Background primer to CMSIS packs

By Colin Funnell

Introduction

CMSIS packs are at the heart of MDK and are a great way to include managed, standardised chunks of code and other resources into your project. The uses range from low-level silicon start-up code in Device Family Packs, through Board Support Packages to high-level protocols and services. There is no reason why you can't benefit from adopting similar structures for your own needs. In fact, Open-CMSIS-Pack encourages this to promote software reuse.

Whilst always a good intention, there comes with it a degree of complexity which this TechTip hopes to help explain.

Let's start off with the basics…

What is a CMSIS pack?

Terms like CMSIS get banded around quite a bit, so it is worthwhile to take a moment to understand the history behind it.

Common Microcontroller Software Interface Standard (CMSIS) itself is a group of technologies and conventions that was borne out of the need for a harmonised approach to developing for the wide range of Arm devices from different manufacturers. There are a number of different CMSIS areas concentrating on different aspects. Device Drivers, Debugging, Neural Nets, Real-Time Operating Systems, DSP… the list goes on.

CMSIS *Pack* concerns itself with bundling various files in a consistent manner, and in a way that other tools can understand and make use of. Delivery of the other CMSIS topics is done using CMSIS Packs. It has moved beyond solving problems for a particular toolchain and is generally useful for all sorts of systems. It has now become a free-standing project – Open-CMSIS Pack. Whilst it has its roots with Arm Cortex-M, it is now a much wider and extensible tool for everyone. With more independence and freedom from MDK release cadence, the descriptors used internally for CMSIS Pack are quickly growing.

Where has it been used already?

The Arm Keil tools - MDK (Microcontroller Development Kit) have been using CMSIS packs for many years as a means of adding to the core MDK system. Originally the MDK installation had to provide everything needed to develop with for the vast number of devices available. The installation size ballooned as newer devices required their own particular pieces of support code. Installing all of this when developers were only interested in a few devices was cumbersome. Having the ability to selectively introduce specific items and resolve dependencies with automatic download was a major step forward. Silicon vendors could manage released code for their devices.

The MDK tools had its own needs which were then translated into CMSIS Pack. The basic concepts of versioning, dependencies, source and documentation location were put in place. As time went on, more advanced concepts were then included into the pack management system. Users of Keil µVision will be familiar with these in the Run Time Environment manager and Pack Installer. The ability to switch between different versions or lock-off a project to a set of packs is incredibly useful. With the modern need to account for all components of your software in an SBOM, CMSIS Pack management offer an immediate shortlist of what is being used.

As CMSIS packs, and CMSIS in general have gained popularity, support for them has been appearing in other toolchains. Whilst some vendor specific toolchains will first and foremost support their own tools and methods, the support for the wider CMSIS approach is no bad thing. From a developer's point of view the common approach offered by CMSIS allows greater freedom to move to other Arm devices which have CMSIS support with them.

What's inside a CMSIS pack?

There's nothing magical about the contents of a CMSIS pack. The pack file itself is a .zip compression of various files and folders. The filename has to follow the rules of being <vendor>.<pack name>.<pack version>.pack. You can rename a .pack file to .zip to explore for yourself. Inside are the sources files from the originator along with a Pack Description file (.pdsc) which is where the useful information lies. The .pdsc file is in XML format, again available to be explored in an expanded pack, which has all the details describing the packages files, their uses and attributes.

As a CMSIS pack user, all of this is hidden away and managed by IDEs that support them. Even for build flows which don't natively understand these packs, CMSIS-Toolbox provides utilities to help manage, use and create them. MDK-6 goes one step further and integrates CMSIS-Toolbox as a means to handle the behind-the-scenes control of CMSIS Packs. By using CMSIS-Toolbox you are using the same tools as MDK uses itself.

CMSIS Toolbox also provides a standalone CLI which can recreate the same build as the IDE allowing you to easily migrate to a continuous integration server.

It is the Pack Description schema which is in continuous evolution, providing more and more useful methods of source control – all backed-up with CMSIS-Toolbox to support them.

Some Pack Description details

There are many different types of CMSIS Packs which can be confusing. Or rather, the file bundling mechanism for delivery can be used to support many different types of code. There are different parts of the .pdsc file which help describe how it appears and can be used by the developer through appropriate tools.

Versioning – Version numbering can be done almost excessively. The pack file has a version number in its filename as well as a release version field in the .pdsc file. Elements within the pack can also have their own versioning too. This way a pack of many things can be updated and the elements which have not changed remain with the same version as before. User configurable header files can have their own independent version to help control the transition of configuration as packs mature.

Dependencies – Quite a complicated set of conditions can be described for use by the various parts of a pack. The basics are mainly to check for another named CMSIS Pack or type and minimum version. Tests can also be done for the class of MCU being used as well as the selected compiler.

Source location – A powerful part is embedding the location of where more packs can be found. This could be publicly hosted, a master list by Keil or even just local filespace for use by your organisation.

Licensing agreements – Enforced by tools, a licence file can be shown and the user actively agrees to using the pack before being used.

Device Family Pack (DFP) – These are device startup files for a family of MCUs. Any new MCU device to be useful ought to have this to enable a developer to get straight into developing their code without worrying about initial stack pointers and other low-level CPU details.

Application Programming Interfaces (APIs) – Interface definitions have their own class within CMSIS pack descriptors. Along with version information this allows application code to be selective to which API to interact with.

Middleware – As MCUs are becoming more powerful and peripherals more complicated, large chunks of useful code to make use of them are needed. The Arm compiler toolchain provides a set of useful middleware such as network protocols and filesystem which plug-in nicely to underlying CMSIS Drivers.

Board Support Packages (BSP) – This section of pdsc contains files relating to using specific board hardware and the devices it has. Ideal for silicon vendors when they have a dev kit which can hold many different core devices or just many different boards to support.

Examples and Templates – Code, even with documentation, isn't the most fun way to try things out. A good set of examples and useful template files are always welcome. By explicitly defining them in the pack, aware IDEs can integrate with them and bring their usefulness to the user.

All of the above is a fraction of what can be described in the pdsc data. The best part is that the power of CMSIS packs is available to end users, not only to use them, but to create them. Software reuse can be done in a much more graceful manner than merely obtaining lots of C files.

If creating your own packs, it is recommended to visit the Open CMSIS pack repository to find the latest schema and details on how best to use them. A good feel of what can be done is available just by using MDK.

In summary

CMSIS Pack is an open-source approach to providing chunks of code for developers. It already has a good history of being used in the MDK toolchain and is being used by tools as well. By utilising more MCU project context useful details, it can become a powerful ally – more than just a blob of source code. The power of using pre-existing CMSIS Packs could be yours - to pass on your work for others to use just as easily.

Useful links

https://www.open-cmsis-pack.org/

https://open-cmsis-pack.github.io/Open-CMSIS-Pack-Spec/main/html/index.html

https://github.com/Open-CMSIS-Pack/cmsis-toolbox/blob/main/docs/README.md

TechTips:

Hitex Knowledge Base

From the Insiders' Guide to CMSIS Packs series:

Creating your own CMSIS pack

Practical Considerations when making your own CMSIS pack

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