject.yml` file defines the content of an independent build. Each project corresponds to one binary file (build artifact).

The `*.csolution.yml` file optionally includes information on the version of CMSIS-Toolbox to use to build your solution. If a CMSIS-Toolbox version is not specified in the `*.csolution.yml` file, Keil® Studio Cloud uses the version specified in the vcpkg manifest file if available. If no information is available on the version of CMSIS-Toolbox to use, Keil® Studio Cloud uses the latest version of CMSIS to build your solution.

You can edit the `*.csolution.yml` and `*.cproject.yml` files of a solution manually.

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.

5.2 CMSIS-Packs

CMSIS-Packs are 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:

- Source code, header files, and software libraries. Examples include RTOS (Real-Time Operating System), DSP (Digital Signal Processing), and generic middleware.
- Device parameters, for example the memory layout or debug settings, along with startup code and Flash programming algorithms
- Board support, for example 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

Various silicon and software vendors develop CMSIS-Packs, 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.

5.3 Select a solution from the workspace

If you have several solutions in your workspace, the **Select solution from workspace** option enables you to set the active solution and switch between solutions.

You can set the active solution either from the **Explorer** view or from the **Solution outline** view.

- To set the active solution from the **Explorer** view, right-click the folder containing your csolution project files, or the `csolution.yml` file itself, and select **Select solution from workspace**. The active solution is indicated by an arrow.
- To open a project from the **Solution outline** view:
  - Click **CMSIS**  in the Activity Bar to open the **CMSIS** view. The **Solution outline** displays on the left.
  - In the header next to the solution name, click **Views and More Actions**  , and then select **Select solution from workspace**.
  - Choose a solution from the list that opens at the top of the window. The **CMSIS** view loads and activates your chosen solution.

5.4 Set a context for your solution

Look at your solution context set. A context set defines how to build an application image from multiple related projects.

Procedure

1. Click **CMSIS** in the Activity Bar to open the **CMSIS** view.
2. Choose one of the following options:
   - Click in the **Solution outline** header
   - Select **CMSIS: Manage Solution Settings** from the Command Palette
   - Click in the status bar.

   The **Manage Solution** view opens.
3. Look at the available contexts for the solution. You can change the target type, the projects included in the build, and the build type.
   You can also change the run and debug configurations. Keil® Studio Cloud uses the **Embedded Debugger** for the debug configuration.
   - **Active Target**: Select a **Target Type** to specify the hardware to use to build the solution. Some examples are also compatible with Arm® Virtual Hardware (AVH) targets, in which case more options are available. For more details, read the AVH solutions overview.

     Click **Edit targets in csolution.yml** to specify your target types by editing the YAML file directly.
   - **Active Projects**:
     - **Project Name**: The project or projects included in the build. If you have multiple projects in your solution, you can select the projects to include here.
     - **Build Type**: The build configuration. A build configuration allows you to configure each target type towards specific testing. You can set different build types for different projects in your solution. You can create your own build types as required by your application. Two commonly used examples are **Debug** for a full debug build of the software for interactive debugging, or **Release** for the final code deployment.

     Click **Edit cproject.yml** next to a project to open the `<project-name>.cproject.yml` file. YAML syntax support helps you with editing.

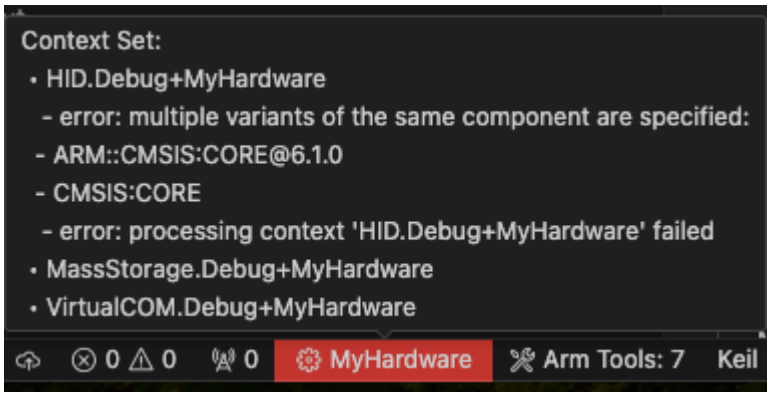
Note

The projects and build types you can select are defined by contexts for a particular target. Some options might be unavailable if they have been excluded for the target selected. To learn more about contexts and how to modify them, see the Context and Conditional build information in the CMSIS-Toolbox documentation. For example, you can use `for-context` and `not-for-context` to include or exclude target types at the `project:` level in the `*.csolution.yml` file.

- **Run and Debug**:
  - **Run Configuration**: Choose a run configuration to use for your solution. Choose **Run on Virtual Hardware** to run your solution on a target that is compatible with Arm® Virtual Hardware (AVH) (see the AVH solutions overview for more details). Choose **Flash Device** to run your solution on a physical board.
  - **Debug Configuration**: Keil® Studio Cloud uses the **Embedded Debugger** for the debug configuration.

4. You can inspect errors and warnings for a context set. Errors and warnings display for the active projects in the context set when you move your cursor over the **Context Set** in the status bar. The **Context Set** indicator is red for errors and yellow for warnings.

   Figure 5-1: Context Set errors and warnings

   Click the **Context Set** indicator to open the **Output** tab for the **CMSIS Solution** category. If you previously closed the **Manage Solution** view, then this action also re-opens the view.
5. You can also go to the **Problems** tab and check for errors.
6. Open the `main.c` file and check the IntelliSense features available. To find out about the different features, read the Visual Studio Code documentation on IntelliSense.

5.5 Use the Solution outline

The **Solution outline** presents the content of your solution in a tree view.

Click **CMSIS** in the Activity Bar to open the **CMSIS** view. The **Solution outline** displays on the left.

The **Solution outline** shows the projects (cprojects) included in the solution that are selected in the **Manage Solution** view. Each cproject file contains configuration settings, source code files, build settings, and other project-specific information. Keil® Studio Cloud uses these settings and files to manage and build a software project for a board or device.

You can have the following details for a project:

- User files: User files are the files that you add to the project and that you can edit. For example, a README or code files. You can organize user files using groups, and add files, sub-groups, or component code templates to groups. Move your cursor over a group and click  , then select one of these options:
  - **Add New File**: Create a file and add it to the group
  - **Add Existing File**: Select an existing file and add it to the group
  - **Add New Group**: Add a sub-group to the group selected
  - **Add From Component Code Template**: Select a component code template in the drop-down list and add to the group. A code template is a predefined file included with the software components for your project to help you start developing your project.

    The files you add are listed in the `*.cproject.yml` file of the solution under the `groups` key. You can also manually add files under the `groups` key. They display as groups without names in the **Solution outline**. For example:

    ```
    groups:
        - files:
            - file: README.md
    ```

- **constructed-files**: Contains files such as the `RTE_Components.h`, `Pre_Include_Global.h`, and `Pre_Include_Local_Component_h` header files that are generated for each context. See RTE_Components.h, Pre_Include_Global_h, and Pre_Include_Local_Component_h for more details.
- **linker**: Contains a linker script file and a <regions>.h file (or other user-defined header files). See Linker Script Management for more details.
- **Components**: All the software components selected for the project. Components are sorted by component class (Cclass). Code files, user code templates, and APIs from selected components display under their parent components. Click the files, templates, or APIs to open them in the editor. Click the book icon  of a component to open the related documentation.
- **Layer Type:<type>** or **Layer:<base-filename>**: The clayer file, `*.clayer.yml`, defines the software layers for the project. A software layer is a set of source files, preconfigured software components, and configuration files. Multiple projects can use the clayer file. The software components used by each layer in the project appear in the tree view.

  If you are using a generator to configure your device or board, then a **Run Generator** option is available from the generator component entry under **Layer** > **Components** to start a generator session. For more details, see Generator Support.

The **Solution outline** label displays the name of your active solution. You can choose one of the following actions next to the solution name:

- **Build solution**: Click  to build the projects included in the context set that you defined in the **Manage Solution** view.
- **Run**: Click  to run the solution on your hardware.
- **Debug**: Click  to debug the solution.

- **Open csolution.yml file**: Open the main `csolution.yml` file. When you move your cursor over a project or a layer, an **Open file** option is also available.
- **Manage Solution Settings**: Click  to set a context for your solution.
- **Views and More Actions**  :
  - **Rebuild solution**: Clean the output directories before building the projects
  - **Refresh (reload packs, update RTE)**: Reload information from all installed packs and run `cbuild setup` including the `--update-rte` option.
  - **Convert uVision project**: Convert an existing µVision project to a solution
  - **Create new solution**: Create a solution from scratch
  - **Select solution from workspace**: Select the active solution. If you have several solutions in your workspace, this option allows you to switch switch between solutions. The same option is available from the **Explorer** when you right-click the folder that contains your csolution project files. You can also access this option when you right-click the `csolution.yml` file itself. The active solution is shown as highlighted.

The **Solution outline** displays the selected build type and target type next to each project. You can check which software components are selected for each project. Click  to open the **Software Components** view. See Manage software components for more details.

The Open-CMSIS-Pack documentation gives more information on the `*.csolution.yml`, `*.cproject.yml`, and `*.clayer.yml` file formats.

5.6 Manage software components

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

5.6.1 Open the Software Components view

This section describes how to open the **Software Components** view.

Procedure

1. Click **CMSIS** in the Activity Bar to open the **CMSIS** view.
2. Move your cursor over the **Solution outline**. Click **Manage software components** at the project level.

Results

The **Software Components** view opens.

The default view displays the components available from the packs listed in your solution (**Software packs: <Solution-name>** drop-down list and **All** toggle button).

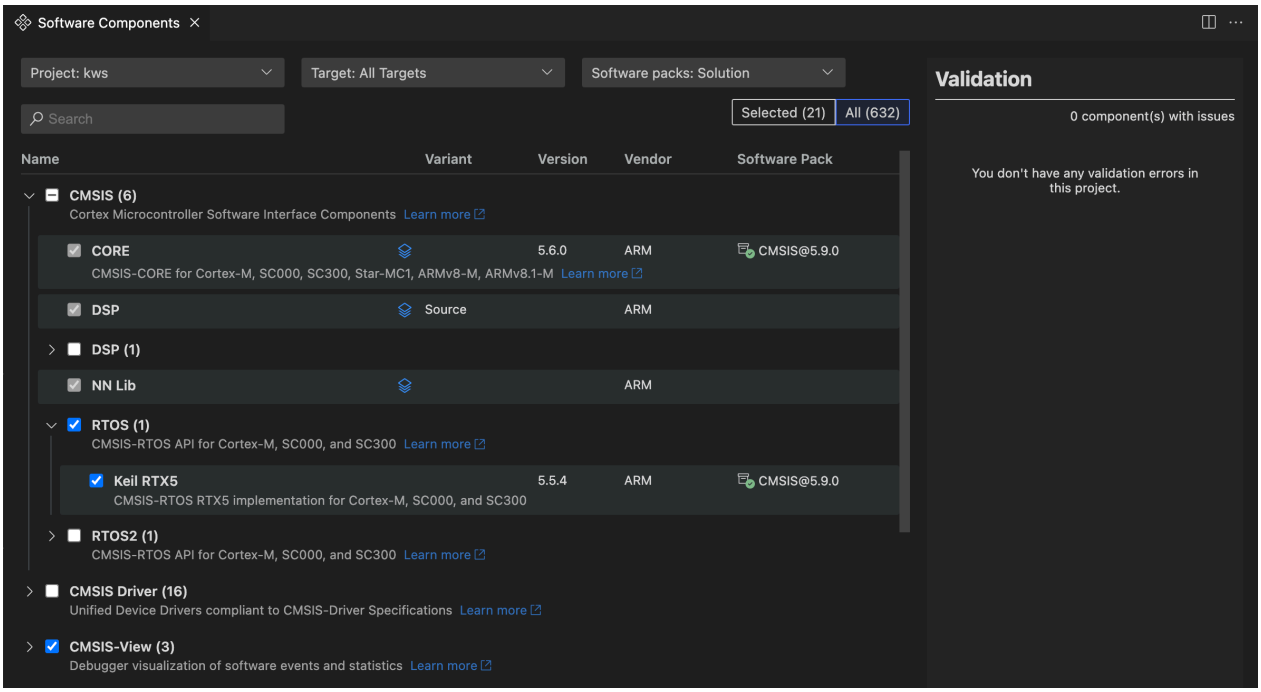
You can use the **Search** field to search the list of components.

With the **Project** drop-down list, select the project for which you want to modify software components.

With the **Target** drop-down list, select **All Targets** or a specific target type to modify software components for all the target types in your solution at once, or for a specific target only.

With the **Software packs** drop-down list, you can filter on the components available from the packs listed in your solution, or display the components from all installed packs.

Figure 5-2: The 'Software Components' view showing all the components available from the packs listed in a solution



The CMSIS-Pack specification states that each software component must 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 subgroup (Csub): A component subgroup 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 that is typically used when the same implementation has multiple top-level configurations, like **Library** 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 associated files.

Documentation links are available for some components at the class, group, or subgroup level. Click the **Learn more** link of a component to open the related documentation.

5.6.2 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 project.

Procedure

1. In the **Project** drop-down list, select the project for which you want to modify software components.
2. In the **Target** drop-down list, select a specific target type. If you want to modify all the target types at once, select **All Targets**. Note that some examples have only one target.
3. In the **Software packs** drop-down list, you can filter on the components available from the packs listed in your solution with the **Solution: <Solution-name>** option. You can display the components from all installed packs with the **All installed packs** option.
4. Check that the **All** toggle button is selected to display all the components available. Switch to **Selected** to display only the components that are already selected.
5. Use the checkboxes to select or clear components as required. For some components, you can also select a vendor, variant, or version.
   The `cproject.yml` file is automatically updated.
6. Manage the dependencies between components and solve validation issues from the **Validation** panel.

   Issues are highlighted in red and have an exclamation mark icon  next to them. You can remove conflicting components from your selection or add missing component dependencies from a suggested list.
7. If there are validation issues, move your cursor over the issues in the **Validation** panel to get more details. 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**.
   There can also be issues such as:
   - A component that you selected is incompatible with the selected hardware and toolchain.
   - A component that you selected has dependencies which are incompatible with the selected hardware and toolchain.
   - A component that you selected has unresolvable dependencies. In such cases, you must remove the component. Click **Apply** from the **Validation** panel.

5.6.3 Undo changes

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

5.7 Use the Configuration Wizard

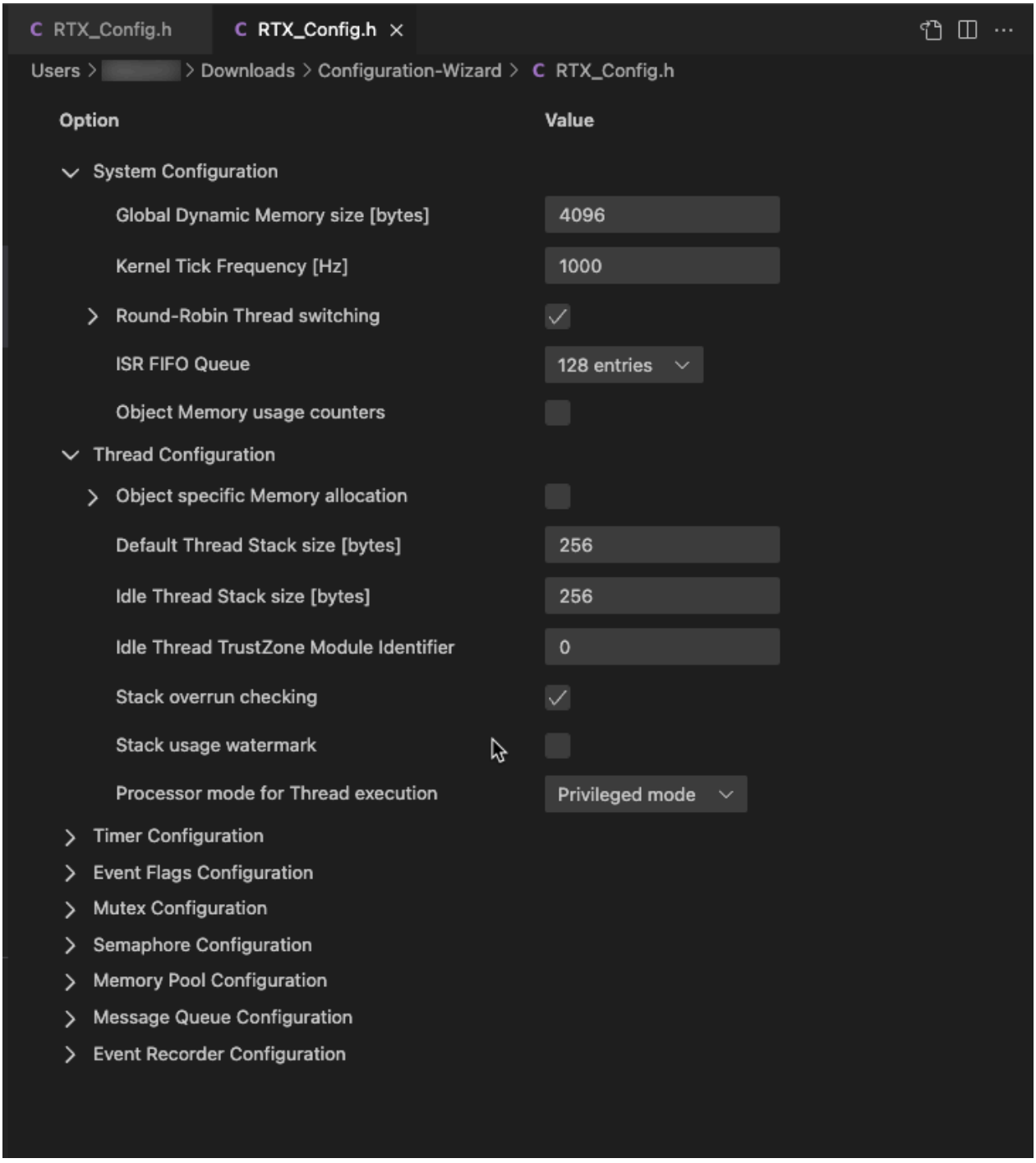
The Configuration Wizard simplifies the customization of startup code and configuration files by providing an intuitive dialog-style interface. The wizard allows you to quickly modify configuration settings without the need for extensive manual editing.

The Configuration Wizard interface is generated from annotations that are included in the configuration files themselves.

These annotations, which are like embedded markup tags, can be already available in the configuration files of software components used in your project, or you can add them yourself. For the set of rules for creating these annotations, see Configuration Wizard Annotations in the Open-CMSIS-Pack documentation.

To open the Configuration Wizard, open a configuration file containing annotations, then click **Open Preview**  .

Figure 5-3: Configuration Wizard



Changes you make in the Configuration Wizard are immediately reflected in the source file. You can also edit the source file directly.

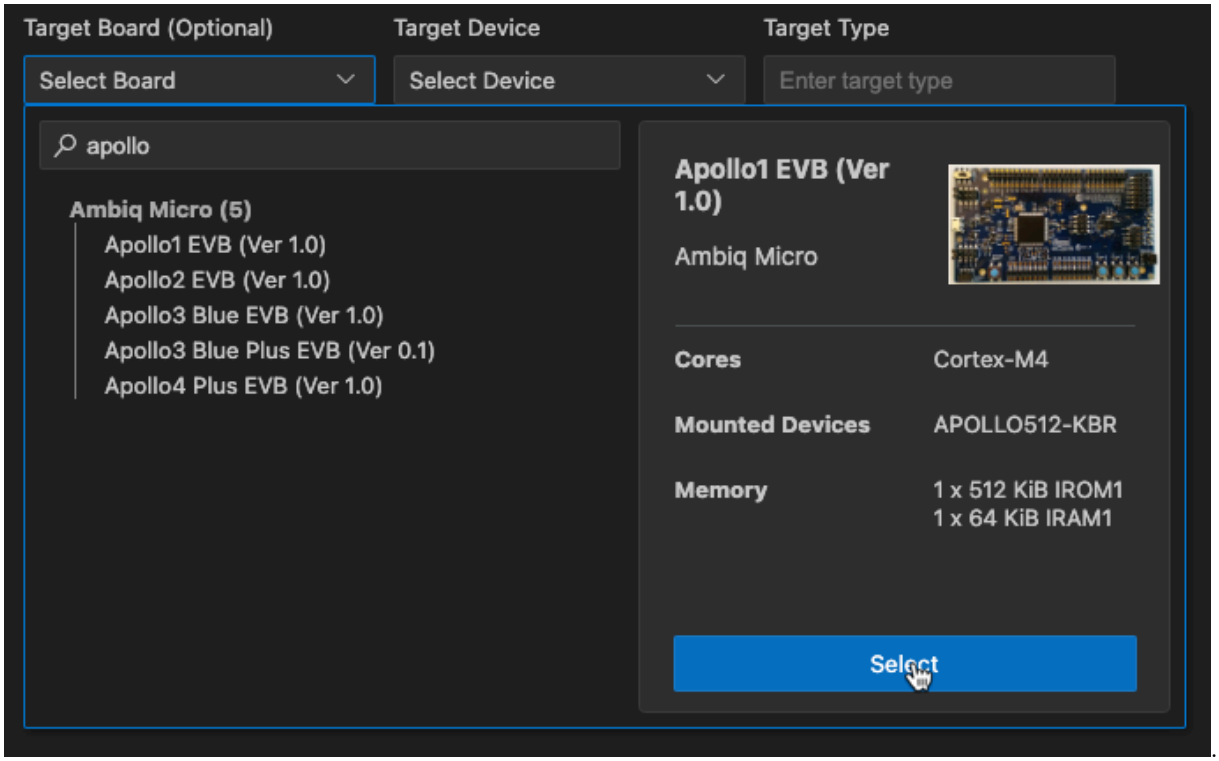
5.8 Create a solution

Create a solution project from scratch.

Procedure

1. To create a solution, either:
   - Click **CMSIS** in the Activity Bar to open the **CMSIS** view. Then:
     - If you are starting from an empty workspace, click **Create a New Solution**.
     - If you already have a solution open in your workspace and want to create a new one, move your cursor over the **Solution outline**. Next, click **Views and More Actions** > **Create new solution**.
   - Go to the **File** menu and select **New File...**, then select **Create new solution** in the drop-down list that opens at the top of the window.

   The **Create Solution** view opens.
2. Click the **Target Board** drop-down list. Enter a search term, and then select a board.
   A picker shows you the details of the board that you selected.



3. Click **Select**.
   By default, the **Target Device** drop-down list and **Target Type** show the name of the device mounted on the board that you selected.

Alternatively, you can directly select a device in the **Target Device** drop-down list, without selecting a board first.

4. In the **Target Type** field, you can customize the name of the target hardware that is used to deploy the solution. The **Target Type** displays in the **Manage Solution** view. The `target-types: - type:` section of the `<solution_name>.csolution.yml` file is updated automatically when you set a target type in the user interface.
5. Select one of the following options from the **Templates, Reference Applications, and Examples** drop-down list. Note that the option or options available depend on the board or device that you selected. If there are many examples available, enter a search term, and then select an example.
   - **Templates**: Templates are projects that you can use to get started. Templates do not include application-specific code.
     - Create a blank solution
     - Create a TrustZone solution. TrustZone is a hardware-based security feature that provides a secure execution environment on Arm-based processors. TrustZone allows the isolation of secure and non-secure zones, enabling the secure processing of sensitive data and applications. If the board or device that you selected is compatible, you can decide whether to use TrustZone, and define whether projects in the solution use secure or non-secure zones.
   - **Reference examples**: Use a reference application. Examples are only available if you selected a board in the **Target Board** drop-down list. Reference applications are not dependent on specific hardware. You can deploy them to various evaluation boards using additional software layers that provide driver APIs for specific target hardware. Layers are provided using Board Support Packs (BSPs).

     Reference applications are available with the MDK-Middleware software pack. These examples show you how to use software components for IPv4 and IPv6 networking, USB Host and Device communication, and file system for data storage. The examples use board layers. See MDK Middleware Reference Applications and the MDK-Middleware repository and documentation for more details.

     Other reference applications that illustrate how to match sensor shields and boards are also available with the Sensor SDK pack. The examples use board and shield layers. See Sensor Reference Applications and the Sensor-SDK-Example repository for more details.
   - **CMSIS solution examples**: Use a CMSIS solution example. CMSIS solution examples are created for a specific hardware or evaluation board. The examples are complete projects that directly interface with board and device peripherals.
   - **µVision examples**: Use a µVision example in `*.uvprojx` format as a starting point. µVision examples are converted automatically.
6. For blank and TrustZone solutions only:
   - If you selected `Blank solution`: Keil® Studio Cloud adds one project for each processor in the target hardware. You can change the project names. You can decide to add `secure` or `non-secure` zones with the **TrustZone** drop-down list if the board or device is compatible. By default, TrustZone is `off`.

- If you selected `TrustZone solution`: Keil® Studio Cloud adds a secure project and a non-secure project for each processor in the target hardware that supports TrustZone. You can change the project names. You can also change the `secure` or `non-secure` zones in the **TrustZone** drop-down list, or remove TrustZone by selecting `off`.

a. Click **Add Project** to add projects to your solution and configure them. For TrustZone, you can add as many `secure` or `non-secure` projects as you need for a particular processor.

b. Select a compiler: **Arm Compiler 6**, **GCC**, or **LLVM**.

7. You can change the name for your solution in the **Solution Name** field.
8. With the **Initialize Git repository** checkbox, you can initialize the solution as a Git repository. Clear the checkbox if you do not want to turn your solution into a Git repository.
9. Click **Create**.
   Keil® Studio Cloud creates and opens the solution. Keil® Studio Cloud automatically converts examples that are available only in `*.uvprojx` format. If there are conversion errors, check the `uv2csolution.log` file.
10. Check that the files for the solution have been created:
    - A `<solution_name>.csolution.yml` file
    - One or more `<project_name>.cproject.yml` files, each available in a separate folder. For reference applications only, each `cproject.yml` file contains a `$Board-Layer$` variable. For reference applications with sensor shields, each `cproject.yml` file contains a `$Shield-Layer$` variable too (`layers: - layer:`). These variables are not yet defined.
    - A main `<filename>.c` template file for each project

Next steps

Explore the autocomplete feature available to edit the `csolution.yml` and `cproject.yml` files. Read the CMSIS-Toolbox > Build Overview documentation for project examples.

See Configure a solution to add board and shield layers to your reference application. You can also select a compiler for reference applications and other solution types.

Add CMSIS components with the **Software Components** view. When you add components, the `cproject.yml` files are updated.

5.9 Configure a solution

If you have not already set a compiler, select a compiler for your solution from the **Configure Solution** view. If you created a reference application from a reference example, you can also add layers to your solution from the same view.

Procedure

1. Click **CMSIS** in the Activity Bar to open the **CMSIS** view.

2. Make sure that your solution is the active solution in the **Solution outline**, otherwise use the **Select solution from workspace** option in **Views and More Actions** ....
3. Run the **CMSIS: Configure Solution** command from the Command Palette.
4. If you are working with a reference application, the **Add Software Layer** area displays, showing the software layers that you can use. Layers are available from the Board Support Packs (BSPs) installed on your machine.

   Note
   - Not all BSPs have board layers.
   - Not all layers are compatible with the connections that your reference application requires.
   - The CMSIS-Packs which contain reference applications and layers generally provide an `Overview.md` file where the connections are detailed.

   If there are no compatible layers, errors display.

   a. Click **Next** to display the different options available.

   b. You can indicate where the layers should be copied to in the **Board-Layer** and **Shield-Layer** fields. Click **Default** to reset the paths to their default values.

   c. If no compiler is set for the reference application, the **Select Compiler** area displays under the layers selection and shows the compilers available in your environment. Select a compiler. For example, **AC6**.
5. If you are working with another solution type, only the **Select Compiler** area displays. Select a compiler.
6. Click **OK**.
   For reference applications only, a `Board.clayer.yml` file and a `Shield.clayer.yml` file are added in the folders that you selected. The files are available from the **Explorer**. Layers are automatically added in the `csolution.yml` file of your solution under `target-types: variables:`.

   For all solution types, the compiler is added with the `compiler:` key.

5.10 Convert a Keil µVision project to a solution

You can convert any Keil® µVision® project to a solution from Keil® Studio Cloud. Note that the conversion does not work with Arm® Compiler 5 (AC5) projects. You can download Arm Compiler 5 projects from the website, but you cannot use them with 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 solutions in Visual Studio Code. For more information, see the Migrate Arm Compiler 5 to Arm Compiler 6 application note and the Arm Compiler for Embedded Migration and Compatibility Guide.

Procedure

1. Drag and drop the folder that contains the `*.uvprojx` that you want to convert onto the Visual Studio Code workspace to open it. Alternatively, import a µVision project from keil.arm.com, or clone a project from GitHub.
2. Right-click the `*.uvprojx` and select **Convert uVision project** from the **Explorer**.
   The conversion starts immediately.

   Alternatively, if you are starting from an empty workspace, click **CMSIS** in the Activity Bar to open the **CMSIS** view. Then, choose one of the following two options:

   - Click **Convert a µVision Project** and open your `*.uvprojx` file to convert it
   - Move your cursor over the **Solution outline**, click **Views and More Actions** ..., then select **Convert uVision project** and open your `*.uvprojx` file to convert it

     You can also run the **CMSIS: Convert uVision project** command from the Command Palette. In that case, select the `*.uvprojx` that you want to convert on your machine and click **OK**.

     If there are conversion errors, check the `uv2csolution.log` file.
3. Check the **Output** tab. In the **View** menu, select **Output** to open the **Output** tab. Select **µVision to Csolution Conversion** in the drop-down list on the right side of the **Output** tab. The `*.cproject.yml` and `*.csolution.yml` files are available in the folder where the `*.uvprojx` is stored.

6. Debug your projects

Keil® Studio Cloud integrates the Arm Embedded Debugger.

6.1 Start an Embedded Debugger session

Start a debug session.

Procedure

1. Check that your device is connected to your computer and has been detected by the Device Manager. See Connect your hardware for more details.
2. To start a debug session, go to the **Run and Debug** view  , then click **Start Debugging**. The **Run and Debug** view displays and the debug session starts. The debugger stops at the `main()` function of your project.
3. To see the debugging output, check the **Debug Console** tab.

Next steps

Look at the Visual Studio Code documentation to learn more about the debugging features available in Visual Studio Code.

7. Submit feedback

If you have suggestions or if you discover an issue with the Keil® Studio Cloud, get in touch with us. Go to https://www.keil.arm.com/support and use the links 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
0200-01	28 November 2024	Non-Confidential	First release

Change history

For information about the functional changes to the Arm® Keil® Studio Cloud CMSIS environment, see the Arm Keil Studio Cloud CMSIS environment 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.


Tip

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


Remember

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

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 Cloud User Guide (Classic)	102497	Non-Confidential
Arm Keil Studio Visual Studio Code Extensions User Guide	108029	Non-Confidential
µVision User Guide	101407	Non-Confidential