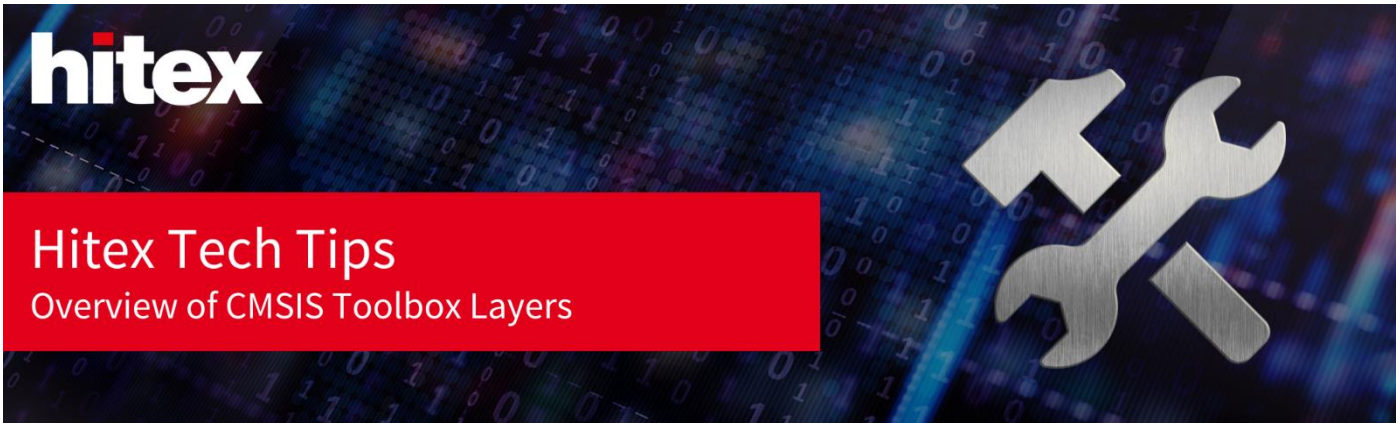

Overview of CMSIS-Toolbox Layers

By Trevor Martin

A Layered Approach

The use of CMSIS layers provides a sophisticated method of managing a project across several different hardware and simulation platforms. A strong example of this layered approach in action is the Keil Middleware components. These components—ranging from file systems and networking stacks to USB and graphical interfaces—are designed to integrate seamlessly within the CMSIS framework. With component layers, developers can easily swap out or reconfigure middleware modules depending on the underlying hardware or functional requirements, all while maintaining a consistent and maintainable codebase.

Within MDK-6 you can create a middleware project by selecting a project template in the initial setup screens.

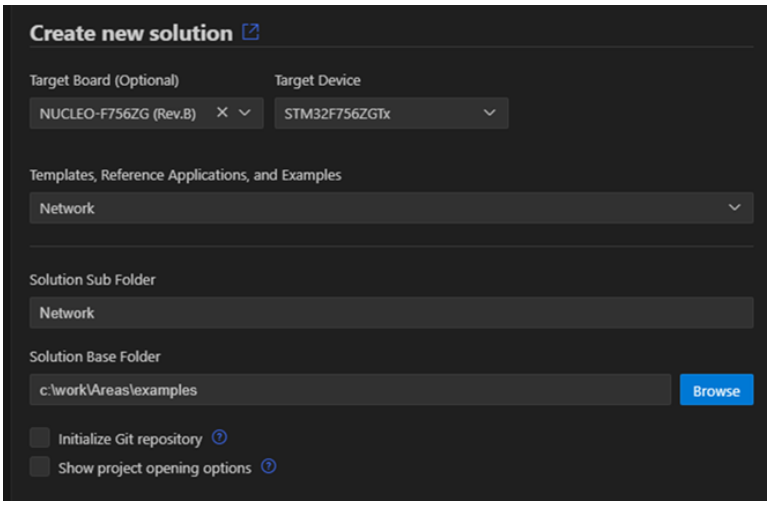

Fig 1 Creating a new project with board and template selection

Here we have selected a network project to run on a specific evaluation board. The board support pack for this board comes with a matching board support layer which can be selected for the project.

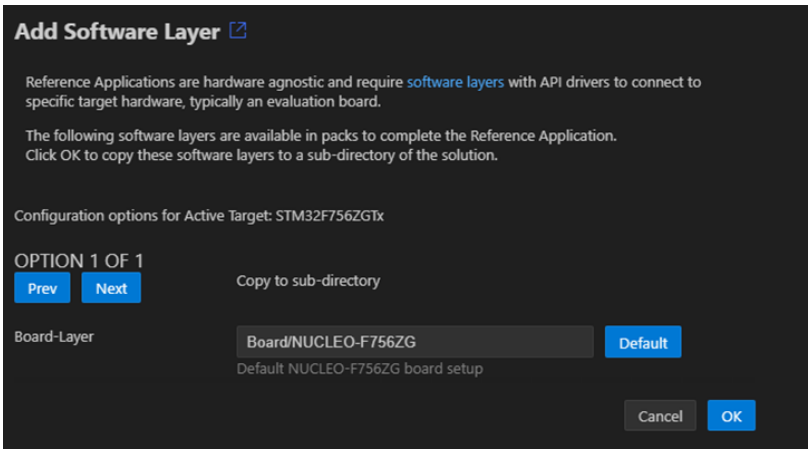

Fig 2 A board support package will provide a preconfigured board layer

This adds all the necessary low-level configuration files for specific board along with the necessary GPIO pin multiplexing and available CMSIS-Driver support.

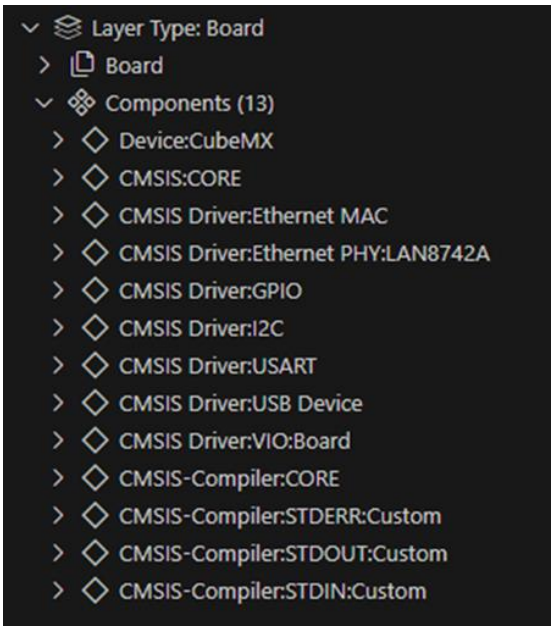

Fig 3 The board layer adds low level drivers and peripheral setup through a Silicon vendor ‘generator’ project ie Cube MX

The networking component now provides a set of initial demo projects which are added as upper software layer, this completes the initial project setup.

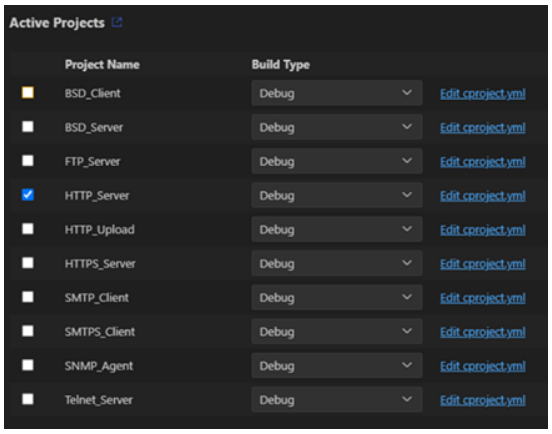

Fig 4 A range of template projects are added for selected software components

So, in a few clicks we can setup a project that will configure the Microcontroller peripherals and GPIO to match the known hardware and then add the platform components along with template code to build a working example that will serve as a project starting point.

Software component Layers

Another key advantage of CMSIS-Toolbox layers is their ability to be stacked, enabling developers to configure and integrate multiple components seamlessly. A prime example of this is the Arm Synchronous Data Streaming (SDS) framework—a versatile data logging component designed to capture sensor data for training machine learning algorithms. The SDS framework supports a variety of data output channels, including TCP/IP, USB, serial interfaces, and the local file system. The desired output method is selected at design time, allowing the framework to be tailored to specific project requirements.

Once the SDS software pack is installed, it becomes readily available as a template when creating new projects. This simplifies setup and ensures consistency across development environments.

Now when we create a project and select the SDS template the SDS layer will appear alongside the board layer.

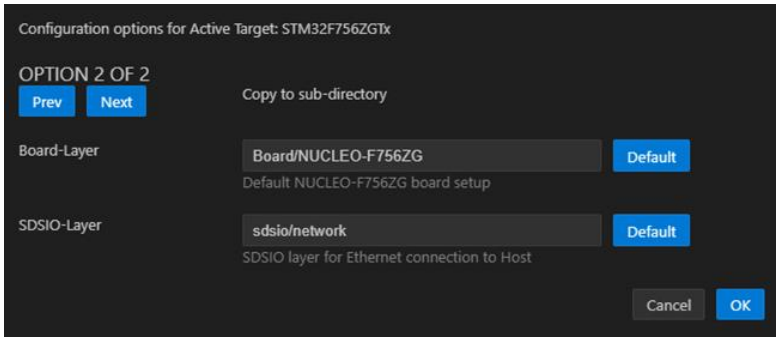

Fig 5 A software component may also provide a range of pre-configured layers

This allows you to select an SDS layer from the range of interfaces supported by the selected evaluation board. In this case we have configured the evaluation board as before and then added a second layer, which consists of the network component along with the SDS framework.

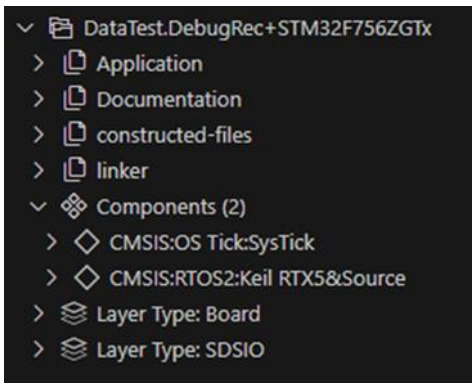

Fig 6 The component layer may be selected to support a desired interface or range of features

Once the project components have been configured the SDS framework provides a set of configuration projects that allow it to perform as a data recorder or player.

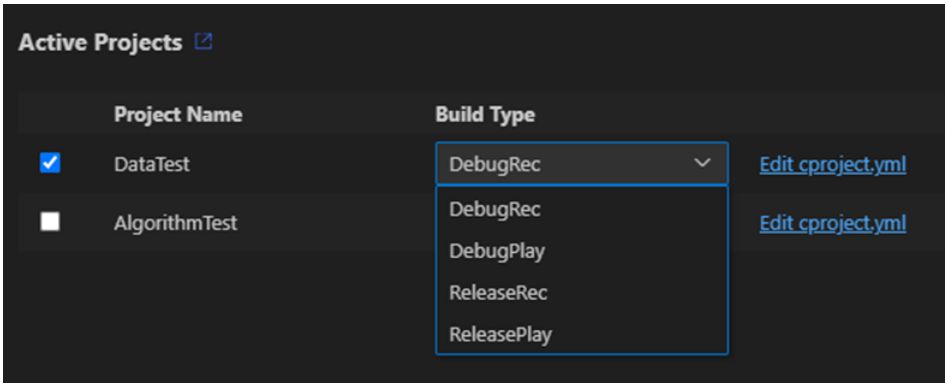

Fig 7 Finally an application template may be added to utilise the configured platform

The introduction of CMSIS-Toolbox layers significantly streamlines the configuration and management of complex software platforms. By abstracting hardware dependencies and organizing functionality into modular layers, developers can more easily integrate software components, whether they are middleware, device drivers, or application logic. This layered architecture aids the rapid adaptation of code across different hardware targets and simulation environments without requiring extensive rewrites.

In Summary

In modern code development, software layers are essential for maintaining scalability, portability, and maintainability. They enable clear separation of concerns, allowing teams to work in parallel on different parts of a system while minimizing integration friction. Additionally, layers promote code reuse and simplify testing by isolating functionality, making it easier to identify bugs and verify behaviour independently. As embedded systems grow in complexity and support for multiple platforms becomes a necessity, layered software design—exemplified by CMSIS-Toolbox—becomes a key enabler of efficient and sustainable development workflows.

Further Information

For more information visit our website: www.hitex.co.uk or get in touch: info@hitex.co.uk. You can also connect with us: LinkedIn

Sign up to receive Tech Tips directly to your inbox. Never miss a trick!