

## 1 Introduction

This document introduces the dual-core application on i.MX RT1170, including the following topics:

- [Boot](#)
- [Brief introduction to MU/SEMA4/RDC/XRDC2/Shared memory](#)
- [Dual-core debug in different tool chain](#)

## 2 Boot

### 2.1 Boot from CM7

By default, i.MX RT1170 boots from CM7 core. This chapter discusses how to boot CM4 when CM7 core is the boot core.

#### 2.1.1 Basic model for understanding how to boot the second core

At first, we need to know a basic model to understand how to boot the second core, as shown in [Figure 1](#).



[Figure 1](#) shows the basic steps to boot the second core (CM4):

1. CM7 loads the image into CM4 ITCM.
2. CM7 kicks off CM4.
3. CM4 core is running.

On i.MX RT1170, CM4 ITCM has two address, `0x1FFE_0000` and the alias of `0x2020_0000`.

- `0x1FFE_0000` is for CM4 ITCM access. Only CM4 core can use this base address and the CM4 code image uses this address for instruction storage.
- `0x2020_0000` is for alias access with slow speed. Both CM7 and CM4 can access this base address. CM7 loads CM4 image to CM4 ITCM with this address.

#### 2.1.2 Detailed Boot flow

The detailed boot flow is as below:

1. Prepare CM4 image. Typically, it is placed inside CM7 image. It can also be from other source like SD card, or U-disk, and so on.
2. After CM7 core runs, CM7 core loads the CM4 image into target memory. Typically, for CM7, it is the space `0x2020_0000` which maps to CM4 ITCM.

#### Contents

1	Introduction.....	1
2	Boot.....	1
3	Brief introduction to MU/SEMA4/RDC/XRDC2/Shared memory .....	2
4	Dual-core debug in different tool chain.....	5
5	References.....	11
6	Revision history.....	11



3. CM7 core sets the vector table address of CM4 by `IOMUXC_LPSR_GPR -> GPR0` and `IOMUXC_LPSR_GPR -> GPR1`.
4. Kick off CM4 core by `SRC_SCR_BT_RELEASE_M4_MASK`.

### 2.1.3 Loading CM4 image into target memory

To load the CM4 image into target memory, use either way as below:

- Load explicitly, using the `memcpy()` function.
- Load implicitly, similar to initializing some variables in RAM.

For different IDE/Tool chain, SDK uses different way. It is controlled by macro `CORE1_IMAGE_COPY_TO_RAM`. MCUXpresso uses implicit way and IAR, KEIL and GCC uses explicit way.

For more details about boot from CM7, refer to SDK example at *SDK root/boards/evkmimxrt1170/multicore\_examples/hello\_world*

## 2.2 Boot from CM4

By default, i.MX RT1170 boot from CM7. By fusing `BT_CORE_SEL` (Bit 12 in `0x960`), the chip switches to CM4 as main core.

The steps to boot from CM4 are as below, similar to booting from CM7.

1. CM4 loads CM7 image into target memory.
2. Release CM7 core.
3. CM7 core is running.

The difference is that CM7 vector table is loaded from `IOMUXC_LPSR_GPR -> GPR26` and the release from CM7 is by `SRC_SCR_BT_RELEASE_M7_MASK`. For more details, see the `mcmgr_start_core_internal()` function in *mcmgr\_internal\_core\_api\_imxrt1170.c*.

#### NOTE

CM4 cannot access CM7 TCM directly, so if necessary, use eDMA.

As NXP official SDK package only supports CM7 as main core (boot core), the enablement for this case is limited. Do not switch to CM4 as main core if it is not a must in application.

## 3 Brief introduction to MU/SEMA4/RDC/XRDC2/Shared memory

### 3.1 MU

By MU, one core can give a 32-bit message to the other, and trigger interrupt of the other immediately. It supports four bi-directional channels. [Figure 2](#) shows the MU structure.

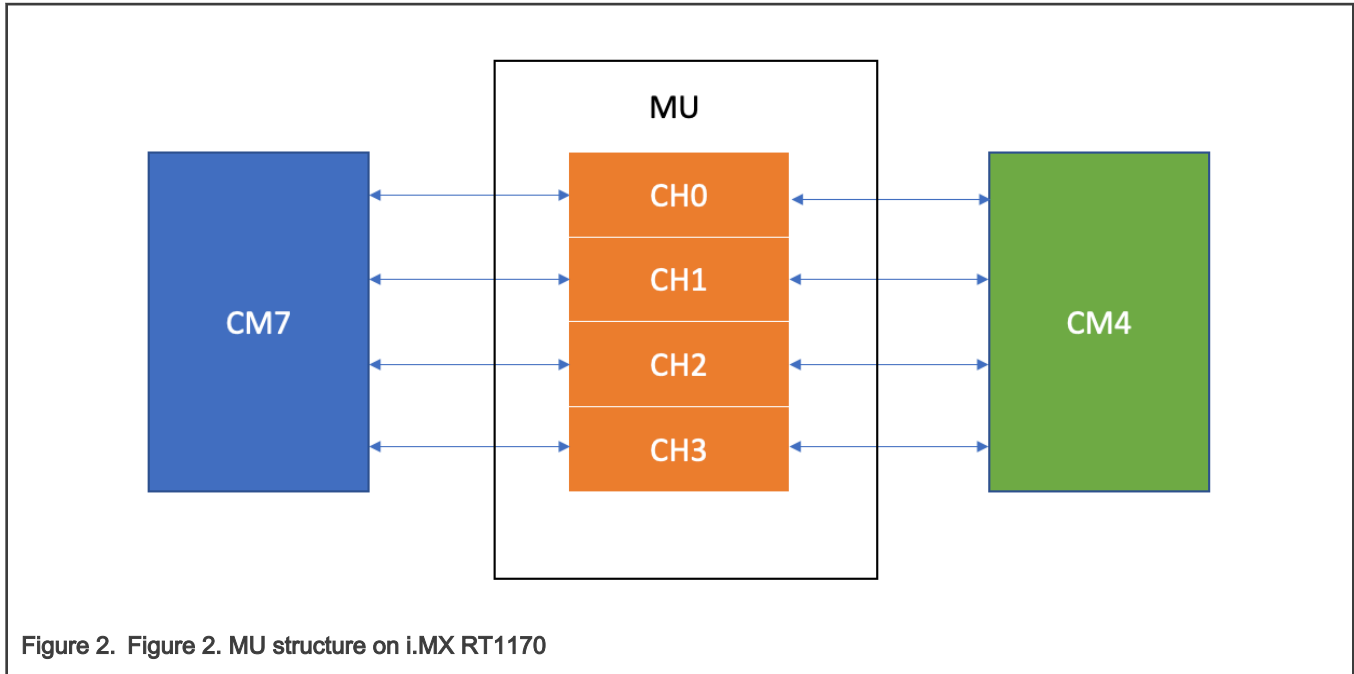


Figure 2. Figure 2. MU structure on i.MX RT1170

For example, CM7 can give a `0x1234_5678` message to CM4, and then CM4 can catch the `0x1234_5678` message in an interrupt immediately.

For related SDK example, see *SDK root/boards/evkmimxr1170/driver\_examples/mu*.

### 3.2 SEMA4

SEMA4 is typically used for resource protection in multi-core environment, to prevent different masters using one resource at the same time.

The resource can be a memory block, a peripheral or even an SW object in memory.

The SEMA4 supports up to 16 gates. Only the master who locks the gate successfully, getting the sema4, can use the resource protected by this SEMA4 gate. Other masters have to wait until the master who owns the gate releases/unlocks this gate.

Figure 3 shows the SEMA4 structure.

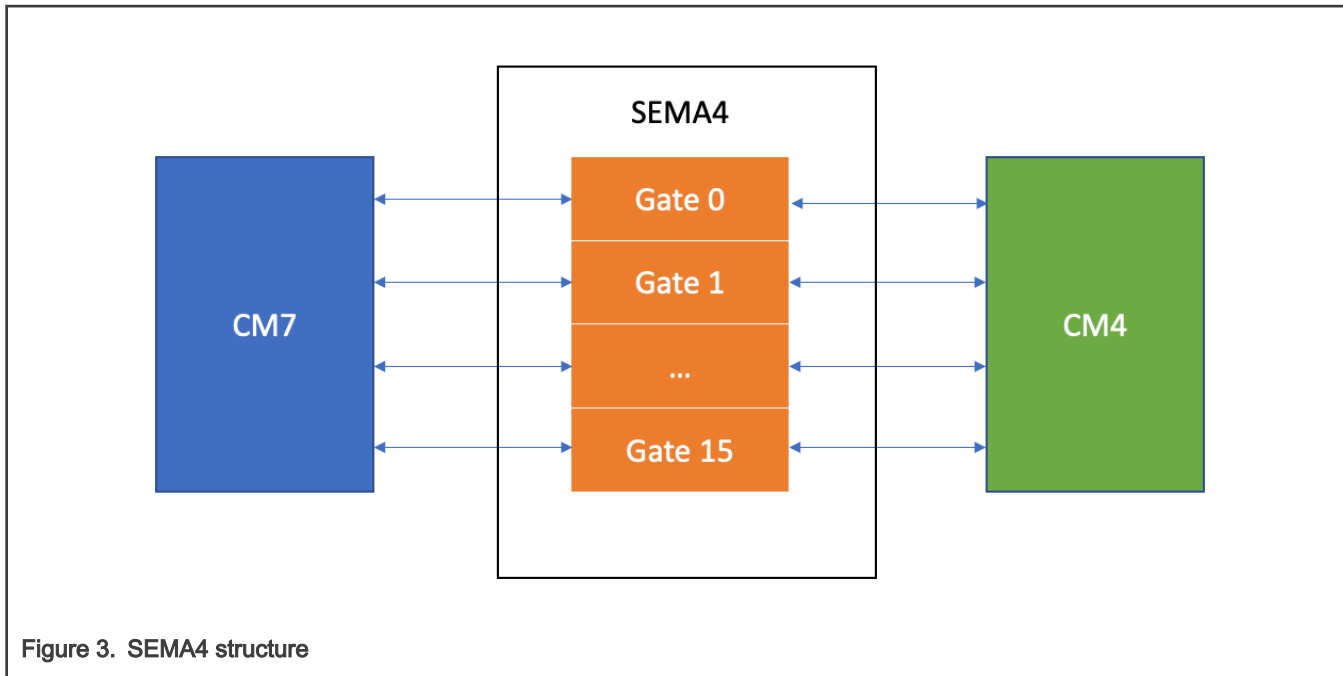


Figure 3. SEMA4 structure

For related SDK example, see *SDK root/boards/evkmimxrt1170/driver\_examples*.

### 3.3 RDC/XRDC2

RDC/XRDC2 is used for resource isolation. The resource can be peripheral or memory. The privilege can be read, written or both read and written.

For example, we can assign LPUART1 to CM7 domain, and then only CM7 core can access LPUART1. If other bus master like CM4 core accesses LPUART1, violation ISR happens.

**NOTE**

Bus masters for RDC/XRDC2 are not limited to CM7 and CM4 core, ENET, LCDIF, USB, CSI and so on, are also bus masters.

### 3.4 Shared memory

Shared memory can be accessed by both CM7 and CM4 core. If a data block needs to be passed from one core to the other core, shared memory with MU can be a good candidate.

The typical working flow is as below:

1. A core write a block data into B core.
2. After [Step 1](#) completes, A core trigger B core MU interrupt, and B core knows there is data block from A core ready.
3. B core read the data block and process.
4. After [Step 3](#) completes, B core trigger A core MU interrupt, and A core knows B core has processed the data block, and then A core can load the next data block for B core.

If necessary, double buffer and even a queue can be used.

[Table 1](#) lists typical memory which can be used as shared memory.

Table 1. Typical memory which can be used for shared memory

	Access address for CM7	Access address for CM4	Size
CM4 DTCM	0x2022_0000	0x2000_0000	128 KB
OCRAM1	0x2024_0000	Same as CM7	512 KB
OCRAM2	0x202C_0000	Same as CM7	512 KB
OCRAM1_ECC	0x2034_0000	Same as CM7	64 KB
OCRAM2_ECC	0x2035_0000	Same as CM7	64 KB

In addition, external memory from SEMC, like SDRAM and memory from Flex SPI, can also be used as shared memory if necessary.

## 4 Dual-core debug in different tool chain

For the guide to dual-core debugging, see *Getting Started with MCUXpresso SDK for MIMXRT1170-EVK* (document MCUXSDKMIMXRT117XGSUG), which is available on [MCUXpresso SDK Builder](#), as shown in Figure 4.

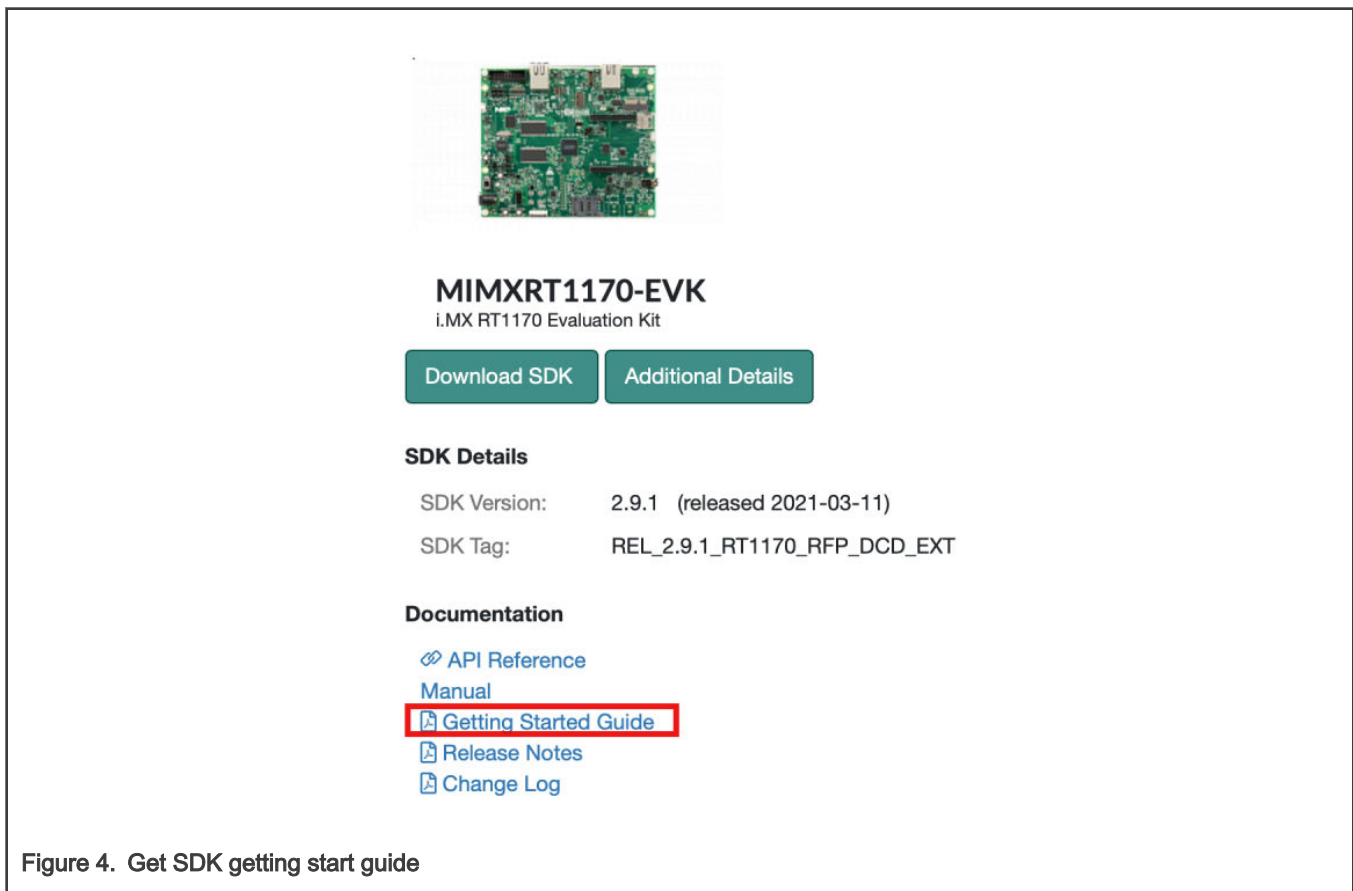


Figure 4. Get SDK getting start guide

See the sections related to multi-core to know how to start a dual-core debugging on RT1170 for *MCUXpresso/IAR/Keil/GCC*. This document also introduces some key points for dual core debugging for user reference.

## 4.1 MCUXpresso

Test environment:

- SDK: 2.9.1 for i.MX RT1170
- MCUXpresso: 11.3.1
- Example: *SDK\_root\boards\evkmimxrt1170\multicore\_examples\hello\_world*

### 4.1.1 By DAP-LINK (CMSIS DAP)

Set the board to XIP boot mode by setting **SW1** to **OFF OFF ON OFF**. Make sure there is a known image in the flash which is working correctly. Some bad images will block debugger connections.

Quick steps for MCUXpresso dual core debug:

1. Import multi-core hello world example, as shown in [Figure 5](#).

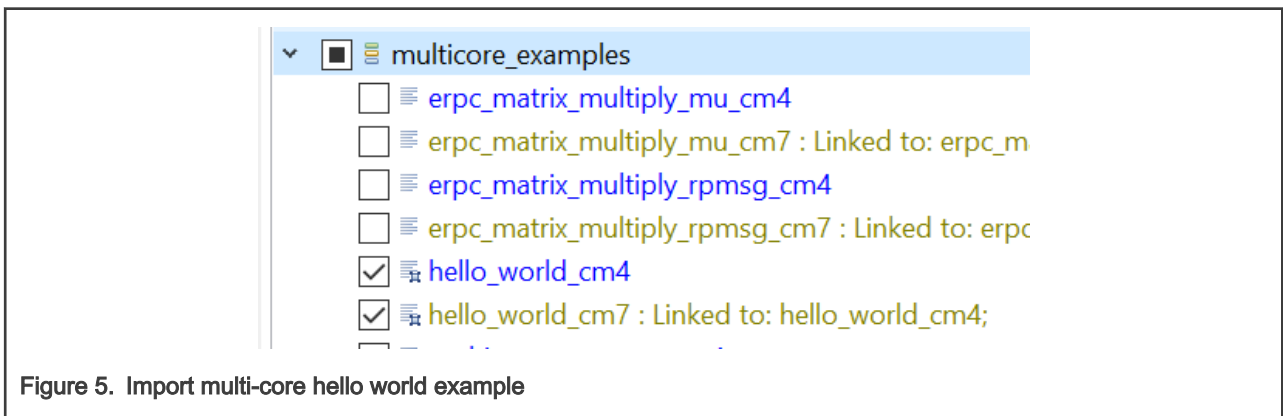


Figure 5. Import multi-core hello world example

2. Build the CM4 project.
3. Build the CM7 project.
4. Select the CM7 project.

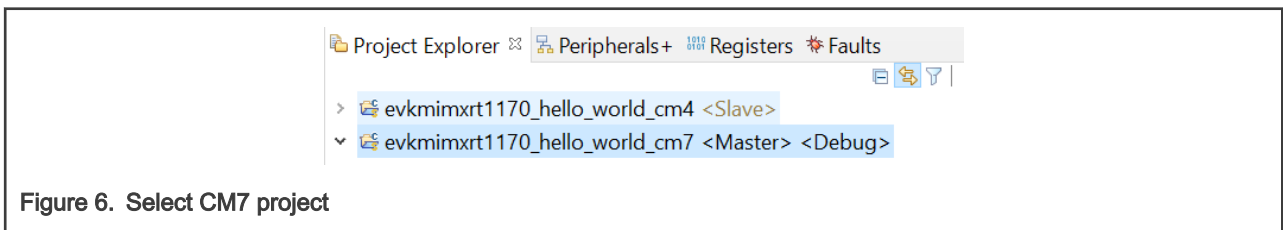






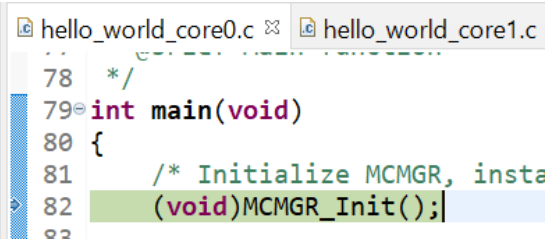
Figure 6. Select CM7 project

5. Start debug session by click the  button.
6. Select CMSIS DAP debugger for CM7 and CM4 separately in [Figure 7](#).

Available attached probes					
	Name	Serial number/ID	Type	Manufacturer	IDE Debug Mode
	CMSIS-DAP v1	02440b03092db67700000...	LinkServer	ARM	Non-Stop
	J-Link PLUS	600109557	USB	SEGGER	All-Stop
	3rd party CMSIS-DAP	1043579317	USB	SEGGER	All-Stop

**Figure 7. Select CMSIS DAP**

7. Then CM7 stops at main.



```

78  */
79  int main(void)
80  {
81      /* Initialize MCMGR, insta
82      (void)MCMGR_Init();
  
```

**Figure 8. CM7 core stops at main()**

8. Click the



button.

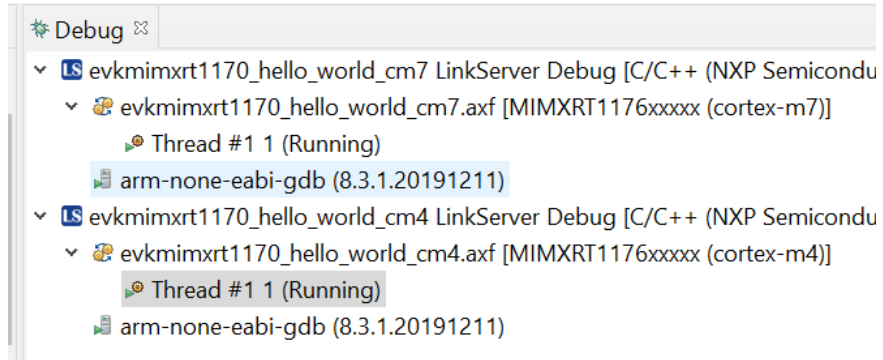
9. CM4 core stops at main.

10. Click the **Run** button in CM4 core project and both cores are running. The console log is as below:

```

Hello World from the Primary Core!
Starting Secondary core.
The secondary core application has been started.
  
```

11. In the **Debug** window, both cores are running. We can select either core and stop it for debugging, or if there is user breakpoint, it hits once the code runs to the breakpoint.



**Figure 9. Debug window**

### 4.1.2 By JLink

Set the board to XIP boot mode by setting SW1 to OFF OFF ON OFF.

For JLink debugger, un-populate J5-J8. Steps for CMSIS DAP are same, and the below only lists the different steps:


1. Select CM7 project and start debug session.
2. Run CM7 project, while CM4 project will not be attached automatically.
3. Attach CM4 project.
  - a. Select CM4 project.
  - b. Click the  button.
4. The dual-core debug is enabled.



Figure 10. Dual-core debug by JLink in MCUXpresso


## 4.2 IAR

Test environment:

- SDK: 2.9.1 for i.MX RT1170
- IAR: 9.10.1
- Example: `SDK_root\boards\evkmimxrt1170\multicore_examples\hello_world`

### 4.2.1 By DAP-LINK (CMSIS DAP)

Quick steps for IAR dual core debug are as below:

1. Build the CM4 project.
2. Build the CM7 project.
3. In the CM7 project, start the debug session by click the  button.

Then both CM7 and CM4 projects are launched for debugging automatically by IAR in multicore mode.



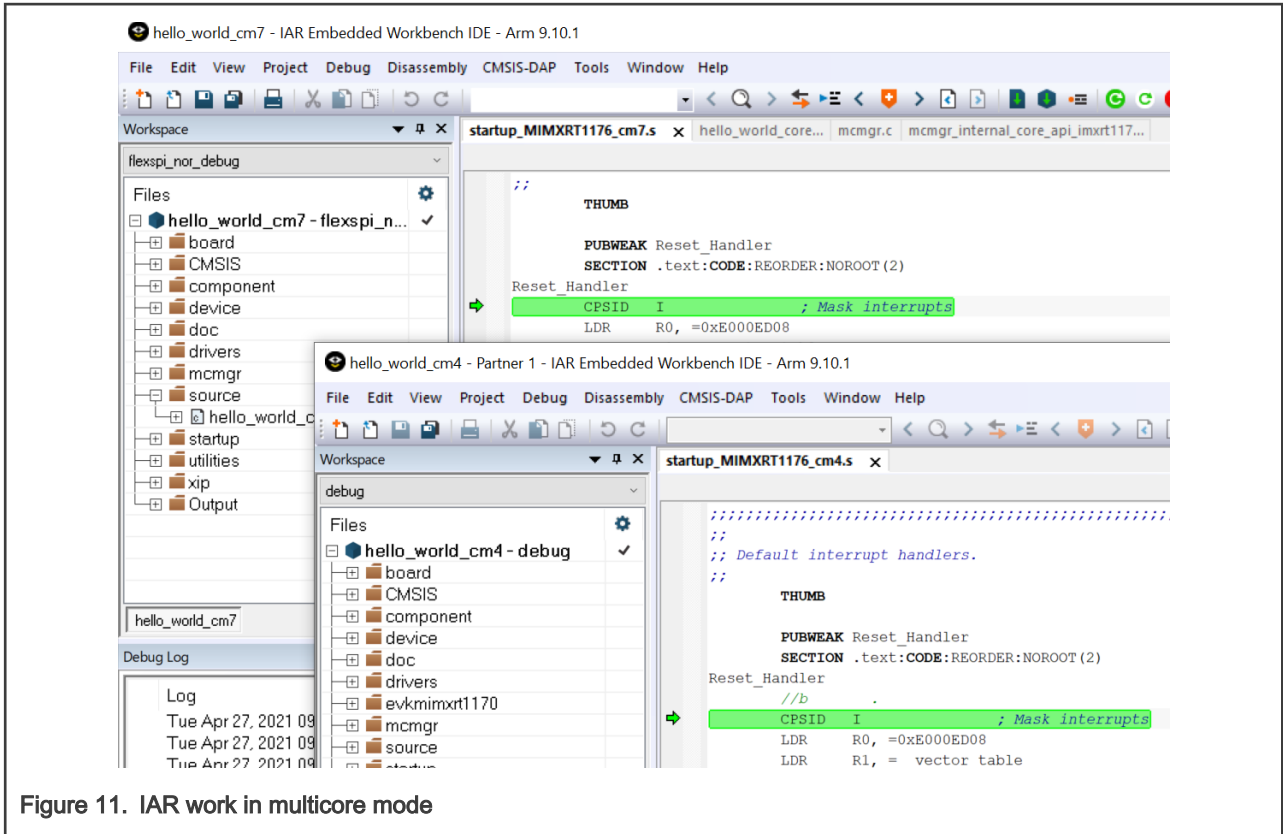


Figure 11. IAR work in multicore mode

4. Run CM7 core in the CM7 project.
5. Run CM4 core in the CM4 project.

### 4.2.2 By JLink

Set the board to XIP boot mode by setting SW1 to OFF OFF ON OFF.

For JLink, IAR does not support multicore mode like what it supports for DAP-Link. We need to open CM7 project and CM4 project separately for dual-core debugging.

The steps for dual core debug by JLink are as below:

1. Build CM4 and CM7 projects.
2. Start CM7 debug session.
3. Run the CM7 project.
4. Attach CM4 project, as shown in [Figure 12](#).

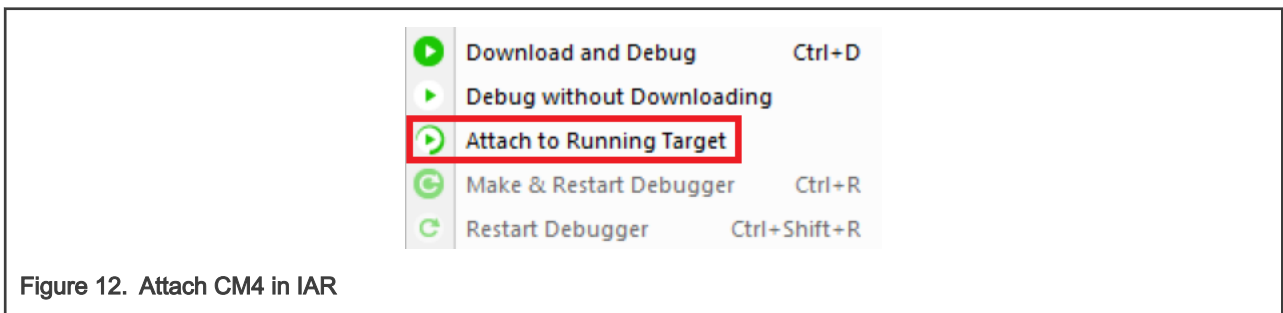



Figure 12. Attach CM4 in IAR

5. Press the  button if necessary, and the code stops where it is running.

### 4.2.3 Debug CM4 core from the first instruction

By default, CM4 core debugging cannot start from the beginning. To debug CM4 from the first instruction, follow steps as below.

1. Add an loop instruction as shown in [Figure 13](#).

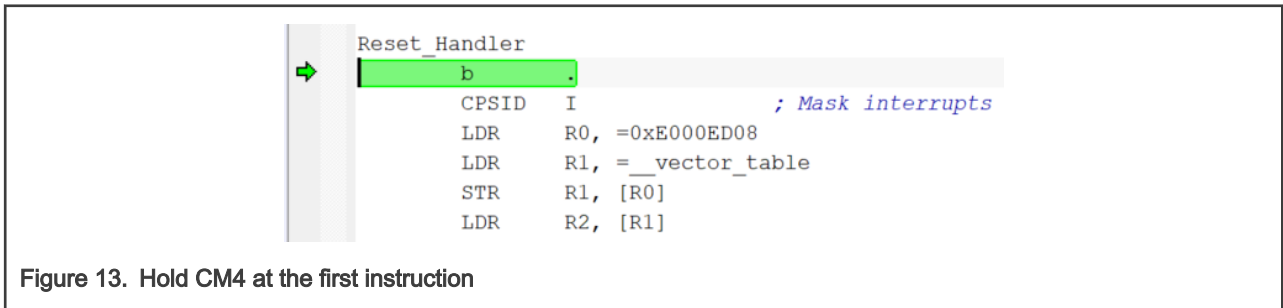


Figure 13. Hold CM4 at the first instruction

2. When CM4 core project is loaded correctly, the code breaks here. Move PC to the next instruction by right-clicking next instruction → **Set next statement** command.

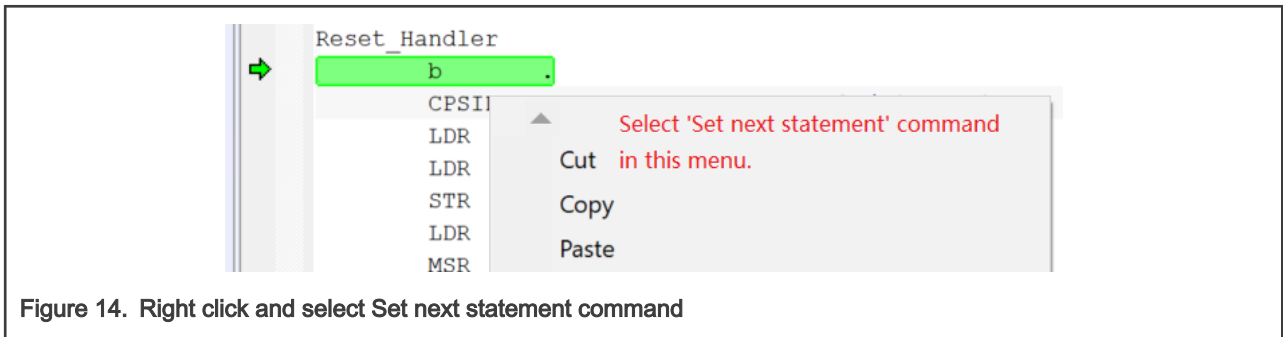


Figure 14. Right click and select Set next statement command

Now, we can debug from the first application instruction.

## 4.3 Keil

Test environment:


- SDK: 2.9.1 for i.MX RT1170
- Keil: 5.34
- Example: *SDK\_root\boards\levkmimxrt1170\multicore\_examples\hello\_world*

### 4.3.1 By DAP-LINK(CMSIS DAP)

Set the board to XIP boot mode.

Follow the steps as below for dual-core debugging:

1. Build the CM4 project.
2. Build the CM7 project.
3. Start the CM7 debug session.

4. Attach the CM4 project by clicking the  button.

### 4.3.2 By Jlink

Set the board to XIP boot mode by setting **SW1** to **OFF OFF ON OFF**.

Follow the steps below for dual-core debugging:

1. Copy `evkmimxrt1170\multicore_examples\hello_world\cm7\evkmimxrt1170_connect_cm4_cm7side.jlinkscript` to `mdk` folder and rename it to **JLinkSettings.jlinkscript**.
2. Copy `evkmimxrt1170\multicore_examples\hello_world\cm4\evkmimxrt1170_connect_cm4_cm4side.jlinkscript` to `mdk` folder and rename it to **JLinkSettings.jlinkscript**.
3. Build the CM4 project.
4. Build the CM7 project.
5. Start the CM7 debug session but do not run.
6. Start the CM4 debug session.
7. Run the CM7 project.

### 4.3.3 Debug CM4 core from the first instruction.

For details, see [Debug CM4 core from the first instruction](#) and one difference is to select **Set Program Counter** after right click.

## 5 References

- *i.MX RT1170 Processor Reference Manual* (document [IMXRT1170RM](#))
- *Getting Started with MCUXpresso SDK for MIMXRT1170-EVK* (document [MCUXSDKMIMXRT117XGSUG](#))

## 6 Revision history

Revision number	Date	Substantive changes
0	27 April, 2021	Initial release

## How To Reach Us

### Home Page:

[nxp.com](http://nxp.com)

### Web Support:

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

**Right to make changes** - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Security** — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at [PSIRT@nxp.com](mailto:PSIRT@nxp.com)) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2021.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: 27 April, 2021

Document identifier: AN13264

