arm CHINA

Application Note

Using the Caches of STAR with CMSIS CMO Functions

Version 1.0
Document ID: ACN-02202102-001
Non-Confidential

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice 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 China. **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 implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM CHINA 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 China makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM CHINA 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 CHINA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any 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 China’s customers is not intended to create or refer to any partnership relationship with any other company. Arm China may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm China, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. 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 the Agreement shall prevail.

The Arm China corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Technology (China) Co., Ltd (or its affiliates) in the People’s Republic of China and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Copyright © 2021 Arm China (or its affiliates). All rights reserved.

Copyright © 2021 Arm China. All rights reserved.

Release Information

Document History

Issue	Date	Confidentiality	Change
A	23/07/2021	Non-Confidential	Initial draft

Contents

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

1 About this document

This Application Note is intended for developers/programmers/users who use the Arm China STAR *Device Family Pack* (DFP). This Application Note gives you a basic understanding of *Cache Maintenance Operation* (CMO) functions of the caches in STAR and provides guidance on how to use the example project which is using CMO functions.

1.1 References

Reference	Document number	Title
-	-	-

1.2 Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
CMSIS	Cortex Microcontroller Software Interface Standard
DFP	Device Family Pack
CMO	Cache Maintenance Operation
CoreMark	CoreMark® is an industry-standard benchmark that measures the performance of central processing units (CPUs) and embedded microcontrollers (MCUs).

1.3 Conventions and feedback

The following describes the typographical conventions and how to give feedback:

Convention	Meaning
monospace	denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
monospace	denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
monospace italic	denotes arguments to commands and functions where the argument is to be replaced by a specific value.
monospace bold	denotes language keywords when used outside example code.
italic	highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate, and for Arm China processor signal names.

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

1.3.1 Feedback on this product

If you have any comments and suggestions about this product, contact your supplier and give:

- Your name and company.
- The serial number of the product.
- Details of the release you are using.
- Details of the platform you are using, such as the hardware platform, operating system type and version.
- A small standalone sample of code that reproduces the problem.
- A clear explanation of what you expected to happen, and what actually happened.
- The commands you used, including any command-line options.
- Sample output illustrating the problem.
- The version string of the tools, including the version number and build numbers.

1.3.2 Feedback on documentation

If you have comments on the documentation, e-mail errata@armchina.com. Give:

- The title.
- The number, [Document ID Value], [Issue].
- If viewing online, the topic names to which your comments apply.
- If viewing a PDF version of a document, the page numbers to which your comments apply.
- A concise explanation of your comments.

Arm China also welcomes general suggestions for additions and improvements.

Arm China periodically provides updates and corrections to its documentation on the Arm China Information Center, together with knowledge articles and *Frequently Asked Questions* (FAQs).

1.3.3 Other information

- Arm Glossary, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

2 Introduction

2.1 CMSIS

The *Cortex Microcontroller Software Interface Standard* (CMSIS) is a vendor-independent hardware abstraction layer for microcontrollers.

The CMSIS defines generic tool interfaces and enables consistent device support.

The CMSIS provides:

- Simple software interfaces to processor and peripherals.
- A common approach to interface to peripherals, real-time operating systems, and middleware components.

2.2 STAR DFP

For CMSIS compliant toolchains such as Keil MDK, IAR EW and Development Studio, additional software components and support for microcontroller devices are provided by software packs.

A DFP is one of the CMSIS software packs. It indicates that a software pack contains support for microcontroller devices.

A DFP provides essential support for the software targets on a specific device, such as 'startup', 'system', linker scripts, and debug configuration.

The STAR processor is the first processor in the Arm China STAR series processor family.

STAR is a fully featured microcontroller class processor based on the Armv8-M mainline architecture with Arm® TrustZone® technology (depending on the actual core).

In STAR CMSIS DFP v1.3.0 and later, there are example projects of STAR application. These example projects can help you quickly build projects and run application software.

2.3 CoreMark

CoreMark is a simple, yet sophisticated benchmark that is designed specifically to test the functionality of a processor core. Running CoreMark produces a single-number score allowing users to make quick performance comparisons between processors.

Typically the total binary size of the pure CoreMark codes is no more than 16K using gcc on an x86 machine. The small size of CoreMark allows it to easily fit in a processor's cache, which means it is suitable for testing on a wide range of processors, from low-end to high-end devices, and you can use it to observe the performance gap between 'with' and 'without' caches.

The copyright of CoreMark is owned by EEMBC. For more information about EEMBC and CoreMark, visit https://www.eembc.org/coremark/.

Note that CoreMark is licensed under Apache V2.0 license.

Before you use CoreMark and STAR DFP, you must read the license and then keep in compliance with the license (http://www.apache.org/licenses/LICENSE-2.0).

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

3 Preparations

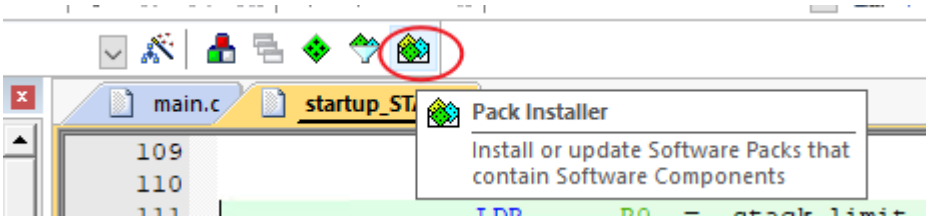
The example project of the application software will run on an MPS2 FPGA board.

Before using the example project, you need to:

- Ensure that you have an MPS2/MPS2+ FPGA board and had a STAR-based device implemented on the board.
- Check the STAR CMSIS DFP version.

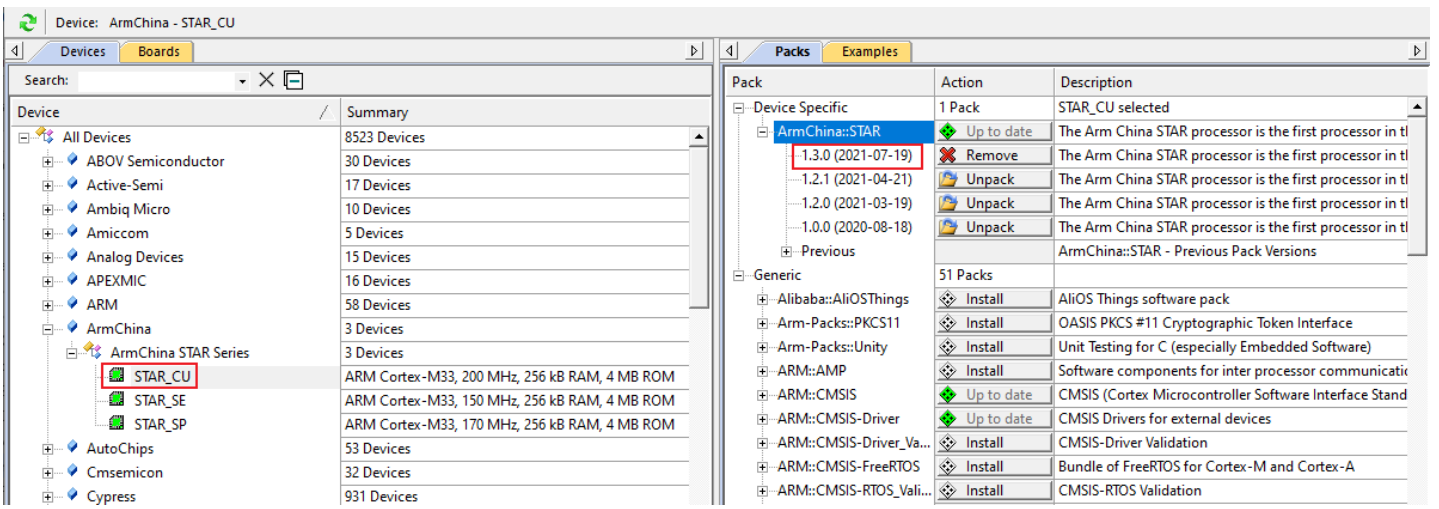
To check the STAR CMSIS DFP version:

1. Start MDK.
2. On the toolbar, click the **Pack Installer** icon.



3. On the **Devices** tab, select a device (for example, STAR_CU) and check the version of the installed pack.

   As shown in the following figure, the version of the ArmChinaSTAR pack should be 1.3.0 or later.



Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

4 CMO functions

All cache maintenance operations are executed by writing to registers in the memory-mapped *System Control Space* (SCS) region of the internal PPB memory space.

The operations supported for the data cache are:

- Enable

  Prototype: __STATIC_FORCEINLINE void SCB_EnableDCache (void)

- Disable

  Prototype: __STATIC_FORCEINLINE void SCB_DisableDCache (void)

- Invalidate

  Prototype: __STATIC_FORCEINLINE void SCB_InvalidateDCache (void)

- Clean

  Prototype: __STATIC_FORCEINLINE void SCB_CleanDCache (void)

- Clean and Invalidate

  Prototype: __STATIC_FORCEINLINE void SCB_CleanInvalidateDCache (void)

- Invalidate by address (Set/Way combination)

  Prototype: __STATIC_FORCEINLINE void SCB_InvalidateDCache_by_Addr (void *addr, int32_t dsize)

  D-cache is invalidated starting from a 32-byte aligned address in 32-byte granularity.

  D-cache memory blocks which are part of given address + given size are invalidated.

- Clean by address

  Prototype: __STATIC_FORCEINLINE void SCB_CleanDCache_by_Addr (uint32_t *addr, int32_t dsize)

  This function is to clean D-Cache for the given address.

  D-cache is cleaned starting from a 32-byte aligned address in 32-byte granularity.

  D-cache memory blocks which are part of given address + given size are cleaned.

- Clean and Invalidate by address (Set/Way combination)

  Prototype: __STATIC_FORCEINLINE void SCB_CleanInvalidateDCache_by_Addr (uint32_t *addr, int32_t dsize)

  D-cache is cleaned and invalidated starting from a 32-byte aligned address in 32-byte granularity.

  D-cache memory blocks which are part of given address + given size are cleaned and invalidated.

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

The CMO functions for operations supported for the instruction cache are:

- Enable

  Prototype: __STATIC_FORCEINLINE void SCB_EnableICache (void)

- Disable

  Prototype: __STATIC_FORCEINLINE void SCB_DisableICache (void)

- Invalidate all

  Prototype: __STATIC_FORCEINLINE void SCB_InvalidateICache (void)

- Invalidate by address

  Prototype: __STATIC_FORCEINLINE void SCB_InvalidateICache_by_Addr (void *addr, int32_t isize)

  I-cache is invalidated starting from a 32-byte aligned address in 32-byte granularity.

  I-cache memory blocks which are part of given address + given size are invalidated.

Some principles for CMO:

- After you enable or disable the instruction cache, you must issue an ISB instruction to flush the pipeline. This ensures that all subsequent instruction fetches see the effect of enabling or disabling the instruction cache.
- After reset, you must invalidate each cache before enabling it. If the INITL1RSTDIS signal is de-asserted upon reset de-assertion, the cache invalidation work for both I-cache and D-cache is done automatically by hardware. When the hardware performs such invalidate-all function, it is done in the background by marking an Invalidation Region to yield the cache memory to the foreground activities from PFU or DPU, while every access that hits the Invalidation Region is automatically turned into a non-cacheable access.
- When disabling the data cache, you must clean the entire cache to ensure that any dirty data is flushed to external memory.
- Before enabling the data cache, you must invalidate the entire data cache because external memory might have changed when the cache was disabled.
- Before enabling the instruction cache, you must invalidate the entire instruction cache if external memory might have changed after the cache was disabled.

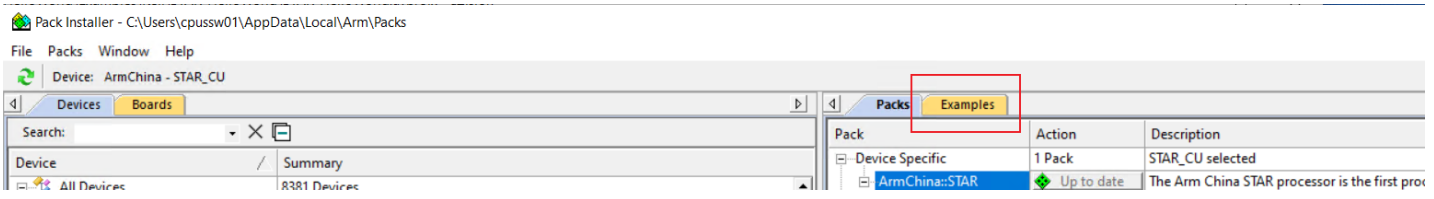
Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

5 Using the example project of STAR CMO

In STAR DFP v1.3.0, there is an example project which demonstrates how to use the CMO functions.

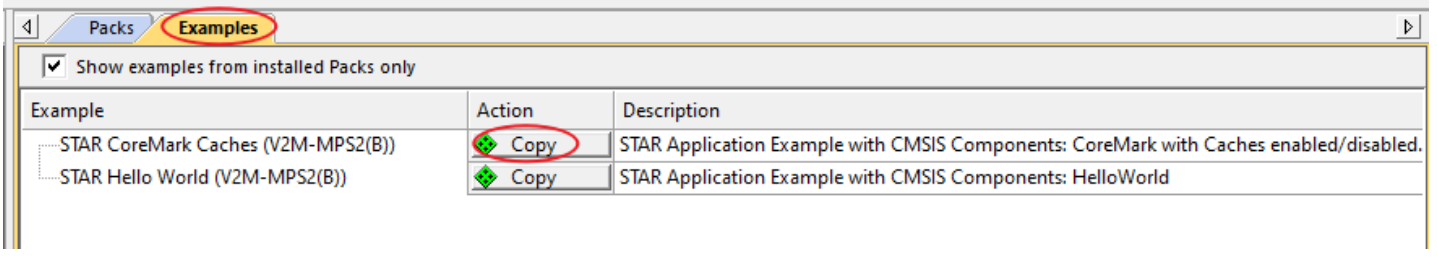
Follow these steps to get started with this example project and understand the cache maintenance operations:

1. In the Pack Installer, click the **Examples** tab.



2. On the **Examples** tab, select the example that you want to use and click **Copy**.

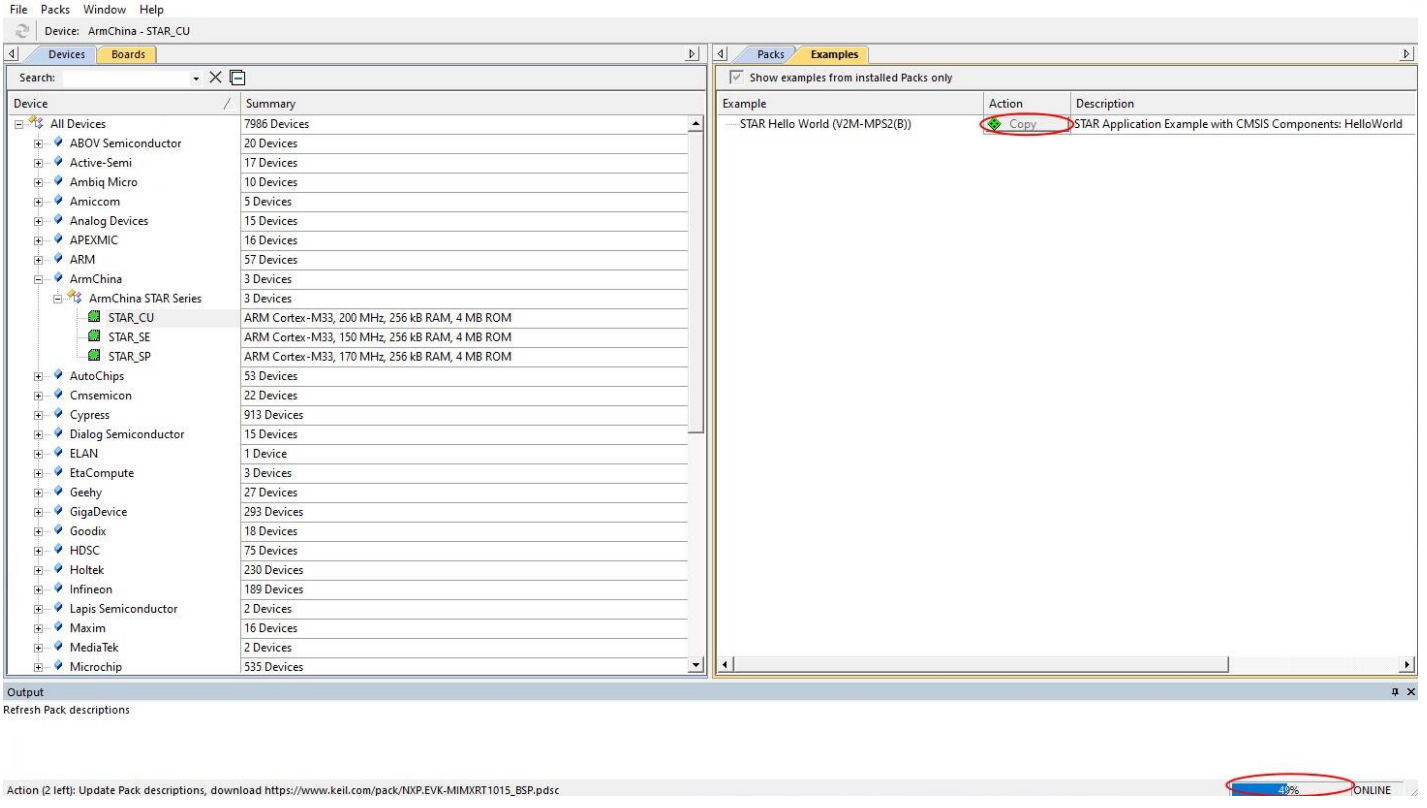
   In STAR CMSIS DFP v1.3.0, there is an example named 'STAR CoreMark Caches' available. This example project invokes CoreMark programs to stress on STAR, and to print out the two scores of enabling and disabling I-cache and D-cache.



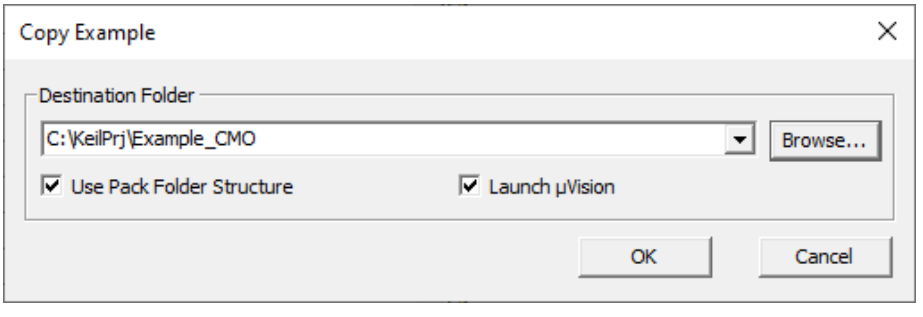
Note:

Sometimes the Copy button is disabled in gray because there are some packs need to be updated. You can check the progress bar to confirm this situation. When the progress reaches 100%, the Copy button will be enabled.

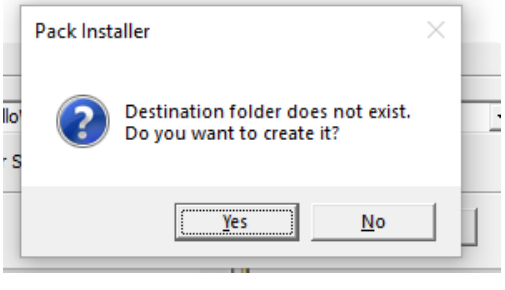
Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential



3. In the **Copy Example** dialog box that appears, specify the destination folder path to save the project, and then click **OK**.



If the destination folder does not exist, click **Yes** to create it.

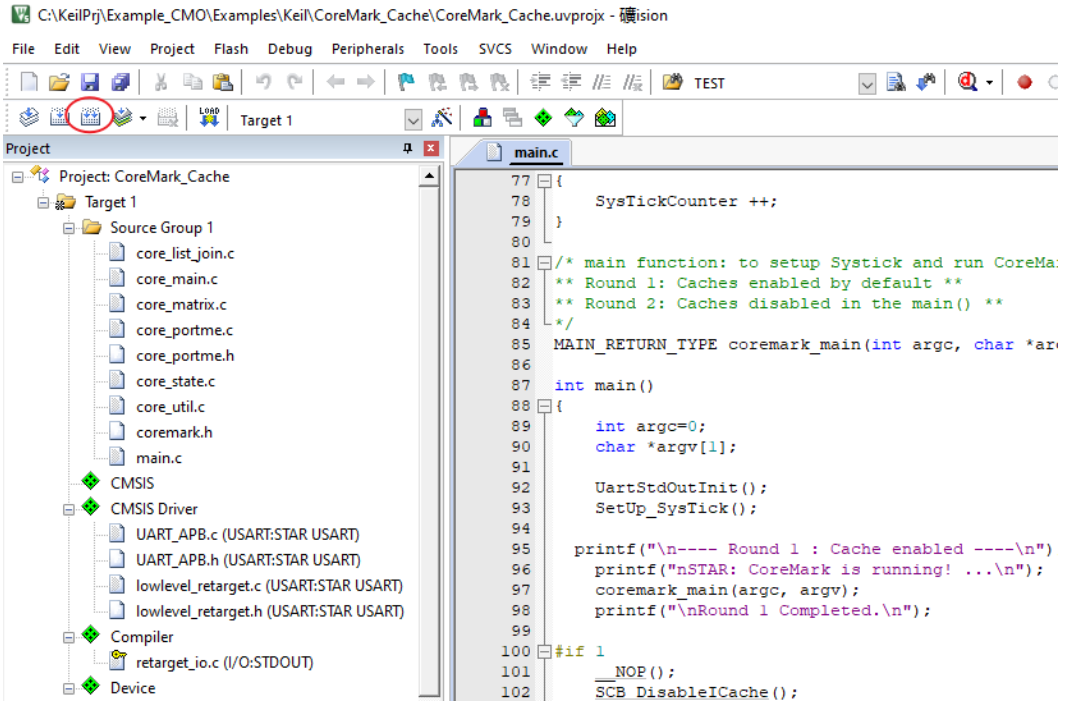


A project is created in the destination folder.

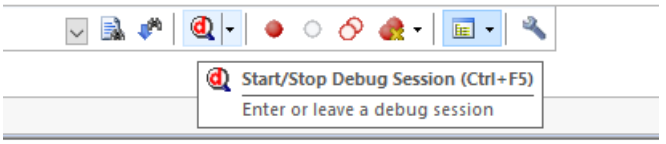
The MDK µVision will start automatically and open the created project. In the Project pane, you can see all the required files.

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

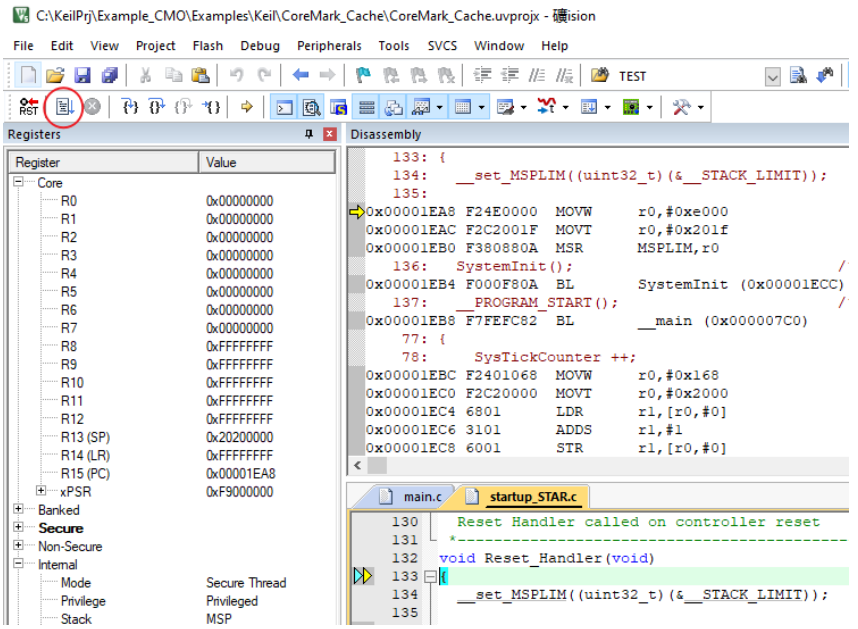
4. Click the **Rebuild** icon to recompile and build the project.



5. Click **Start/Stop Debug Session** to start the debug session.



6. Click the **Run** icon to run the built software.



Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

Then you can see the following message in the UART terminal window.

COM23 - PuTTY

```
---- Round 1 : Cache enabled ----
nSTAR: CoreMark is running! ...
```

You can observe the difference between enabling and disabling the caches for CoreMark.

COM23 - PuTTY

```
---- Round 1 : Cache enabled ----
nSTAR: CoreMark is running! ...
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 13544
Total time (secs): 13.544000
Iterations/Sec   : 81.216775
Iterations       : 1100
Compiler version : Clang 12.0.0 (ssh://ds-gerrit/armcompiler/llvm-project e64d64
4232aba72041b013571dc92e0d3fb7b4e2)
Compiler flags   : -Omax
Memory location  : STACK
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0x33ff
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 81.216775 / Clang 12.0.0 (ssh://ds-gerrit/armcompiler/llvm-projec
t e64d644232aba72041b013571dc92e0d3fb7b4e2) -Omax / STACK
CoreMark 1.0: 4.060839 Coremark/Mhz

Round 1 Completed.

---- Round 2 : Cache disabled ----
nSTAR: CoreMark is running! ...
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 17375
Total time (secs): 17.375000
Iterations/Sec   : 63.309353
Iterations       : 1100
Compiler version : Clang 12.0.0 (ssh://ds-gerrit/armcompiler/llvm-project e64d64423
2aba72041b013571dc92e0d3fb7b4e2)
Compiler flags   : -Omax
Memory location  : STACK
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0x33ff
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 63.309353 / Clang 12.0.0 (ssh://ds-gerrit/armcompiler/llvm-project e
64d644232aba72041b013571dc92e0d3fb7b4e2) -Omax / STACK
CoreMark 1.0: 3.165468 Coremark/Mhz

Round 2 Completed.
```

Note:

I-cache and D-cache are enabled in the `SystemInit()` function in `system_STAR.c`.

Alternatively, you can use compiler command flags `-D__EN_ICACHE="1"` and `-D__EN_DCACHE="1"` to enable the caches in this project.

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential

```
74  /*----------------------------------------------------------------------------
75    System initialization function
76   *----------------------------------------------------------------------------
77  void SystemInit (void)
78  {
79
80  #if defined (__VTOR_PRESENT) && (__VTOR_PRESENT == 1U)
81    SCB->VTOR = (uint32_t) &__VECTOR_TABLE;
82  #endif
83
84  #if defined (__FPU_USED) && (__FPU_USED == 1U)
85    SCB->CPACR |= ((3U << 10U*2U) |           /* enable CP10 Full Access */
86                   (3U << 11U*2U)  );         /* enable CP11 Full Access */
87  #endif
88
89  #ifdef UNALIGNED_SUPPORT_DISABLE
90    SCB->CCR |= SCB_CCR_UNALIGN_TRP_Msk;
91  #endif
92
93  #if defined (__ARM_FEATURE_CMSE) && (__ARM_FEATURE_CMSE == 3U)
94    TZ_SAU_Setup();
95  #endif
96
97    SystemCoreClock = SYSTEM_CLOCK;
98
99  #if defined __EN_ICACHE
100   if (SCB->CLIDR & SCB_CLIDR_IC_Msk)
101     SCB_EnableICache();
102   //else
103     //_sys_exit("no ICache included");/*implement your __exit code to pass
104 #endif
105 #if defined __EN_DCACHE
106   if (SCB->CLIDR & SCB_CLIDR_IC_Msk)
107     SCB_EnableDCache();
108   //else
109     //_sys_exit("no DCache included");
110 #endif
111
112
113 }
```

Note:

For demonstration, this example project disables I-cache/D-cache in the `main()` function in Round 2 by calling `SCB_DisableICache()` and `SCB_DisableDCache()`. However, it is strongly recommended that you enable/disable the caches when initializing the system before entering the `main()` function.

Based on this project, you can now start to use the caches and try more CMO functions in your STAR-based application software.

Copyright © 2021 Arm China. All rights reserved.
Confidentiality: Non-Confidential