Application Note: 252

Product Lifecycle Management with Software Packs

ARM® KEIL®
Microcontroller Tools

Abstract

This application note explains how Software Packs can help you throughout the Product Lifecycle Management (PLM) process. The main benefits of Software Packs are explained and brought into context with regard to PLM. Using advanced versioning features of the Pack concept help to develop applications faster and to reduce maintenance costs in the end.

Contents

Revision History

- November 2013: Initial Version

What is a Software Pack?

Software Packs provide support for microcontroller devices, contain software components such as drivers and middleware, and may include project examples and code templates. These are the major categories of Software Packs:

- Device Pack: generated by a silicon supplier or tool vendor; provides support to create software applications for a specific target microcontroller.
- Board Support Pack: published by a development board vendor to support the peripheral hardware mounted on the board
- CMSIS Pack: provided by ARM® and includes support for CMSIS Core, DSP, and RTOS.
- Middleware Pack: created by a silicon supplier, tool vendor or a third party; reduces development time by giving access to popular software components (such as software stacks, special hardware libraries. etc).
- In-house components: developed by the tool user for internal or external distribution.

A Software Pack is a ZIP archive containing all required libraries and files and a package description file (PDSC) with all the information about the Software Pack. The structure of a Software Pack is defined in CMSIS. Refer to CMSIS-Pack (www.keil.com/CMSIS/Pack) for more information.

What is a Software Component?

Software components may contain

- Source code, libraries, header/configuration files and documentation.
- Complete example projects that show the usage of the software component and which can be downloaded and executed on evaluation hardware.
- Code templates that can be used as a starting point for using software components.

What is Product Lifecycle Management (PLM)?

The process of managing a product's entire lifecycle from the concept stage up to end-of-life is called Product Lifecycle Management (PLM). It was first used by car manufacturers to get competitive advantages and to raise the overall quality of their product. Nowadays, there are many variants of PLM available, such as Application Lifecycle Management (ALM) that is more concerned about application software development. To be more generic, this white paper sticks to the term PLM.

There are four distinct phases of PLM:

- Concept: Product definition based on customer requirements. Major features are defined and a first functional prototype is created.
- Design: Prototype testing and implementation of the product based on the final technical features and requirements. Extensive tests are made to verify the product's features against the specification.
- Release: The product is manufactured and brought to market.
- Service: Maintenance of the products including support for customers; finally phase-out or end-of-life.

Benefits of Software Packs

In general, the new Software Pack concept gives three major benefits to the application developer:

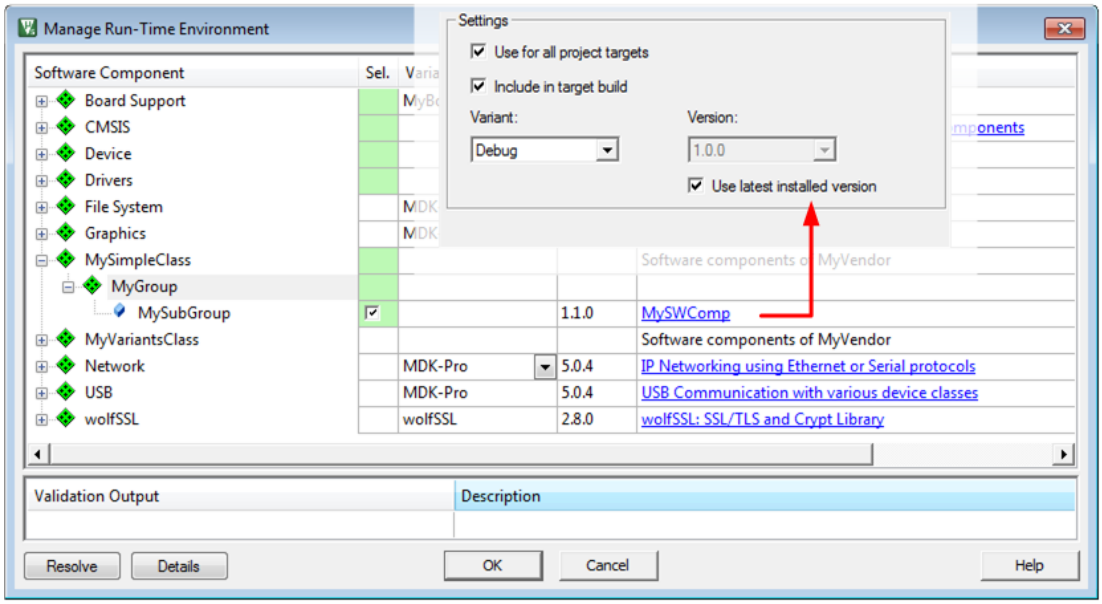
- Enhanced productivity
  - Software components are easily selected in the Manage Run-Time Environment window and can be exchanged quickly in early development phases.
  - All required files for a software component will be automatically added to the project. Libraries are referenced depending on the specific microcontroller core.
  - Dependencies to other software components (such as drivers, CMSIS or cores) are automatically detected and resolved (where applicable).
  - Integrated access to documentation enables quick access to software component specific help files.
  - Source code templates provide an application framework for frequent use cases.
  - Example projects enable faster development of software applications.
- Improved flexibility
  - It is no longer required to install a new version of MDK when support for a new microcontroller device is needed, as this is also covered by Software Packs
  - Being an open standard, third party software components are supplied in the same easy-to-use way, which improves overall flexibility.
- Simplified long-term project maintenance
  - Software Packs are easy to update and maintain. Pack Installer's update functionality takes care of revision control of Software Packs.
  - Centralized storage of Software Packs prevents the use of local copies of component source code or libraries.
  - Changing the target device at a late stage in the project life-cycle, selects the correct set of components automatically.
  - By default, projects use the latest versions of an installed component, but may be locked to a specific version.

How are these benefits reflected in the PLM process? The usage of Software Packs in the design flow is described in the following in more detail.

Concept — Use Various Software Packs to Evaluate Features

In the **Concept** phase, the features of the product are explored. Proven software components help to implement standard product features quickly. The **Pack Installer** manages and updates Software Packs and lists example projects that provide a starting point for a prototype. These examples can be modified quickly to explore the features of pre-built middleware.

The **Manage Run-Time Environment** window helps you to quickly add or remove software components. This helps you to identify the right set of software components for your application and saves a lot of work as you do not have to add or remove all the necessary files manually. To make use of software component feature updates, always use the latest version of the Software Packs:



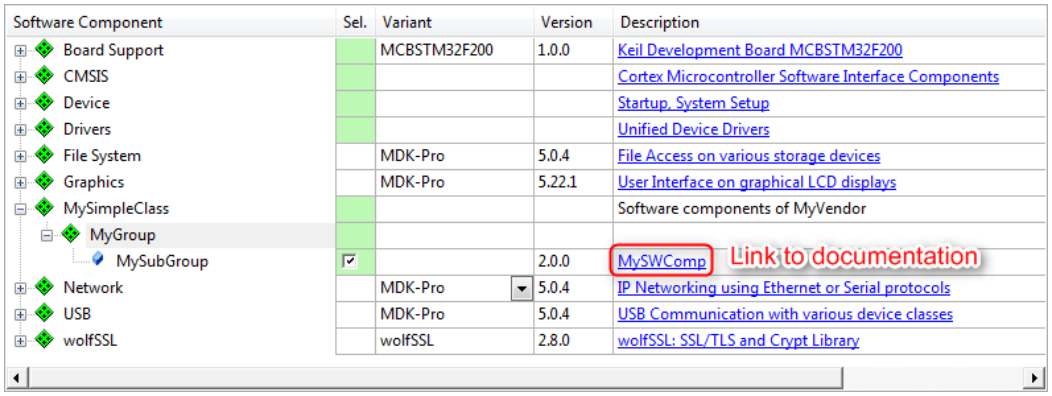
Note: This is the default setting in MDK for all software components.

Design — Use Documentation, Code Templates and the Latest Software Packs

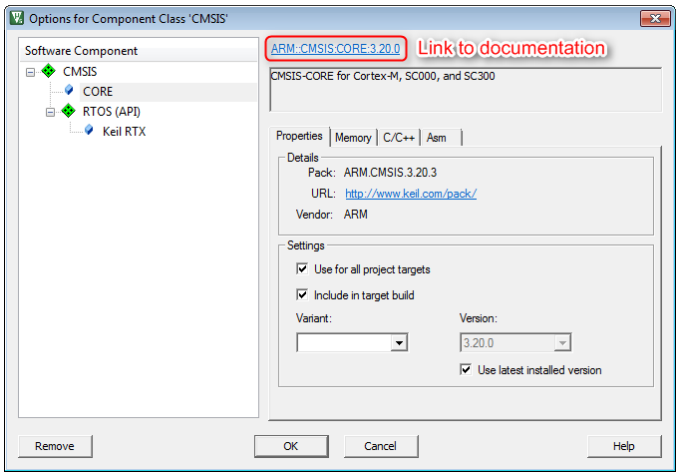
In the **Design** phase, the features of your application are implemented. Easy access to documentation and code templates help you to reduce development time.

There are three ways to access a software component's documentation:

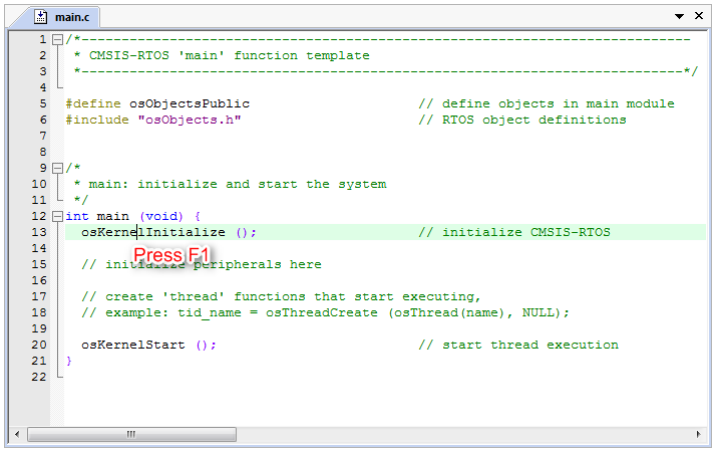
- Manage Run-Time Environment window



- Component Options in the Project window

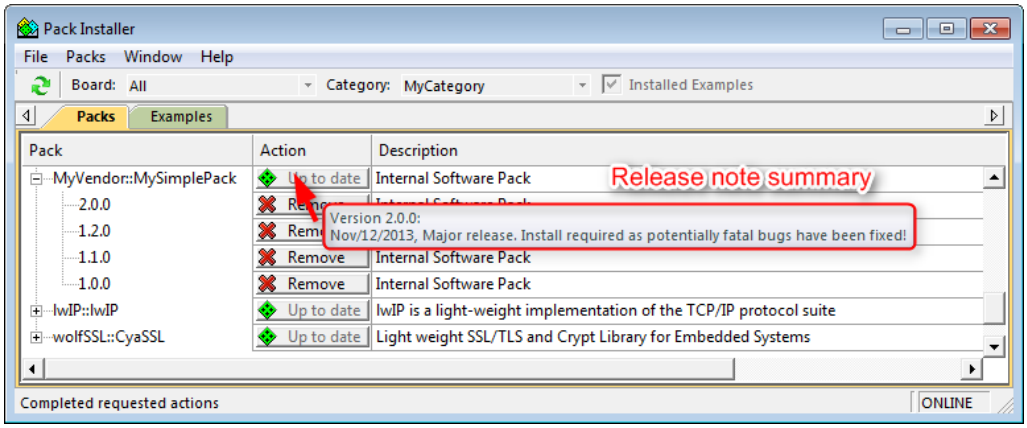


- Using the F1 key within the source code



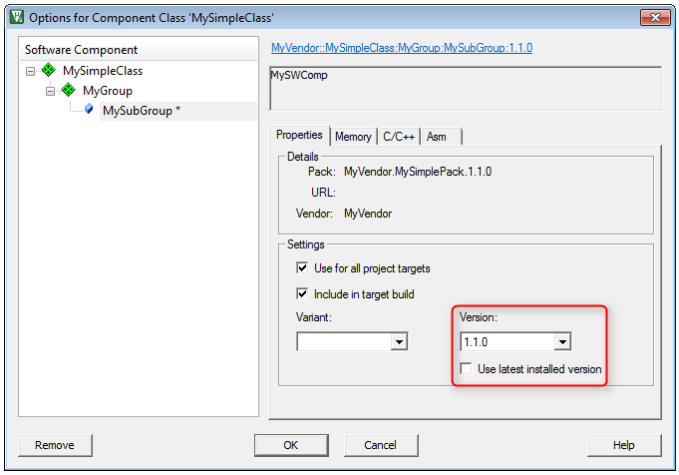
Code templates can be added using the in the **Add New Item to Group** dialog. Right-clicking on a source group in the Project window will show you this option. Code templates help you to get to speed with a software component, carrying pre-coded examples that can be easily adapted to your application.

In MDK Version 5, software components used in your application are updated by installing a new Software Pack. During the product development, typically new versions of software components become available and the Pack Installer allows you to keep track of updates. Easy access to release notes spot software issues that may affect your application.



Stay with Current Software Components

Just before the final **Release** of the product, freeze all software components to the current version. All testing has been done and the project is released to manufacturing. Thus, all components need to be maintained at that approved state for further reference, especially during the **Service** phase. Using the **Options for Component Class** dialog, the revision of a software component can be set to a specific version:



Simply uncheck "Use latest installed version" and choose the version that is to be used from the drop-down list above.

The *project*.build_log.htm file in the project directory contains a list of all software components sorted by Software Pack:

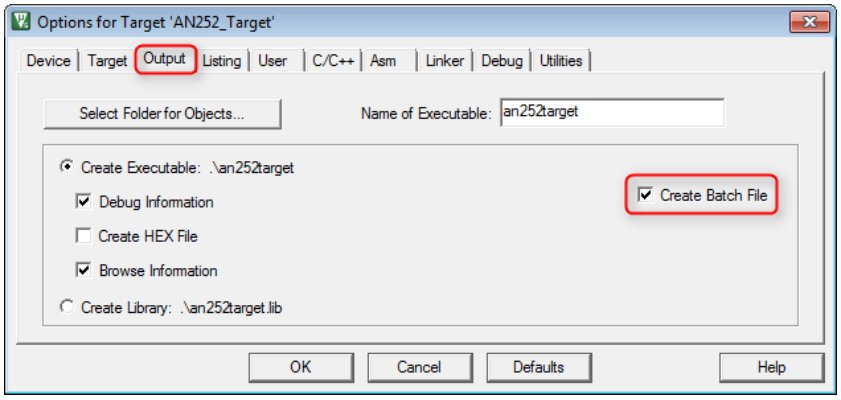
```
Software Packages used:
Package Vendor: ARM
                http://www.keil.com/pack/ARM.CMSIS.3.20.3.pack
                ::CMSIS:RTOS:1.0 (API)
                Cortex Microcontroller Software Interface Standard (CMSIS) CORE, DSP, RTOS,
Driver
   * Component: RTOS Version: 1.0
   * Component: CORE Version: 3.20.0
   * Component: Keil RTX Version: 4.73.0

Package Vendor: Keil
                http://www.keil.com/pack/Keil.STM32F2xx_DFP.1.0.4.pack
                Keil::Device:Startup:1.0.0
                STMicroelectronics STM32F2 Series Device Support, Drivers and Examples
   * Component: Startup Version: 1.0.0
   * Component: GPIO Version: 1.0.0
   * Component: LED Version: 1.0.0

Package Vendor: MyVendor
                http://localhost/MyVendor.MySimplePack.2.0.0.pack
                MyVendor::MySimpleClass:MyGroup:MySubGroup:2.0.0
                Internal Software Pack
   * Component: MySubGroup Version: 2.0.0
```

The basic information from the package description file (PDSC) is displayed here: The package vendor, the name of the Pack, and the download link. This is very helpful to keep an eye on the components in the project and the Software Packs that need to be archived in the **Service** phase. You will find the PACK files in your MDK installation directory, for example `C:\Keil\ARM\Pack\.Download`. Pack Installer stores all PACK and PDSC files in this directory before installing them on the system.

Archive Your Project and the Related Software Packs

In the **Service** phase, the project is usually saved to some kind of repository that keeps the final design files in one place. The Pack concept makes it easy to save all required software components as well. Simply save the PACK files of all used software components in the repository for further use. Together with revision control systems for your code, this helps you to open the project for example one year after deployment as if you had left it just the day before.

If you simply wish to rebuild your project, you can use a "make" type of approach. First, you need to select "Create Batch File" in the Output tab of the **Options for Target** dialog box:



This will create a batch file and some additional invocation files with each build. Executing the batch file will invoke the compiler with the same settings as in your project. The invocation files carry the information about the required files for a successful build.

To archive the µVision project for further use, you should think about archiving the following files:

- *project*.uvprojx file (µVision project)
- All source Code (listed in the <Group> section of the project.uvprojx file)
- *project*.uvoptx file (µVision project options)
- All Software Packs (refer to the list in the *project*.build_log.htm file)
- Directory \RTE (to have the same configuration settings available as within your project)
- project.bat (batch file for command line project build)
- *.__i and *.lnp (input files for batch file)