



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

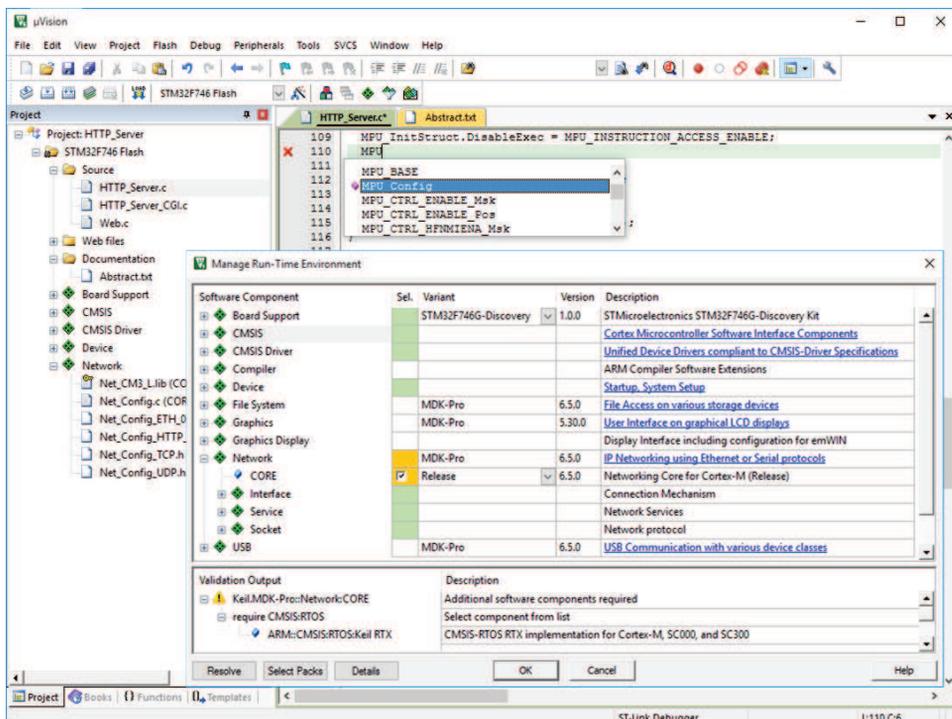
Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



Getting started with MDK

Create applications with μ Vision[®]
for ARM[®] Cortex[®]-M microcontrollers



Information in this document is subject to change without notice and does not represent a commitment on the part of the manufacturer. The software described in this document is furnished under license agreement or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. The purchaser may make one copy of the software for backup purposes. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than for the purchaser's personal use, without written permission.

Copyright © 1997-2017 ARM Germany GmbH
All rights reserved.

ARM[®], Keil[®], μ Vision[®], Cortex[®], TrustZone[®], CoreSight[™] and ULINK[™] are trademarks or registered trademarks of ARM Germany GmbH and ARM Ltd.

Microsoft[®] and Windows[™] are trademarks or registered trademarks of Microsoft Corporation.

PC[®] is a registered trademark of International Business Machines Corporation.

NOTE

We assume you are familiar with Microsoft Windows, the hardware, and the instruction set of the ARM[®] Cortex[®]-M processor.

Every effort was made to ensure accuracy in this manual and to give appropriate credit to persons, companies, and trademarks referenced herein.

Preface

Thank you for using the ARM Keil® MDK Microcontroller Development Kit. To provide you with the best software tools for developing ARM Cortex-M processor based embedded applications we design our tools to make software engineering easy and productive. ARM also offers complementary products such as the ULINK™ debug and trace adapters and a range of evaluation boards. MDK is expandable with various third party tools, starter kits, and debug adapters.

Chapter overview

The book starts with the installation of MDK and describes the software components along with complete workflow from starting a project up to debugging on hardware. It contains the following chapters:

MDK Introduction provides an overview about the MDK Tools, the software packs, and describes the product installation along with the use of example projects.

CMSIS is a software framework for embedded applications that run on Cortex-M based microcontrollers. It provides consistent software interfaces and hardware abstraction layers that simplify software reuse.

Software Components enable retargeting of I/O functions for various standard I/O channels and add board support for a wide range of evaluation boards.

Create Applications guides you towards creating and modifying projects using CMSIS and device-related software components. A hands-on tutorial shows the main configuration dialogs for setting tool options.

Debug Applications describes the process of debugging applications on real hardware and explains how to connect to development boards using a wide range of debug adapters.

Middleware gives further details on the middleware that is available for users of the MDK-Professional and MDK-Plus editions.

Using Middleware explains how to create applications that use the middleware available with MDK-Professional and MDK-Plus and contains essential tips and tricks to get you started quickly.

Contents

Preface	3
MDK Introduction	7
MDK Tools.....	7
Software Packs	8
MDK Editions.....	8
Installation	9
Software and hardware requirements	9
Install MDK-Core.....	9
Install Software Packs.....	10
MDK-Professional Trial License.....	11
Verify Installation using Example Projects	12
Use Software Packs	16
Access Documentation	20
Request Assistance	20
Learning Platform.....	21
Quick Start Guides.....	21
CMSIS	22
CMSIS-CORE	23
Using CMSIS-CORE.....	23
CMSIS-RTOS2.....	26
Software Concepts.....	26
Using Keil RTX5.....	27
Component Viewer for RTX RTOS	36
CMSIS-DSP.....	37
CMSIS-Driver	39
Configuration.....	40
Validation Suites for Drivers and RTOS	41
Software Components	42
Compiler:Event Recorder	42
Compiler:I/O.....	43
Board Support.....	45
Create Applications	46
Blinky with Keil RTX5	46
Blinky with Infinite Loop Design.....	54
Device Startup Variations.....	56
Example: STM32Cube	56
Secure/non-secure programming.....	61
Create ARMv8-M software projects.....	61

Debug Applications	62
Debugger Connection	62
Using the Debugger	63
Debug Toolbar	64
Command Window	65
Disassembly Window	65
Component Viewer	66
Event Recorder	67
Breakpoints	69
Watch Window	70
Call Stack and Locals Window	70
Register Window	71
Memory Window	71
Peripheral Registers	72
Trace	73
Trace with Serial Wire Output	74
Trace Exceptions	76
Logic Analyzer	77
Debug (printf) Viewer	78
Event Counters	79
Trace with 4-Pin Output	80
Trace with On-Chip Trace Buffer	80
Middleware	81
Network Component	83
File System Component	85
USB Component	86
Graphics Component	87
IoT Connectivity	88
Migrating to Middleware Version 7	89
FTP Server Example	90
Using Middleware	92
USB Device HID Example	94
Add Software Components	95
Configure Middleware	97
Configure Drivers	99
Implement Application Features	100
Build and Download	103
Verify and Debug	103
Index	105

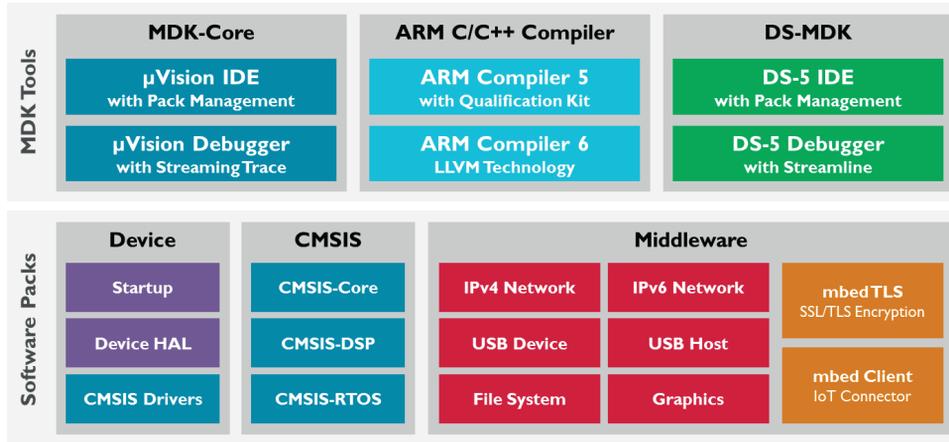
NOTE

This user's guide describes how to create projects for ARM Cortex-M microcontrollers using the μ Vision IDE/Debugger.

*Refer to the **Getting Started with DS-MDK** user's guide for information how to create applications with the Eclipse-based DS-5 IDE/Debugger for ARM Cortex-A/Cortex-M devices.*

MDK Introduction

MDK helps you to create embedded applications for ARM Cortex-M processor-based devices. MDK is a powerful, yet easy to learn and use development system. It consists of MDK-Core and software packs, which can be downloaded and installed based on the requirements of your application.



MDK Tools

The MDK Tools include all the components that you need to create, build, and debug an embedded application for ARM based microcontroller devices.

MDK-Core consists of the genuine Keil μ Vision IDE and debugger with leading support for Cortex-M processor-based microcontroller devices including the new ARMv8-M architecture. **DS-MDK** contains the Eclipse-based DS-5 IDE and debugger and offers multi-processor support for devices based on 32-bit Cortex-A processors or hybrid systems with 32-bit Cortex-A and Cortex-M processors.

MDK includes two **ARM C/C++ Compilers** with assembler, linker, and highly optimize run-time libraries tailored for optimum code size and performance:

- ARM Compiler version 5 is the reference C/C++ compiler available with a TÜV certified qualification kit for safety applications, as well as long-term support and maintenance.
- ARM Compiler version 6 is based on the innovative LLVM technology and supports the latest C language standards including C++11 and C++14. It offers the smallest size and highest performance for Cortex-M targets.

Software Packs

Software packs contain device support, CMSIS libraries, middleware, board support, code templates, and example projects. They may be added any time to MDK-Core or DS-MDK, making new device support and middleware updates independent from the toolchain. The IDE manages the provided software components that are available for the application as building blocks.

MDK Editions

The product selector, available at www.keil.com/editions, gives an overview of the features enabled in each edition:

- **MDK-Lite** is code size restricted to 32 KByte and intended for product evaluation, small projects, and the educational market.
- **MDK-Essential** supports Cortex-M processor-based microcontrollers up to Cortex-M7 and non-secure programming of Cortex-M23 and M33 targets.
- **MDK-Plus** adds middleware libraries for IPv4 networking, USB Device, File System, and Graphics. It supports ARM Cortex-M, selected ARM Cortex-R, ARM7, and ARM9 processor based microcontrollers. It also includes DS-MDK for programming heterogeneous devices.
- **MDK-Professional** contains all features of **MDK-Plus**. In addition, it supports IPv4/IPv6 dual-stack networking, IoT connectivity, and a USB Host stack. It also offers secure and non-secure programming of Cortex-M23 and M33 targets as well as multicore debugging of heterogeneous devices including the Linux kernel and Streamline performance analysis.

License Types

With the exception of **MDK-Lite**, all MDK editions require activation using a license code. The following licenses types are available:

Single-user license (node-locked) grants the right to use the product by one developer on two computers at the same time.

Floating-user license or **FlexNet license** grants the right to use the product on several computers by a number of developers at the same time.

For further details, refer to the *Licensing User's Guide* at www.keil.com/support/man/docs/license.

Installation

Software and hardware requirements

MDK has the following minimum hardware and software requirements:

- A PC running a current Microsoft Windows desktop operating system (32-bit or 64-bit)
- 4 GB RAM and 8 GB hard-disk space
- 1280 x 800 or higher screen resolution; a mouse or other pointing device

Install MDK-Core

Download MDK from www.keil.com/download - Product Downloads and run the installer.

Follow the instructions to install MDK-Core on your local computer. The installation also adds the software packs for ARM **CMSIS** and MDK **Middleware**.

MDK version 5 is capable of using MDK version 4 projects after installation of the legacy support from www.keil.com/mdk5/legacy. This adds support for ARM7, ARM9, and Cortex-R processor-based devices.

After the MDK-Core installation is complete, the **Pack Installer** starts automatically, which allows you to add supplementary software packs. As a minimum, you need to install a software pack that supports your target microcontroller device.

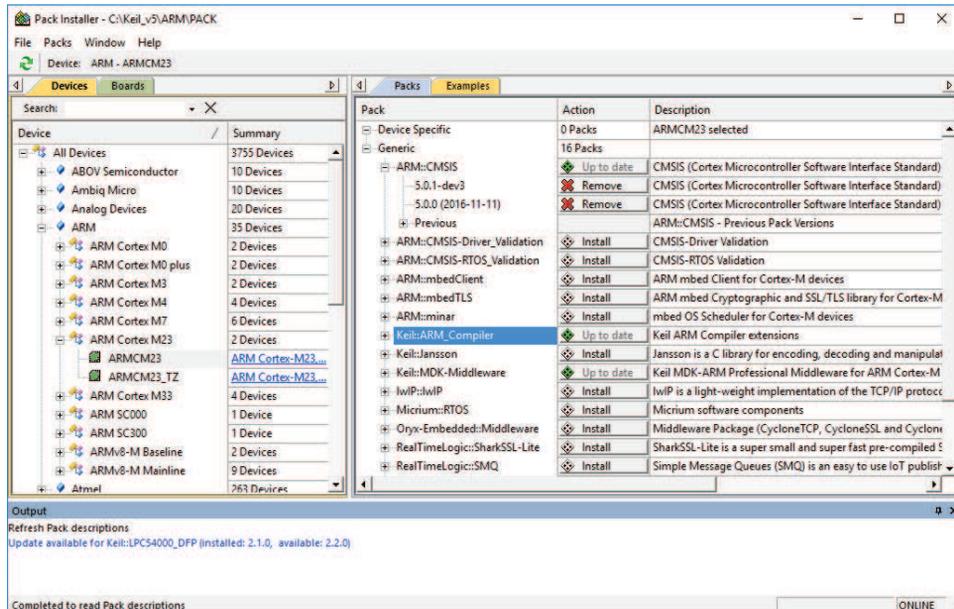
Install Software Packs

The **Pack Installer** manages software packs on the local computer.

 The **Pack Installer** runs automatically during the installation, but also can be run from μ Vision using the menu item **Project – Manage – Pack Installer**. To get access to devices and example projects, install the software pack related to your target device or evaluation board.

NOTE

To obtain information of published software packs the Pack Installer connects to www.keil.com/pack.



The status bar, located at the bottom of the Pack Installer, shows information about the Internet connection and the installation progress.

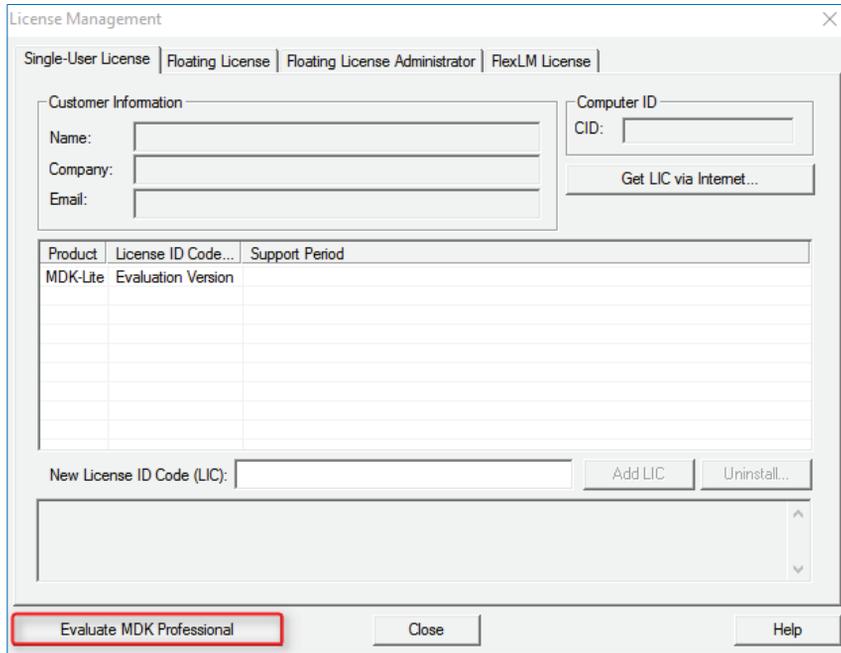
TIP: The device database at www.keil.com/dd2 lists all available devices and provides download access to the related software packs. If the Pack Installer cannot access www.keil.com/pack you can manually install software packs using the menu command **File – Import** or by double-clicking *.PACK files.

MDK-Professional Trial License

MDK has a built-in **free** seven-day trial license for MDK-Professional. This removes the code size limits and you can explore and test the comprehensive middleware.

Start μ Vision with administration rights.

 In μ Vision, go to **File – License Management...** and click **Evaluate MDK Professional**



The screenshot shows the 'License Management' dialog box with the following elements:

- Tabs: Single-User License, Floating License, Floating License Administrator, FlexLM License
- Customer Information section: Name, Company, Email (text input fields)
- Computer ID section: CID (text input field), Get LIC via Internet... button
- Table with columns: Product, License ID Code..., Support Period
- Table content: MDK-Lite, Evaluation Version
- New License ID Code (LIC): (text input field), Add LIC, Uninstall... buttons
- Buttons at the bottom: Evaluate MDK Professional (highlighted with a red rectangle), Close, Help

 On the next screen, click **Start MDK Professional Evaluation for 7 Days**. After the installation, the screen displays information about the expiration date and time.

NOTE

*Activation of the 7-day MDK Professional trial version enables the option **Use Flex Server** in the tab **FlexLM License** as this license is based on FlexNet.*

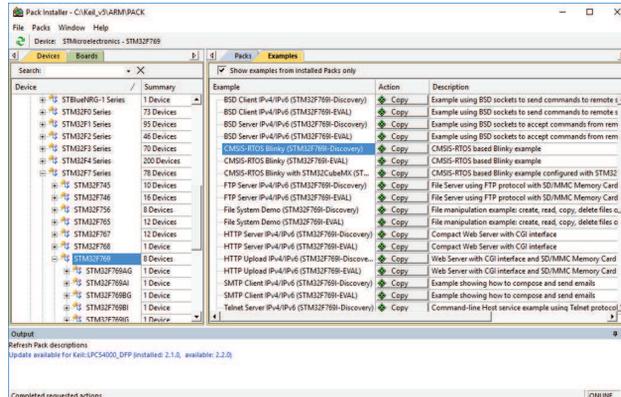
Verify Installation using Example Projects

Once you have selected, downloaded, and installed a software pack for your device, you can verify your installation using one of the examples provided in the software pack. To verify the software pack installation, we recommend using a *Blinky* example, which typically flashes LEDs on a target board.

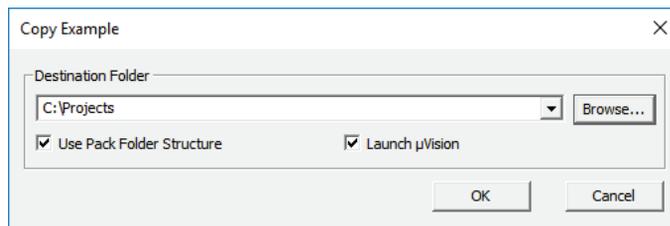
TIP: Review the getting started video on www.keil.com/mdk5/install that explains how to connect and work with an evaluation kit.

Copy an Example Project

 In the Pack Installer, select the tab **Examples**. Use filters in the toolbar to narrow the list of examples.



Click **Copy** and enter the **Destination Folder** name of your working directory.



NOTE

You must copy the example projects to a working directory of your choice.

Enable **Launch µVision** to open the example project directly in the IDE.

Enable **Use Pack Folder Structure** to copy example projects into a common folder. This avoids overwriting files from other example projects. Disable **Use Pack Folder Structure** to reduce the complexity of the example path.

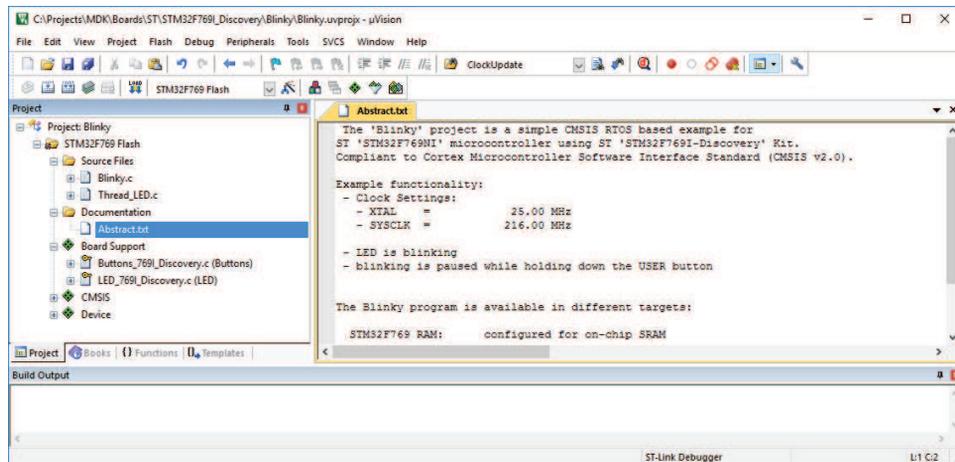
Click **OK** to start the copy process.

Use an Example Application with μ Vision

Now μ Vision starts and loads the example project where you can:

-  Build the application, which compiles and links the related source files.
-  Download the application, typically to on-chip Flash ROM of a device.
-  Run the application on the target hardware using a debugger.

The step-by-step instructions show you how to execute these tasks. After copying the example, μ Vision starts and looks similar to the picture below.



TIP: Most example projects contain an *Abstract.txt* file with essential information about the operation and hardware configuration.

Build the Application



Build the application using the toolbar button **Rebuild**.

The **Build Output** window shows information about the build process. An error-free build shows information about the program size.

```

Build Output
*** Using Compiler 'V5.06 update 4 (build 422)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Rebuild target 'STM32F769 Flash'
compiling Thread_LED.c...
compiling LED_769I_Discovery.c...
compiling Blinky.c...
compiling Buttons_769I_Discovery.c...
compiling RTX_Conf_CM.c...
compiling stm32f7xx_hal_cortex.c...
compiling stm32f7xx_hal.c...
compiling stm32f7xx_hal_gpio.c...
compiling stm32f7xx_hal_pwr_ex.c...
compiling stm32f7xx_hal_pwr.c...
assembling startup_stm32f769xx.s...
compiling stm32f7xx_hal_rcc.c...
compiling system_stm32f7xx.c...
compiling stm32f7xx_hal_rcc_ex.c...
linking...
Program Size: Code=10288 RO-data=696 RW-data=68 ZI-data=4756
".\Flash\Blinky.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:09
  
```

Download the Application

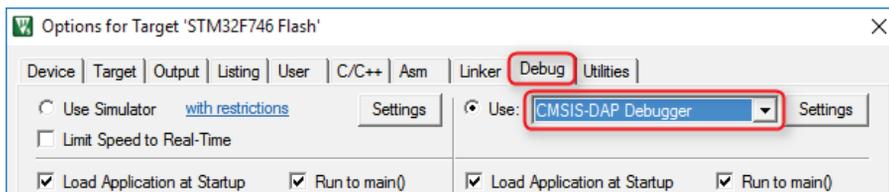
Connect the target hardware to your computer using a *debug adapter* that typically connects via USB. Several evaluation boards provide an on-board debug adapter.



Now, review the settings for the debug adapter. Typically, example projects are pre-configured for evaluation kits; thus, you do not need to modify these settings.



Click **Options for Target** on the toolbar and select the **Debug** tab. Verify that the correct debug adapter of the evaluation board you are using is selected and enabled. For example, **CMSIS-DAP Debugger** is a debug adapter that is part of several starter kits.



 Enable **Load Application at Startup** for loading the application into the μ Vision debugger whenever a debugging session is started.

Enable **Run to main()** for executing the instructions up to the first executable statement of the main() function. The instructions are executed upon each reset.

TIP: Click the button **Settings** to verify communication settings and diagnose problems with your target hardware. For further details, click the button **Help** in the dialogs. If you have any problems, refer to the user guide of the starter kit.

 Click **Download** on the toolbar to load the application to your target hardware.



```
Build Output
Load "C:\\Workspaces\\MDK\\STM32\\MDK\\Boards\\ST\\STM32F746G_Discovery\\Blinky\\Flash\\Blinky.axf"
Erase Done.
Programming Done.
Verify OK.
Application running ...
Flash Load finished at 14:38:29
```

The **Build Output** window shows information about the download progress.

Run the Application

 Click **Start/Stop Debug Session** on the toolbar to start debugging the application on hardware.

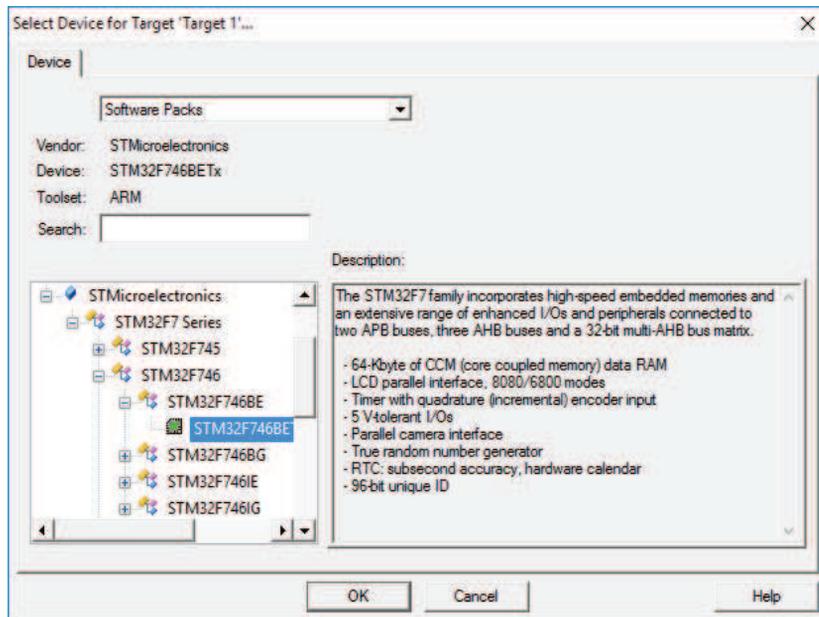
 Click **Run** on the debug toolbar to start executing the application. LEDs should flash on the target hardware.

Use Software Packs

Software packs contain information about microcontroller devices and software components that are available for the application as building blocks.

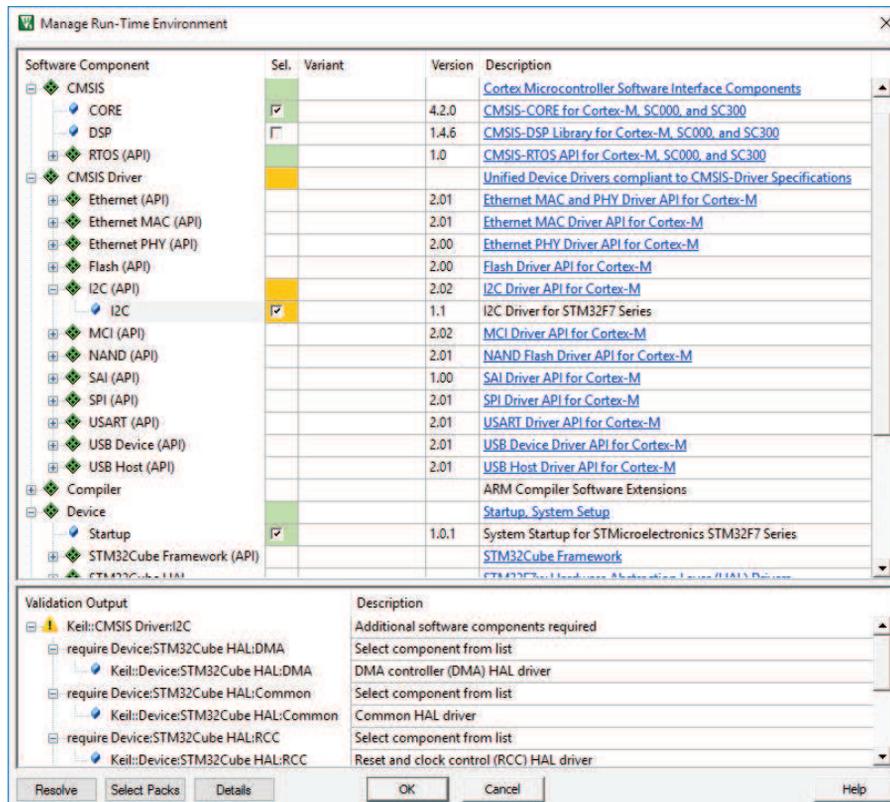
The device information pre-configures development tools for you and shows only the options that are relevant for the selected device.

 Start μ Vision and use the menu **Project - New μ Vision Project**. After you have selected a project directory and specified the project name, select a target device.



TIP: Only devices that are part of the installed software packs are shown. If you are missing a device, use the Pack Installer to add the related software pack. The search box helps you to narrow down the list of devices.

- ◆ After selecting the device, the **Manage Run-Time Environment** window shows the related software components for this device.



TIP: The links in the column *Description* provide access to the documentation of each software component.

NOTE

The notation `::<Component Class>:<Group>:<Name>` is used to refer to components. For example, `::CMSIS:CORE` refers to the component `CMSIS-CORE` selected in the dialog above.

Software Component Overview

The following table shows the software components for a typical installation. Depending on your selected device, some of these software components might not be visible in the Manage Run-Time Environment window. In case you have installed additional software packs, more software components will be available.

Software Component	Description	Page
Board Support	Interfaces to the peripherals of evaluation boards.	45
CMSIS	CMSIS interface components, such as CORE, DSP, and CMSIS-RTOS.	22
CMSIS Driver	Unified device drivers for middleware and user applications.	39
Compiler	ARM Compiler specific software components to retarget I/O operations for example for printf style debugging. Event recorder for debugging software components and user application code.	42
Device	System startup and low-level device drivers.	47
File System	Middleware component for file access on various storage device types.	85
Graphics	Middleware component for creating graphical user interfaces.	87
Network	Middleware component for TCP/IP networking using Ethernet or serial protocols.	83
USB	Middleware component for USB Host and USB Device supporting standard USB Device classes.	86

Product Lifecycle Management with Software Packs

MDK allows you to install multiple versions of a software pack. This enables product lifecycle management (PLM) as it is common for many projects.

There are four distinct phases of PLM:

Concept: Definition of major project requirements and exploration with a functional prototype.

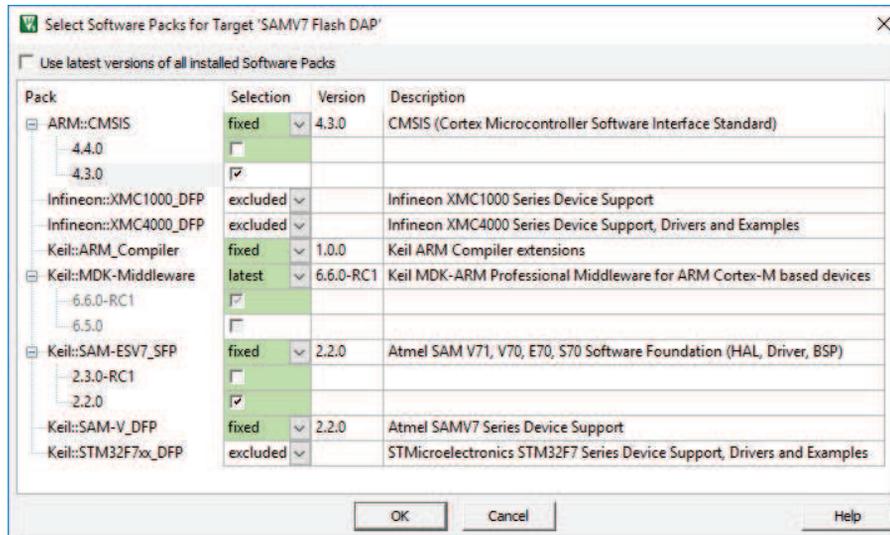
Design: Prototype testing and implementation of the product based on the final technical features and requirements.

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.

In the concept and design phase, you normally want to use the latest software packs to be able to incorporate new features and bug fixes quickly. Before product release, you will freeze the software components to a known tested state. In the product service phase, use the fixed versions of the software components to support customers in the field.

 The dialog **Select Software Packs** helps you to manage the versions of each software pack in your project:



When the project is completed, disable the option **Use latest version of all installed Software Packs** and specify the software packs with the settings under **Selection**:

latest: use the latest version of a software pack. Software components are updated when a newer software pack version is installed.

fixed: specify an installed version of the software pack. Software components in the project target will use these versions.

excluded: no software components from this software pack are used.

The colors indicate the usage of software components in the current project target:

 Some software components from this pack are used.

 Some software components from this pack are used, but the pack is excluded.

 No software component from this pack is used.

Software Version Control Systems (SVCS)

μ Vision carries template files for GIT, SVN, CVS, and others to support Software Version Control Systems (SVCS).

Application note 279 “Using Git for Project Management with μ Vision” (www.keil.com/appnotes/docs/apnt_279.asp) describes how to establish a robust workflow for version control of projects using software packs.

Access Documentation

MDK provides online manuals and context-sensitive help. The μ Vision **Help** menu opens the main help system that includes the *μ Vision User’s Guide*, getting started manuals, compiler, linker and assembler reference guides.

Many dialogs have context-sensitive **Help** buttons that access the documentation and explain dialog options and settings.

You can press **F1** in the editor to access help on language elements like RTOS functions, compiler directives, or library routines. Use **F1** in the command line of the **Output** window for help on debug commands, and some error and warning messages.

The **Books** window may include device reference guides, data sheets, or board manuals. You can even add your own documentation and enable it in the **Books** window using the menu **Project – Manage – Components, Environment, Books – Books**.

The **Manage Run-Time Environment** dialog offers access to documentation via links in the *Description* column.

In the **Project** window, you can right-click a software component group and open the documentation of the corresponding element.

You can access the **μ Vision User’s Guide** on-line at www.keil.com/support/man/docs/uv4.

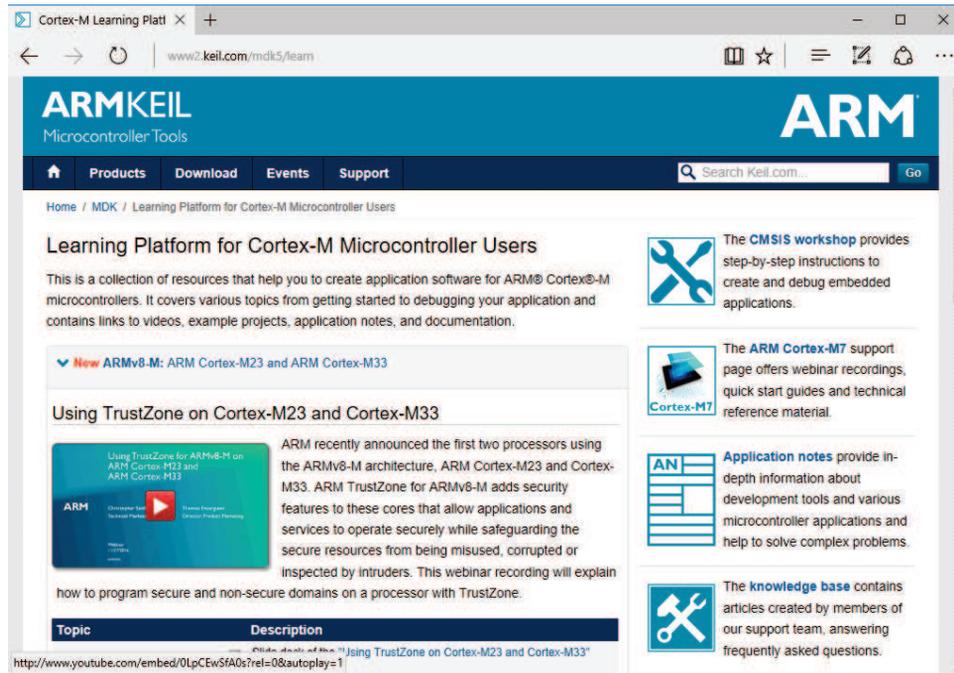
Request Assistance

If you have suggestions or you have discovered an issue with the software, please report them to us. Support and information channels are accessible at www.keil.com/support.

When reporting an issue, include your license code (if you have one) and product version, available from the μ Vision menu **Help – About**.

Learning Platform

Our www.keil.com/learn website helps you to learn more about the programming of ARM Cortex-based microcontrollers. It contains tutorials, videos, further documentation, as well as useful links to other websites.



The screenshot shows the ARMKEIL website's learning platform. The header includes the ARMKEIL logo and navigation tabs for Home, Products, Download, Events, and Support. A search bar is located in the top right. The main content area is titled "Learning Platform for Cortex-M Microcontroller Users" and provides an overview of resources. A featured video titled "Using TrustZone on Cortex-M23 and Cortex-M33" is displayed, with a description explaining ARM's TrustZone security features. To the right, four resource cards are listed: "CMSIS workshop" (step-by-step instructions), "ARM Cortex-M7 support" (webinar recordings and guides), "Application notes" (development tools and technical info), and "Knowledge base" (support team articles).

Quick Start Guides

Quick start guides help you to bring up your target hardware quickly. They describe the required steps to get a development board up and running with MDK and list required software packs as well as driver requirements for integrated debug adapters.

NOTE

www.keil.com/mdk5/qsg explains how to download the quick start guides

CMSIS

The **Cortex Microcontroller Software Interface Standard** (CMSIS) provides a ground-up software framework for embedded applications that run on Cortex-M based microcontrollers. CMSIS enables consistent and simple software interfaces to the processor and the peripherals, simplifying software reuse, reducing the learning curve for microcontroller developers.

CMSIS is available under an Apache 2.0 license and is publicly developed on GitHub: https://github.com/ARM-software/CMSIS_5.

NOTE

This chapter is a reference section. The chapter [Create Applications](#) on page 46 shows you how to use CMSIS for creating application code.

CMSIS provides a common approach to interface peripherals, real-time operating systems, and middleware components. The CMSIS application software components are:

- **CMSIS-CORE**: Defines the API for the Cortex-M processor core and peripherals and includes a consistent system startup code. The software components `::CMSIS:CORE` and `::Device:Startup` are all you need to create and run applications on the native processor that uses exceptions, interrupts, and device peripherals.
- **CMSIS-RTOS2**: Provides a standardized real-time operating system API and enables software templates, middleware, libraries, and other components that can work across supported RTOS systems. This manual explains the usage of the Keil RTX5 implementation.
- **CMSIS-DSP**: Is a library collection for digital signal processing (DSP) with over 60 Functions for various data types: fix-point (fractional q7, q15, q31) and single precision floating-point (32-bit).
- **CMSIS-Driver**: Is a software API that describes peripheral driver interfaces for middleware stacks and user applications. The CMSIS-Driver API is designed to be generic and independent of a specific RTOS making it reusable across a wide range of supported microcontroller devices.

CMSIS-CORE

This section explains the usage of CMSIS-CORE in applications that run natively on a Cortex-M processor. This type of operation is known as *bare-metal*, because it does not use a real-time operating system.

Using CMSIS-CORE

A native Cortex-M application with CMSIS uses the software component **::CMSIS:CORE**, which should be used together with the software component **::Device:Startup**. These components provide the following central files:

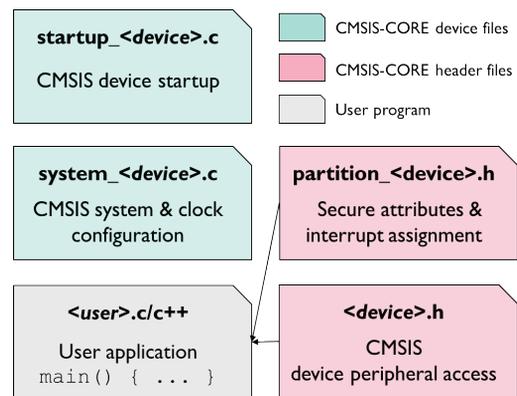
The *startup_<device>.s* file with reset handler and exception vectors.

The *system_<device>.c* configuration file for basic device setup (clock and memory bus).

The *<device>.h* header file for user code access to the microcontroller device. This file is included in C source files and defines:

- Peripheral access with standardized register layout.
- Access to interrupts and exceptions, and the Nested Interrupt Vector Controller (NVIC).
- Intrinsic functions to generate special instructions, for example to activate sleep mode.
- SysTick timer (SYSTICK) functions to configure and start a periodic timer interrupt.
- Debug access for *printf*-style I/O and ITM communication via on-chip CoreSight.

The *partition_<device>.h* header file contains the initial setup of the TrustZone hardware in an ARMv8-M system (refer to chapter **Secure/non-secure programming**).

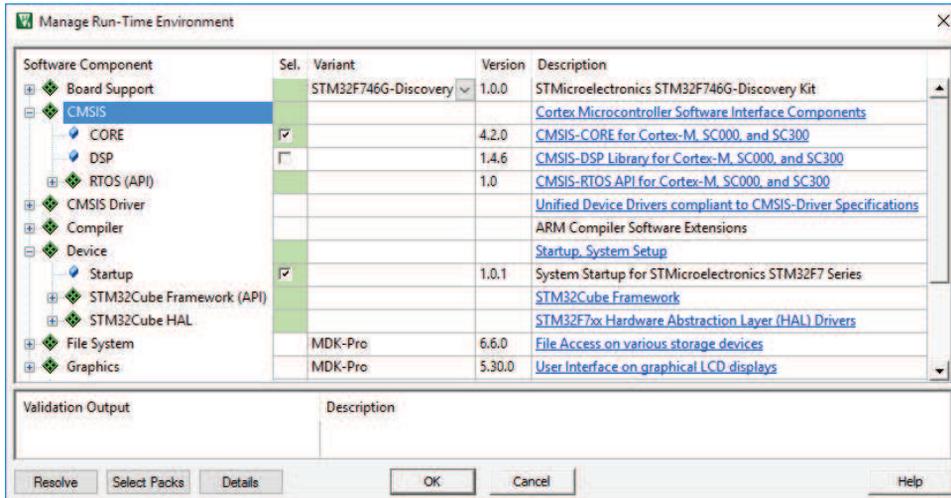


NOTE

In actual file names, <device> is the name of the microcontroller device.

Adding Software Components to the Project

The files for the components `::CMSIS:CORE` and `::Device:Startup` are added to a project using the μ Vision dialog **Manage Run-Time Environment**. Just select the software components as shown below:



The μ Vision environment adds the related files.

Source Code Example

The following source code lines show the usage of the CMSIS-CORE layer.

Example of using the CMSIS-CORE layer

```
#include "stm32f4xx.h" // File name depends on device used

uint32_t volatile msTicks; // Counter for millisecond Interval
uint32_t volatile frequency; // Frequency for timer

void SysTick_Handler (void) { // SysTick Interrupt Handler
    msTicks++; // Increment Counter
}

void WaitForTick (void) {
    uint32_t curTicks;
    curTicks = msTicks; // Save Current SysTick Value
    while (msTicks == curTicks) { // Wait for next SysTick Interrupt
        __WFE (); // Power-Down until next Event
    }
}

void TIM1_UP_IRQHandler (void) { // Timer Interrupt Handler
    ; // Add user code here
}
```

```

void timer1_init(int frequency) { // Set up Timer (device specific)
    NVIC_SetPriority (TIM1_UP_IRQn, 1); // Set Timer priority
    NVIC_EnableIRQ (TIM1_UP_IRQn); // Enable Timer Interrupt
}

// Configure & Initialize the MCU
void Device_Initialization (void) {
    if (SysTick_Config (SystemCoreClock / 1000)) { // SysTick lms
        : // Handle Error
    }
    timer1_init (frequency); // Setup device-specific timer
}

// The processor clock is initialized by CMSIS startup + system file
int main (void) { // User application starts here
    Device_Initialization (); // Configure & Initialize MCU

    while (1) { // Endless Loop (the Super-Loop)
        __disable_irq (); // Disable all interrupts
        // Get_InputValues ();
        __enable_irq (); // Enable all interrupts
        // Process_Values ();
        WaitForTick (); // Synchronize to SysTick Timer
    }
}

```

For more information, right-click the group CMSIS in the Project window, and choose **Open Documentation**, or refer to the CMSIS-CORE documentation www.keil.com/cmsis/core.

The screenshot shows a web browser window displaying the CMSIS-CORE documentation. The browser address bar shows the URL `keil.com/pack/doc/CMSIS/Core/html/index.html`. The page title is "CMSIS-CORE Version 5.0.0" and the subtitle is "CMSIS-CORE support for Cortex-M processor-based devices". The navigation menu includes "General", "Core", "Driver", "DSP", "RTOS v1", "RTOS v2", "Pack", "SVD", and "DAP". The "Core" section is expanded, showing "Main Page", "Usage and Description", and "Reference". The "Overview" page is selected, displaying the following content:

Overview

CMSIS-CORE implements the basic run-time system for a Cortex-M device and gives the user access to the processor core and the device peripherals. In detail it defines:

- **Hardware Abstraction Layer (HAL)** for Cortex-M processor registers with standardized definitions for the SysTick, NVIC, System Control Block registers, MPU registers, FPU registers, and core access functions.
- **System exception names** to interface to system exceptions without having compatibility issues.
- **Methods to organize header files** that makes it easy to learn new Cortex-M microcontroller products and improve software portability. This includes naming conventions for device-specific interrupts.
- **Methods for system initialization** to be used by each MCU vendor. For example, the standardized **SystemInit()** function is essential for configuring the clock system of the device.
- **Intrinsic functions** used to generate CPU instructions that are not supported by standard C functions.
- A variable to determine the **system clock frequency** which simplifies the setup the SysTick timer.

Generated on Fri Nov 11 2016 12:41:20 for CMSIS-CORE by ARM Ltd. All rights reserved.