AN12123

LPC541xx dual core starting guide

Rev. 1.0 — 7 March 2018 Application note

Document information

Info	Content
Keywords	LPC541xx, LPC54102, LPC54114, dual core
Abstract	This application note describes the dual core feature of the LPC541xx MCU series, and the method to develop applications based on the features from the user perspective.

Revision history

Rev	Date	Description
1.0	20180307	Initial version

Contact information

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

1. Introduction

LPC541xx is a mainstream and power efficient MCU sub-series within the LPC Cortex-M microcontrollers, integrating dual core capability. Both LPC54102 and LPC54114 have this feature. This application note focuses on LPC54114 as an example. However, there are differences in the dual core implementation for these two devices that are explained in this application note.

LPC541xx adopts an asymmetric dual core mechanism by integrating Cortex-M4F and Cortex-M0+ within the same die. For detailed specification of these cores, see the following documentations from ARM:

Cortex ‑M4 Processor Technical Reference Manual

Cortex-M4 Devices Generic User Guide

Cortex-M0+ Technical Reference Manual

Cortex-M0+ Devices Generic User Guide

This application note starts with a brief introduction of LPC541xx dual core features and implementation, followed by details of the development process. MCUXpresso IDE and MCUXpresso SDK are used to illustrate the steps.

2. Features and implementation

2.1 Features

The LPC541xx dual processor core features are:

- ARM® Cortex®-M4 CPU
  - ARM Cortex-M4 processor, running at a frequency of up to 100 MHz
  - Floating Point Unit (FPU) and Memory Protection Unit (MPU)
  - ARM Cortex-M4 built-in Nested Vectored Interrupt Controller (NVIC)
  - Non-maskable Interrupt (NMI) with a selection of sources
  - Serial Wire Debug (SWD) with eight breakpoints and four watchpoints; includes serial wire output for enhanced debug capabilities.
  - System tick timer
- ARM® Cortex®-M0+ CPU
  - ARM Cortex-M0+ processor, running at a frequency of up to 100 MHz (using the same clock as the Cortex-M4).
  - ARM Cortex-M0+ built-in Nested Vectored Interrupt Controller (NVIC)
  - Non-Maskable Interrupt (NMI) with a selection of sources
  - Serial Wire Debug (SWD) with four breakpoints and two watchpoints.
  - System tick timer

2.2 Implementation

See Fig 1 for the block diagram of LPC5410x.

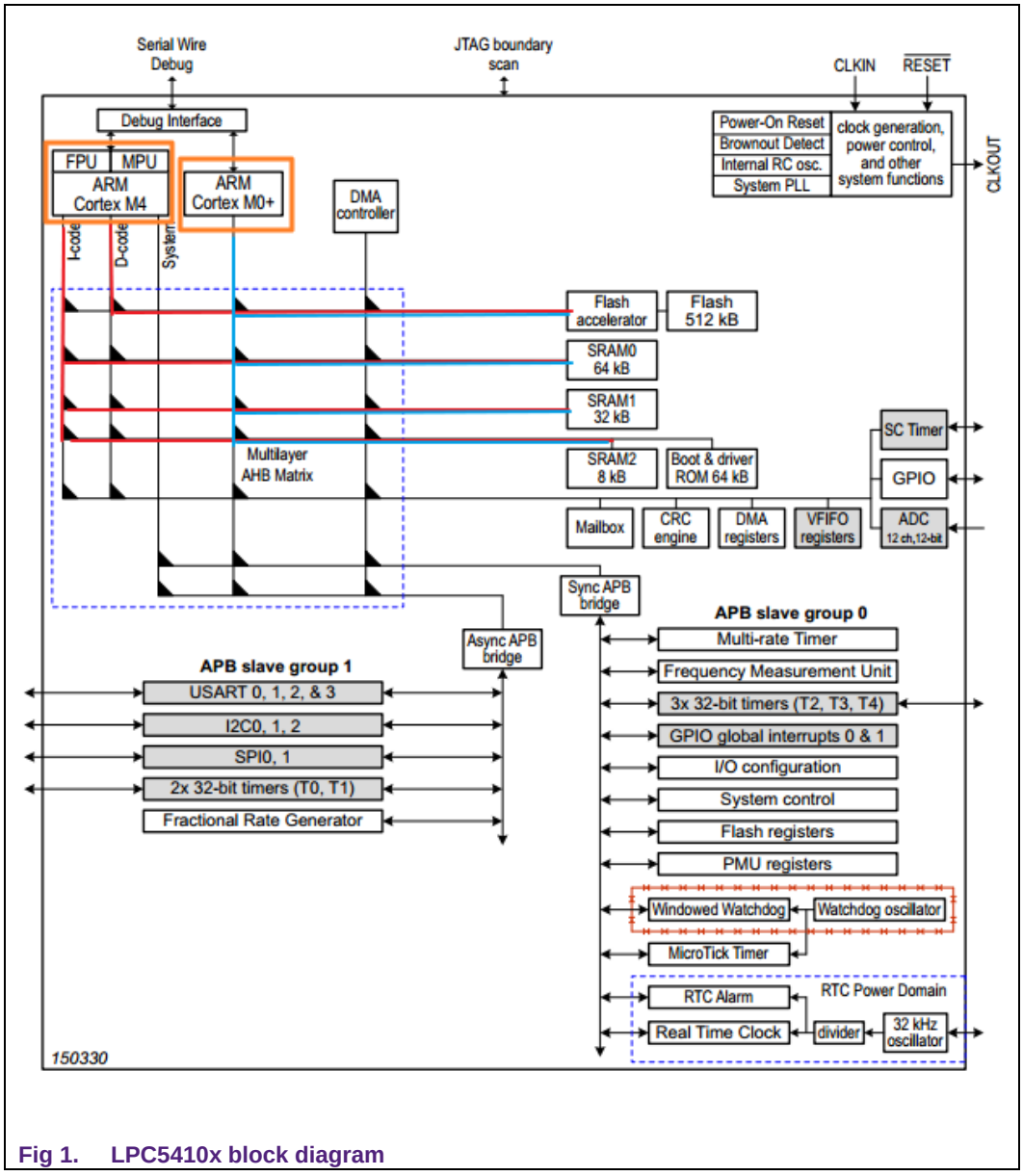

Fig 1. LPC5410x block diagram

See Fig 2 for the block diagram of LPC5411x.

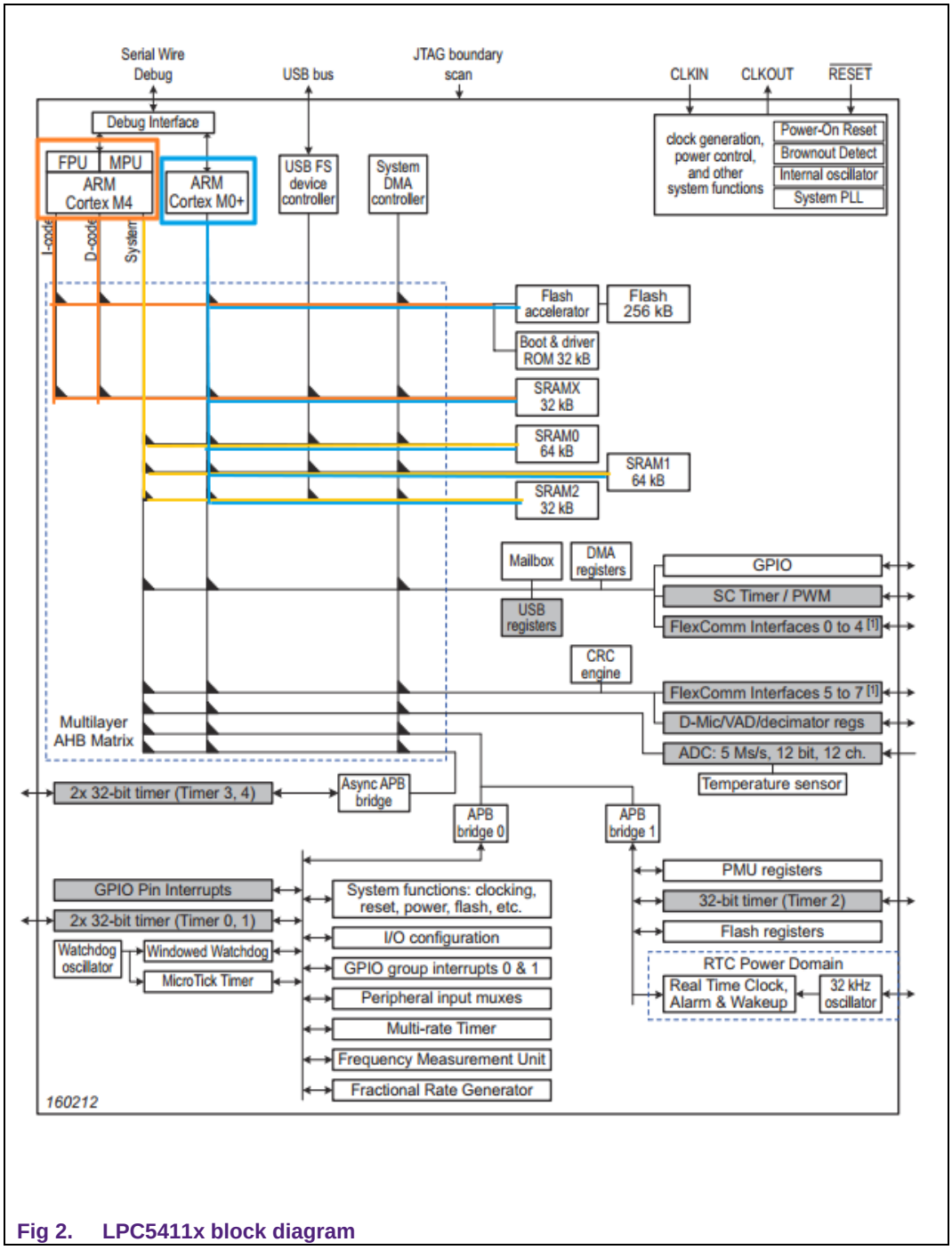

Fig 2. LPC5411x block diagram

In LPC5410x and LPC5411x, both cores reside in the MCU AHB bus as master, whose access priority can be configured depending on the user application. For the details of the AHB master priority setting, see LPC5410x or LPC5411x user manual (Chapter 4: System configuration).

Both cores run on the same clock up to 100 MHz and all the AHB/APB slaves such as memory, timers, and other peripherals are available for them to access or control equally. There are multiple SRAM banks available within the MCU that can be powered ON and OFF individually for power saving and the two cores can access different SRAM banks simultaneously without contention. However, there is only one flash memory block, so one of the cores need to run its code in the SRAM.

The highlighted connections between the cores and memory in Fig 1 and Fig 2 show that there are different memory implementations for the two parts. For LPC5410x, all the memories (flash, and SRAM) are connected to its I-code and D-code bus. For LPC5411x, only flash and SRAMX are connected to I-code and D-code bus. The rest of the memories are connected to the system bus. Different bus connections on memory may result to different performances. So, resource should be carefully allocated; see Section 3.

Cortex-M4 contains three external AHB bus interfaces:

- I-code memory interface for instruction fetch
- D-code memory interface for data and debug access
- System interface for instruction fetch, data and debug access

Cortex-M0+ has only one AHB bus interface, which connects to all AHB/APB slaves. For detailed bus interface definition, see: Cortex -M4 Processor Technical Reference Manual and Cortex-M0+ Technical Reference Manual.

This dual core architecture is asymmetric. The roles of the two cores are different within the implementation. One core is the master and the other is the slave. Cortex-M4 acts as the master CPU by default when the power is ON. Master CPU can disable or reset the slave core, but not vice versa. Only master CPU can call the power APIs to cause the MCU enter into low-power mode. Boot address registers and initial stack pointer address registers are available for the slave CPU.There are boot address and initial stack pointer address registers available for the slave CPU. Application can use these registers to setup the appropriate boot and stack address for the slave CPU, which is different from the master CPU. For the register description details, see LPC5410x or LPC5411x user manual (System configuration of Chapter 4: Dual-CPU related registers).

Inter-core communication is supported by mailbox, allowing interrupt generation between the two cores. Hardware mutex is also available for shared resource access, to avoid contention. For the mailbox and mutex description details, see LPC5410x user manual (Chapter 27: Mailbox) or LPC5411x user manual (Chapter 30: Inter-CPU mailbox).

3. Design consideration

A dual core application should be developed with an aim to utilize its features effectively. The application can be developed to provide better performance, high power efficiency, reasonable labor division, or any customized requirement from the user.

3.1 Application task division

Cortex-M4 and Cortex-M0+ have different features and instruction sets, which enable them for different applications and tasks. See Fig 3 for comparison diagram on instruction set and architecture between Cortex-M serial cores.

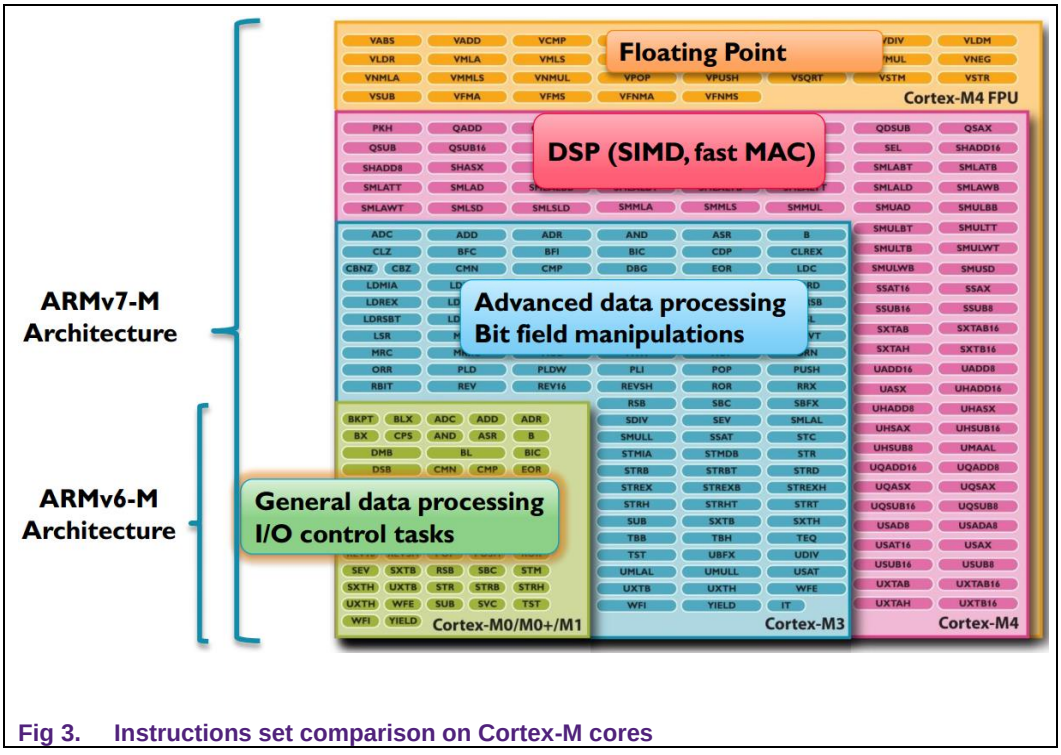

Fig 3. Instructions set comparison on Cortex-M cores

Cortex-M4 instructions are downward compatible with Cortex-M0+, and with enhancements on advanced data processing and DSP instructions. If FPU is integrated, as in the case of LPC541xx MCU, floating algorithm process can be handled efficiently by Cortex-M4. Because of the rich instruction set and Harvard bus architecture, high performance algorithm and data processing can be achieved with Cortex-M4.

Cortex-M0+ is based on ARMv6-M architecture which integrates basic data processing instructions. With its short pipeline, it is suitable for simple I/O control tasks.

3.2 Resource allocation

The main principle for resource allocation of the two cores is ensuring high performance parallel operation without contention. Memory allocation mainly depends on the task

labor division. Since there is only one flash bank available in the LPC5411x MCUs, one of the cores should run its code and data in SRAM. The size of the SRAM is relatively small. The flash memory should be allocated to the task-intensive core which is in most of cases, Cortex-M4.

As mentioned in Chapter 2.2, LPC5411x has four SRAM banks. Only SRAMX is connected to the Cortex-M4 D-code and I-code bus while the other three SRAM banks (SRAM0, SRAM1, and SRAM2) are connected to the system bus. It is recommended that SRAMX is assigned to Cortex-M4 for data or code storage. Since the four SRAM banks are same from the perspective of Cortex-M0+, they are all connected to the system bus.

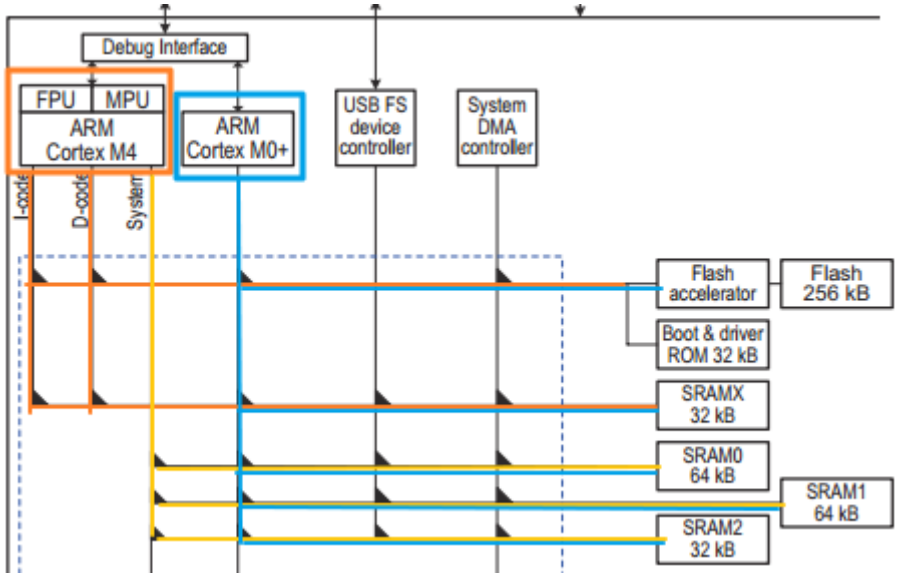

Fig 4. LPC5411x memory allocation

Resource access contention between the two cores should be avoided in a real application system using dual core feature. It is important for the application to allocate resources properly to the cores. In cases such as data sharing where the same memory area is accessed by both the cores, hardware mutex can be used to avoid race condition.

4. Development process

This application note uses LPC54114 and MCUXpresso SDK to illustrate the dual core development process. MCUXpresso IDE is a NXP MCU development and debugging tool, which has free edition available for user to download and use. MCUXpresso SDK is the software development package for NXP MCU serials, which includes various peripheral drivers, middleware such as USB stack, LWIP and FATFS. SDK can be a good start for user application development. MCUXpresso SDK can be downloaded from the link: MCUXpresso SDK
and MCUXPresso IDE can be downloaded from the link: MCUXpresso IDE.

4.1 Configure and download SDK

After opening the MCUXpress SDK web page, user should first register as *Guest* and log in.

1. Register and log in.

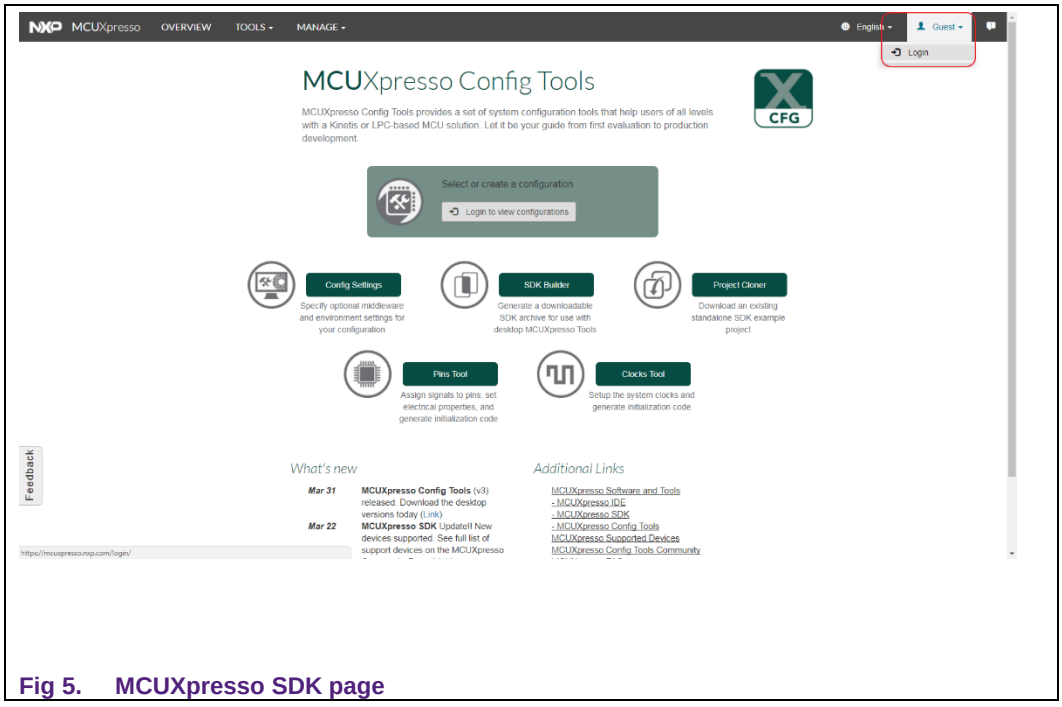

Fig 5. MCUXpresso SDK page

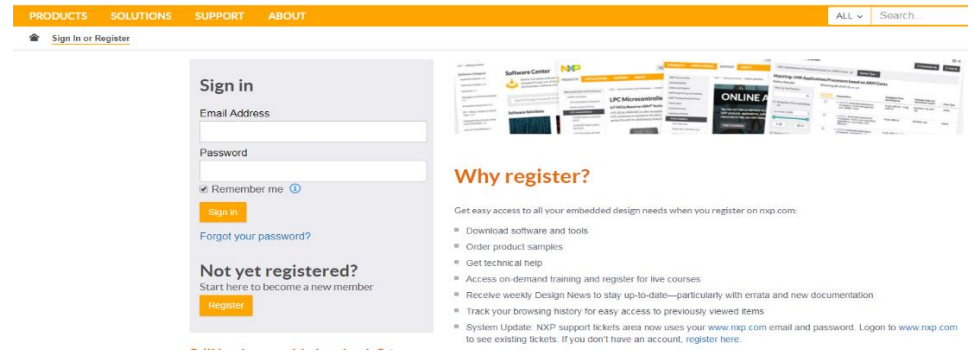

Fig 6. Register and log in page

2. Create a new SDK configuration for LPC54114.

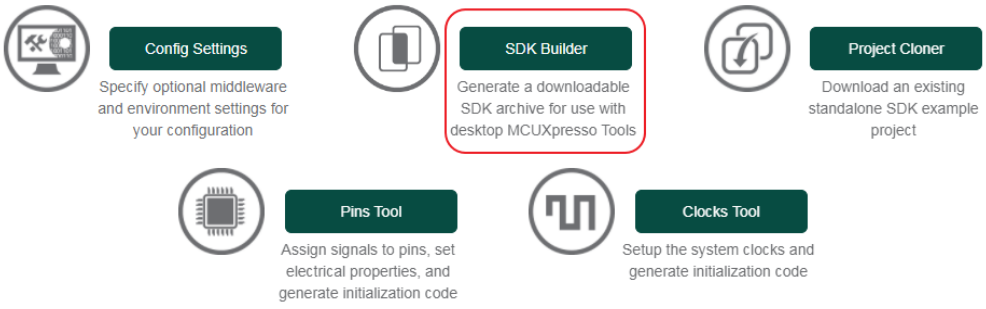

Fig 7. Register and log in page

Click *SDK Builder* to start a new SDK configuration.

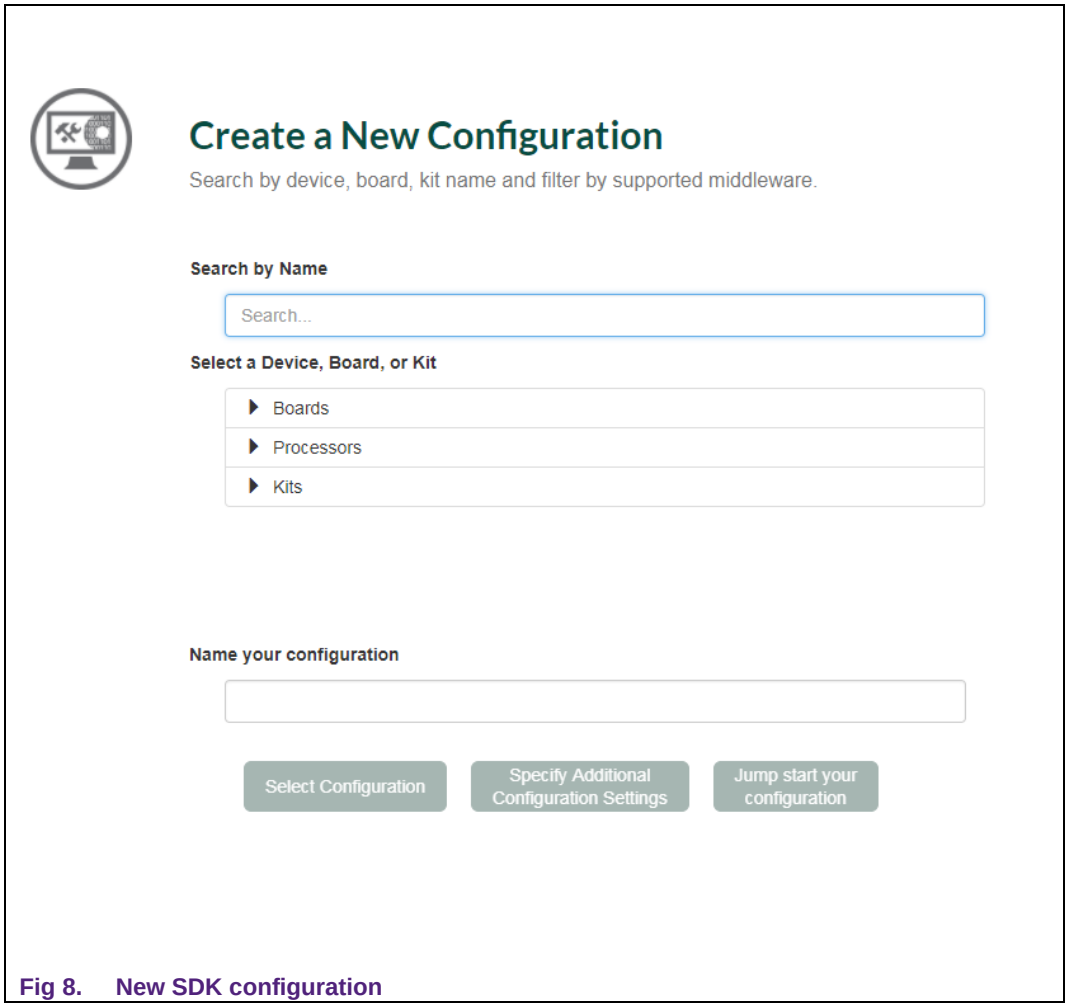

Fig 8. New SDK configuration

Choose LPCXpresso54114 for the new configuration.

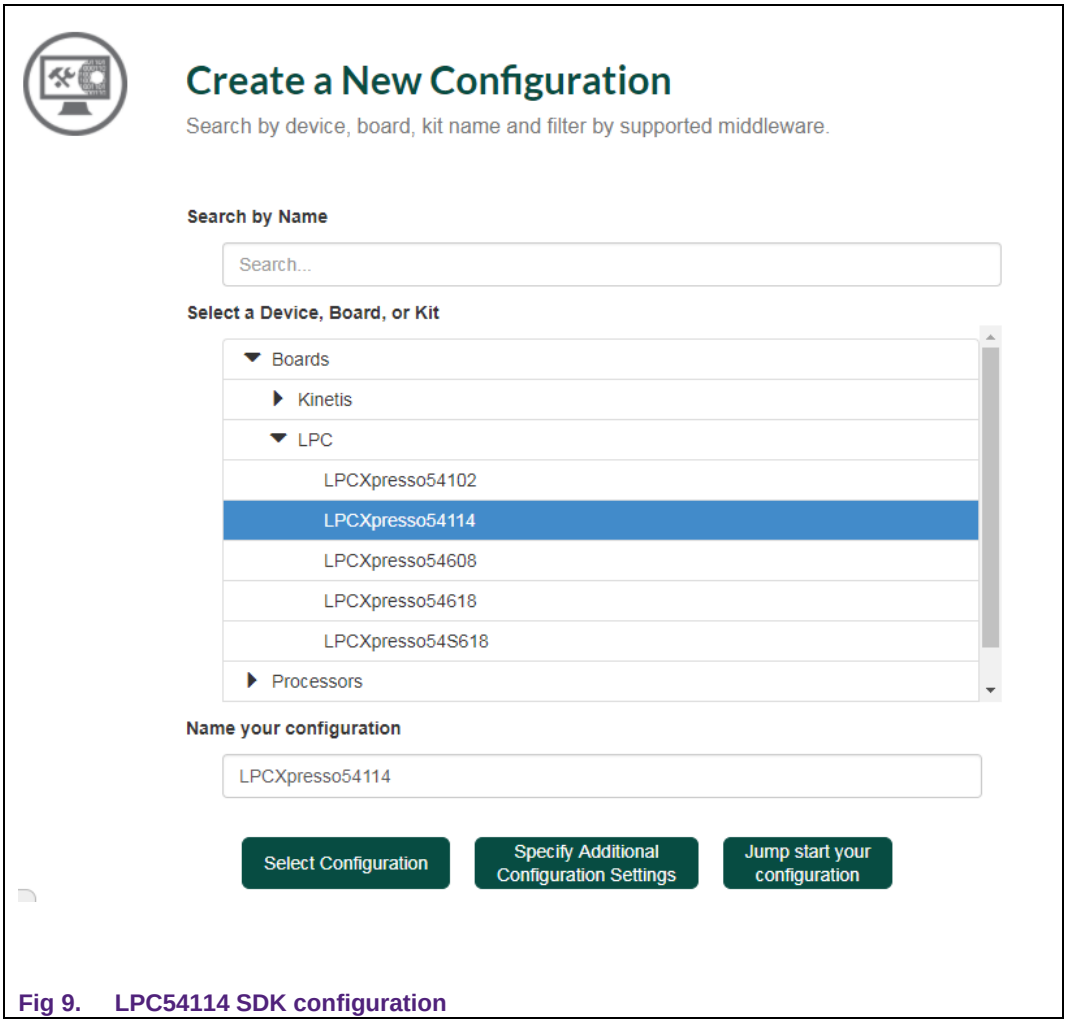

Fig 9. LPC54114 SDK configuration

Click *Specify Additional Configuration Settings* to specify further settings. Choose *MCUXpreso IDE* as the toolchain.

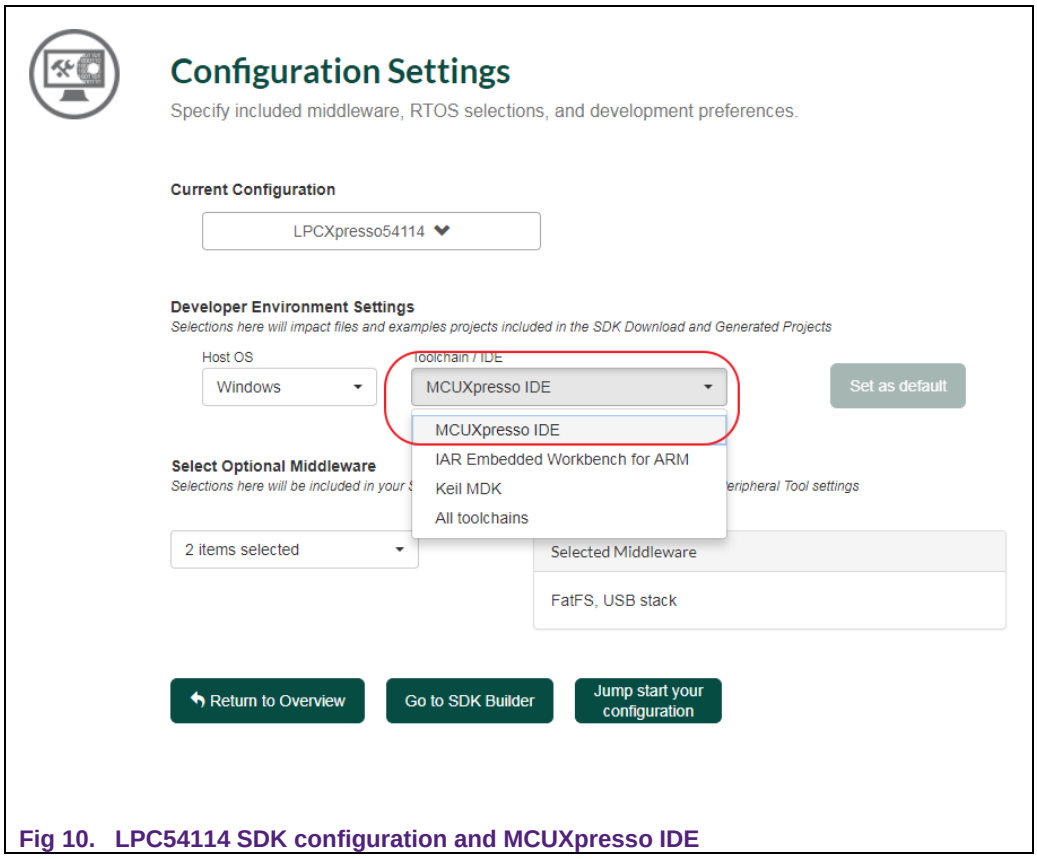

Fig 10. LPC54114 SDK configuration and MCUXpresso IDE

Click *Go to SDK Builder* to complete the configuration.

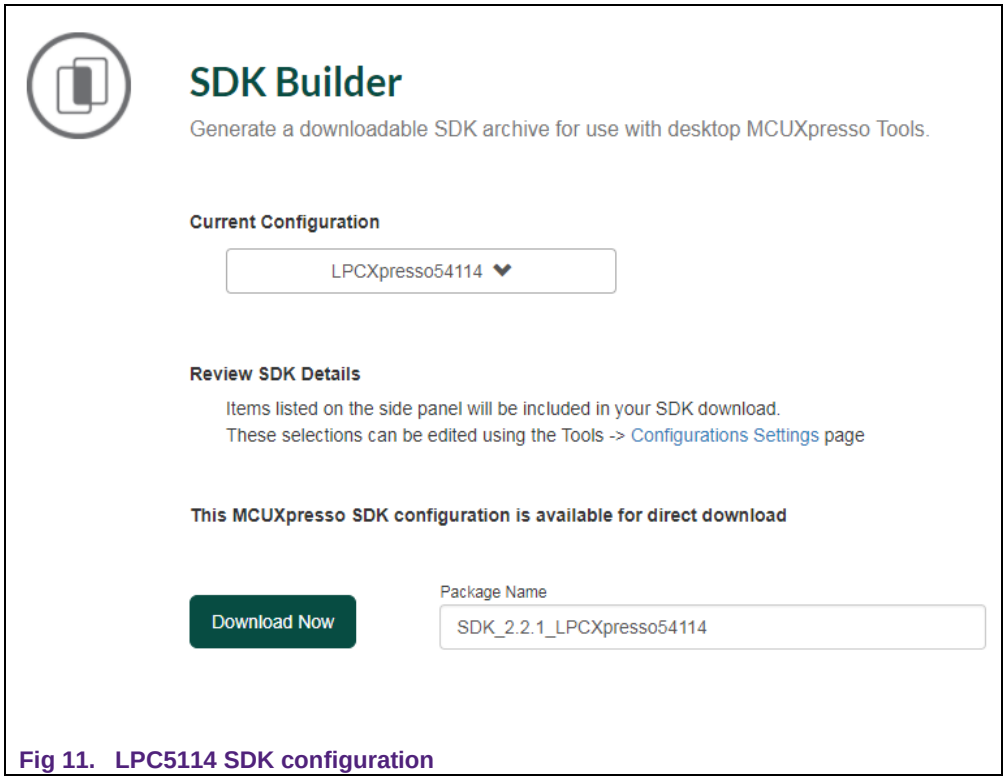

Fig 11. LPC5114 SDK configuration

3. Download LPC54114 SDK.

   Click *Download Now* and agree with the software terms and conditions. The LPC54114 SDK will start downloading.

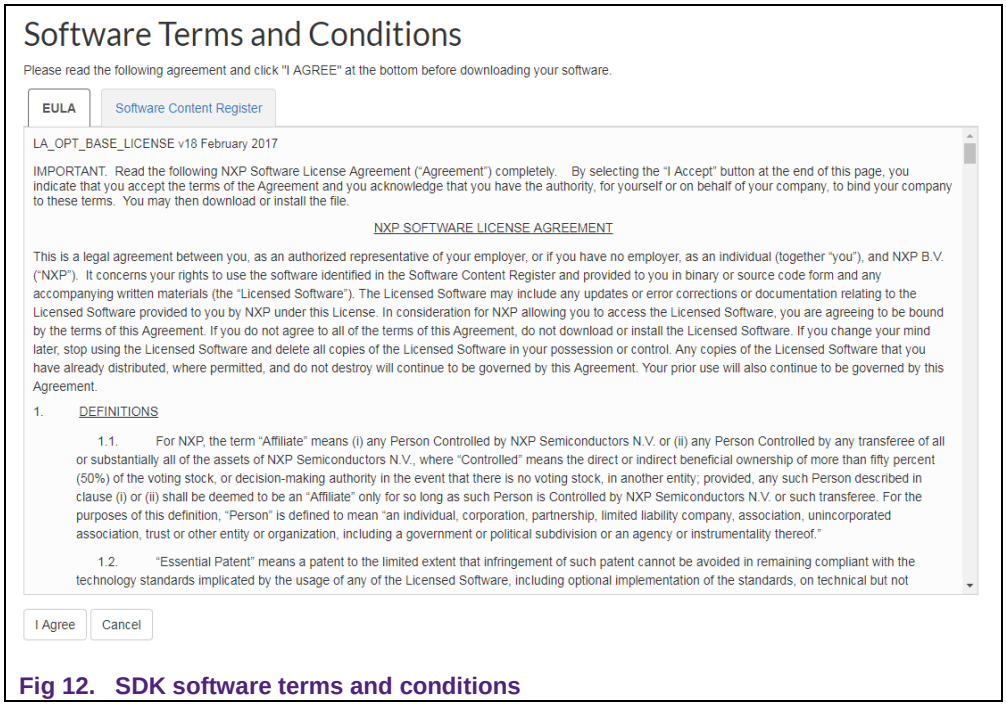

Fig 12. SDK software terms and conditions

4.2 Import SDK project

After the LPC54114 SDK is available, user can open the MCUXpresso IDE to import the dual core example project from SDK and start the development.

1. Install LPC54114 SDK

   Open MCUXpresso IDE and specify a directory for the workspace.

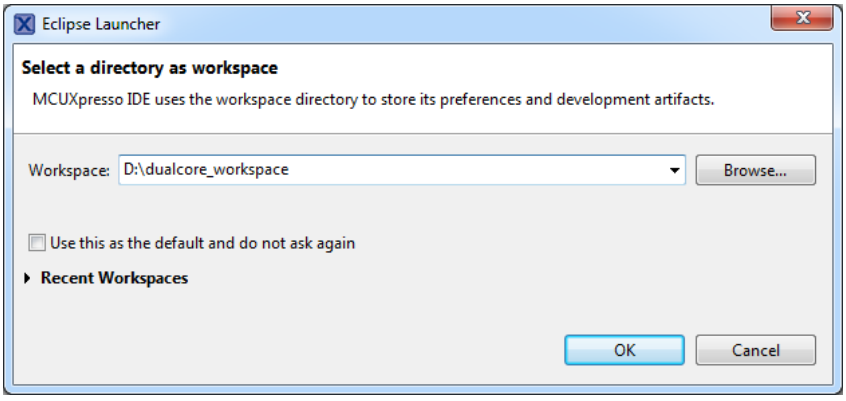

Fig 13. Specify workspace for MCUXpresso IDE project

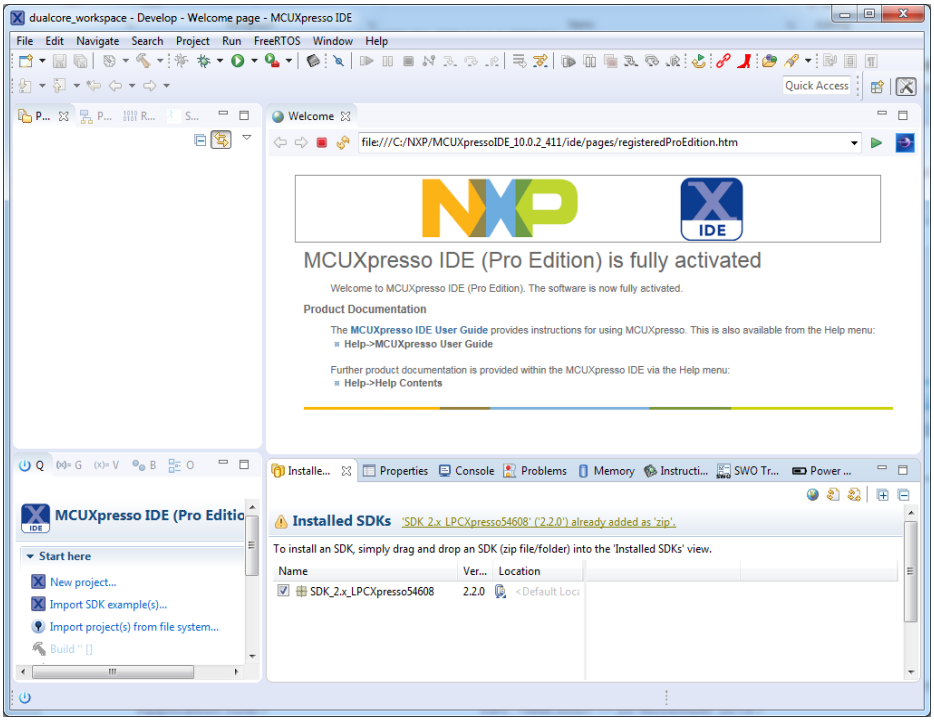

Fig 14. MCUXpresso IDE workspace

Drag and drop to install LPC54114 SDK.

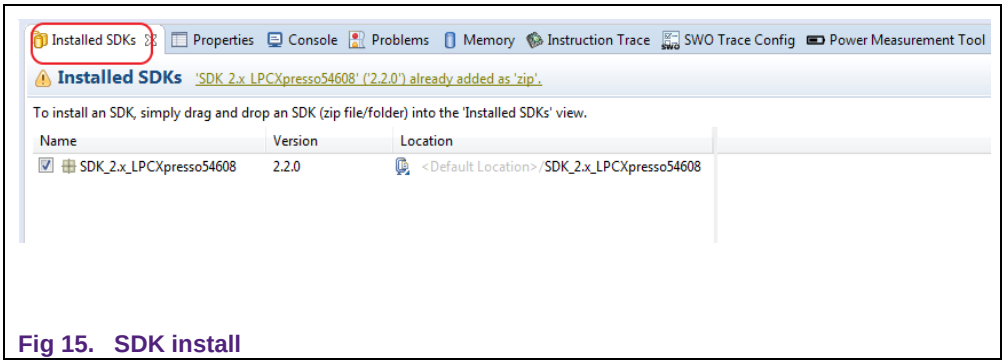

Fig 15. SDK install

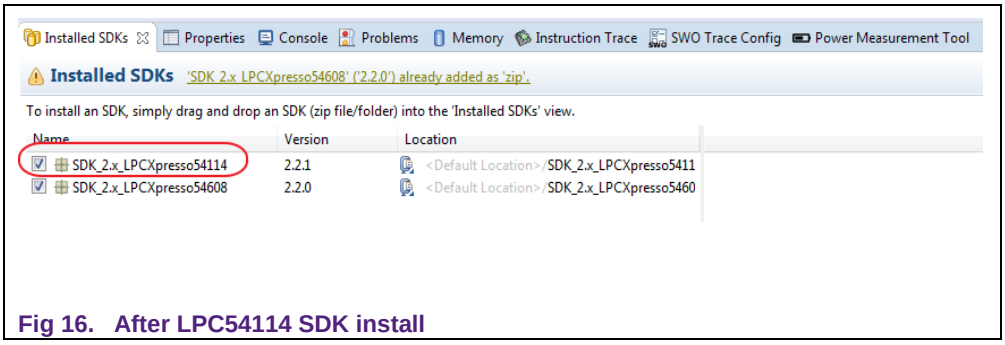

Fig 16. After LPC54114 SDK install

2. Import dual core project

After SDK installation, the dual core project within it can be imported. Click *Import SDK example(s)….* from the *Quick Start Panel*.

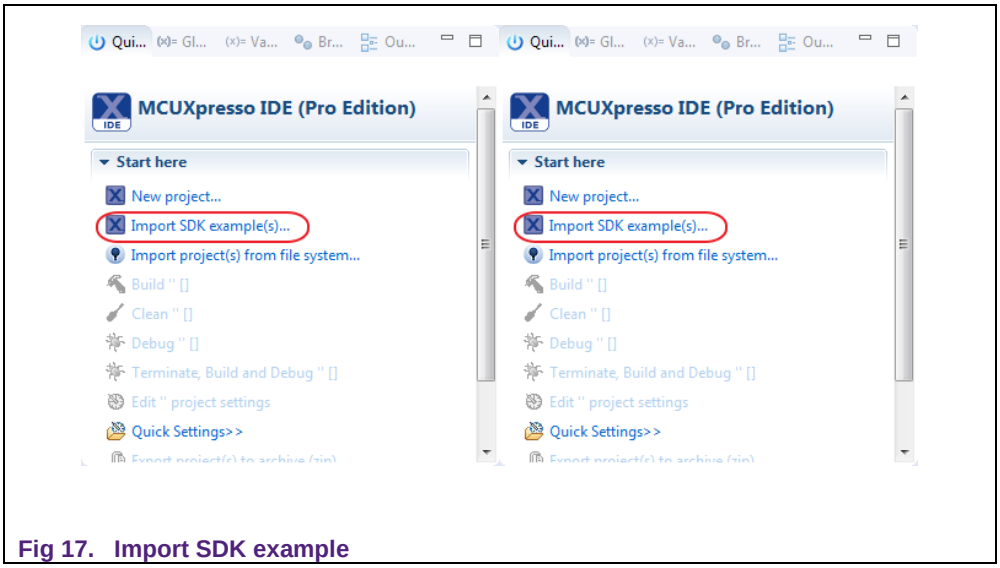

Fig 17. Import SDK example

Choose *LPCXpresso54114* board and click *Next*.

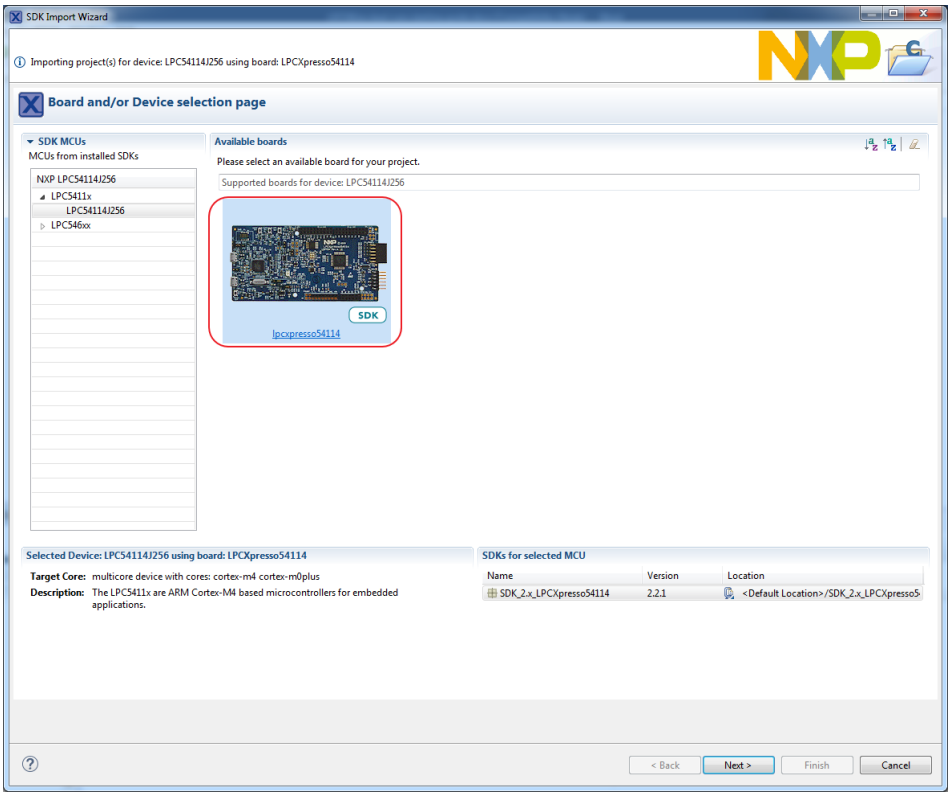

Fig 18. LPCXpresso54114 board

Choose project *hello_world* in the *multicore_examples*, and click *Finish*. The dualcore example is imported in the workspace.

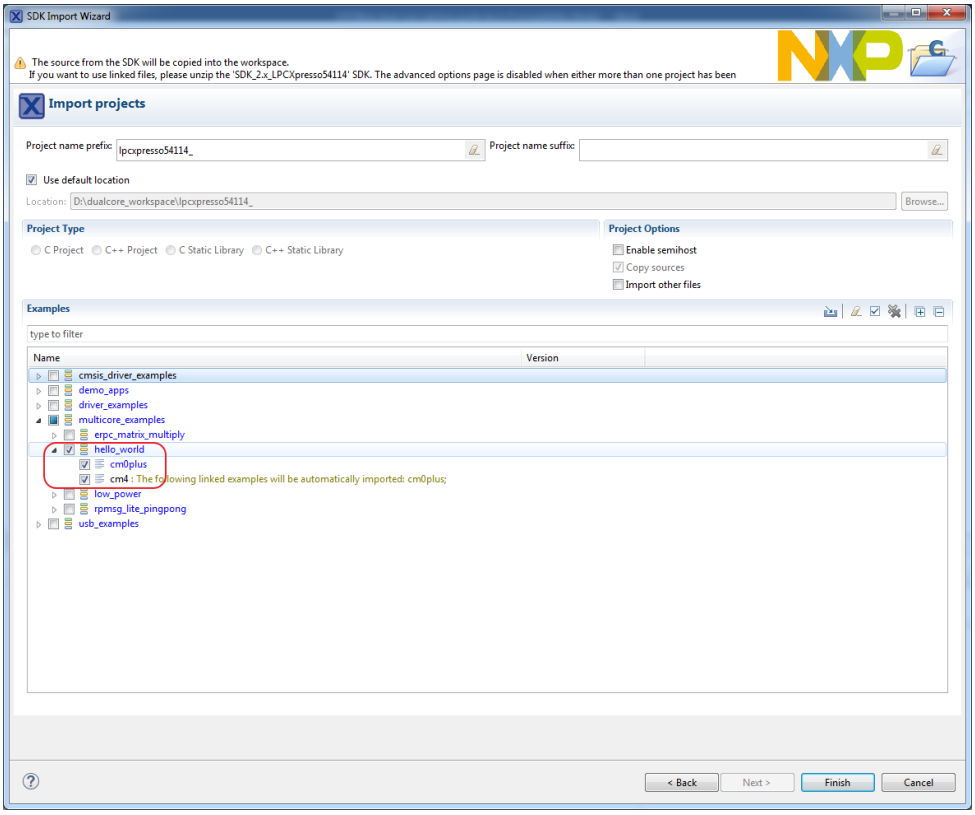

Fig 19. Multicore example “hello_world”

Imported multicore example master and slave project.

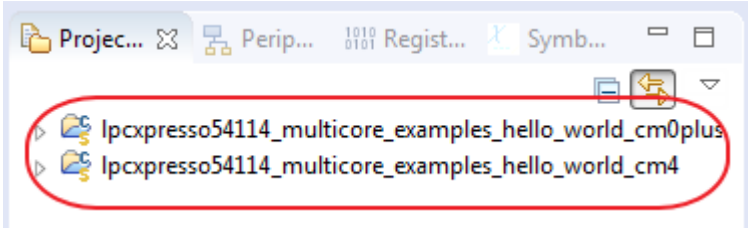

Fig 20. Imported multicore projects

4.3 Configure multicore project

Multicore development requires two projects, one for master and the other for slave. *lpcxpresso54114_multicore_examples_hello_world_cm0plus* is the slave project for Cortex-M0+. *lpcxpresso54114_multicore_examples_hello_world_cm4* is the master project for Cortex-M4.

- Master project setting

  See Fig 21 for the project setting of master projects. It has LPC54114 as the MCU and Cortex-M4 as the core. Flash is assigned for code allocation and SRAM0(or Ram0_64) is the default data allocation memory. Other SRAM banks should be specified by the application or multicore option for data allocation.

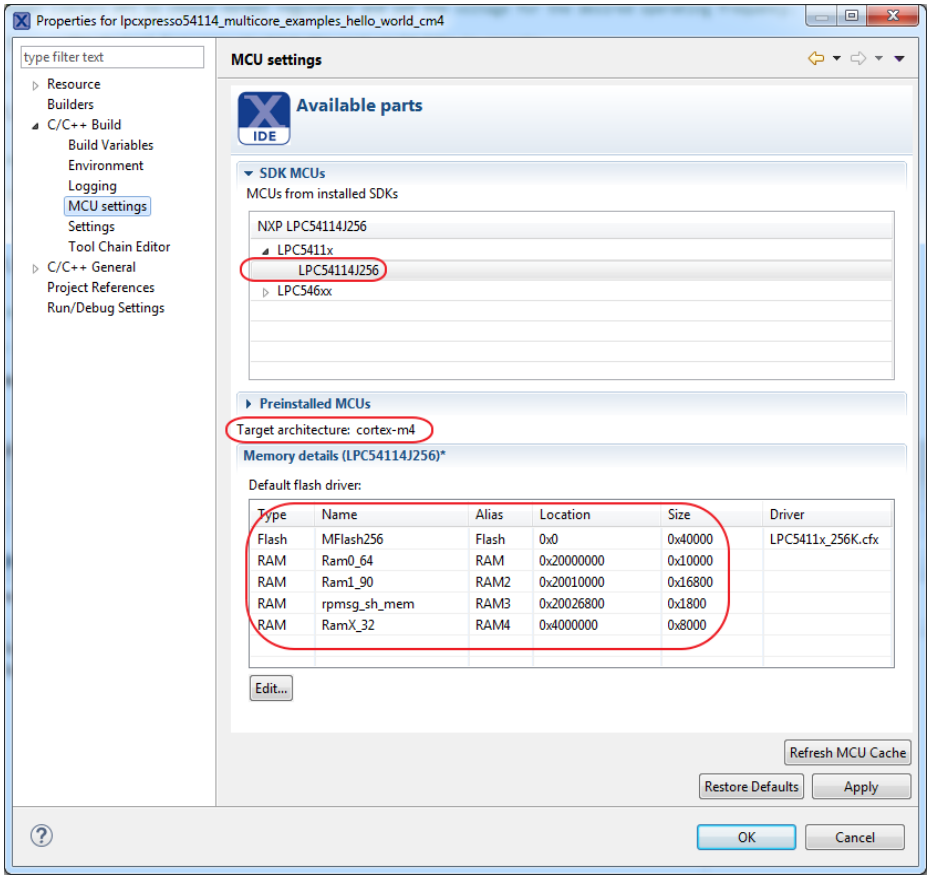

Fig 21. Cortex-M4 master project setting

SRAM1(or Ram1_90) is assigned for slave project according to multicore option setting. This memory space is used for Cortex-M0+ project code and data allocation, which should be in accordance with the slave project setting. The slave project is also associated to the *slave application(object)* setting.

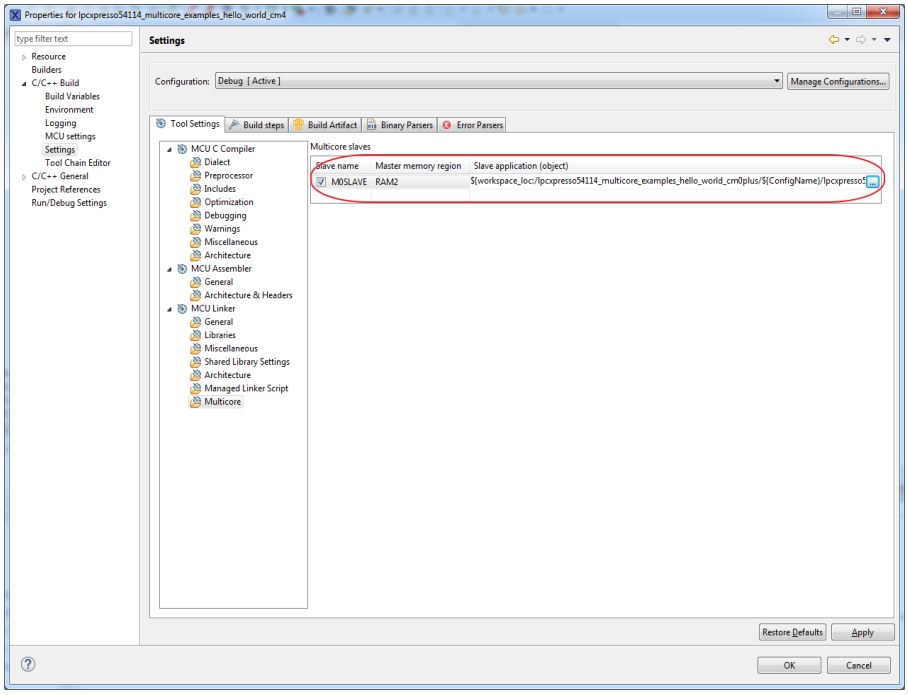

Fig 22. Multicore option to associate slave project

- Slave project setting

  See Fig 23 for the Cortex-M0+ slave project setting. SRAM1(or Ram1_90) is the default memory space for code and data allocation, which is in accordance with the *multicore option* setting of the master project. Other SRAM banks should be specified in the application for specific data or code allocation. Because these banks are shared between the two projects, care should be taken to avoid data access contention. If two cores try to access the same RAM bank simultaneously, arbitration is carried out and only one core can gain access at a given time. The other core will have to wait for access.

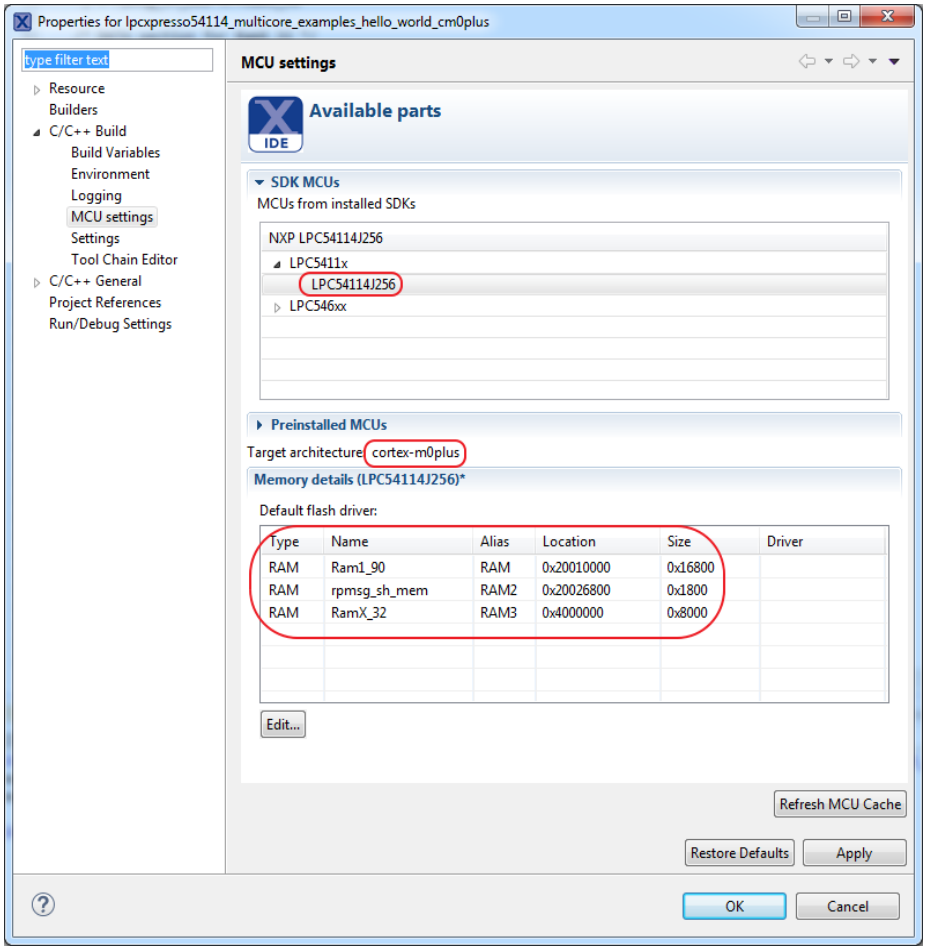

Fig 23. Cortex-M0+ slave project setting

4.4 Debug multicore project

1. Compile the project

   Click to choose the master project

   lpcxpresso54114_multicore_examples_hello_world_cm4, and then compile it.

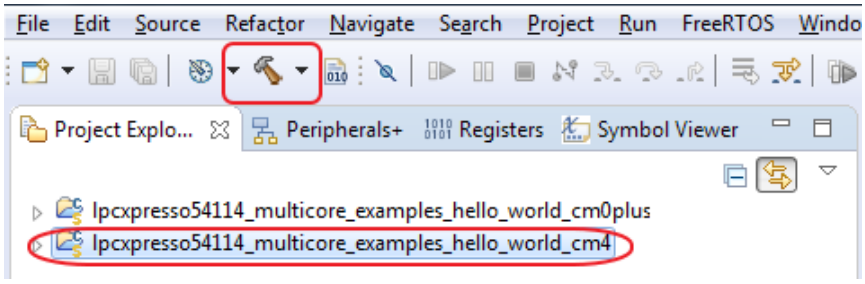

Fig 24. Compile the master project

The slave project is associated with the master project. So, it will be compiled first and then the image data will be linked into the master image file. Fig 25 is the compiled statistics for the slave project. The text section size is 6746 Bytes, which will be integrated into master project as data section assigned to SRAM1(or Ram1_90).

```
Building target: lpcxpresso54114_multicore_examples_hello_world_cm0plus.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"D:\dualcore_workspace\lpcxpresso54114_multicore_examples_hello_world
Memory region         Used Size  Region Size  %age Used
       Ram1_90:       15212 B        90 KB     16.51%
    rpmsg_sh_mem:          0 GB         6 KB      0.00%
        RamX_32:          0 GB        32 KB      0.00%
copy from `lpcxpresso54114_multicore_examples_hello_world_cm0plus.axf' [elf32-littlearm] to `lpcxpr
Finished building target: lpcxpresso54114_multicore_examples_hello_world_cm0plus.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "lpcxpresso54114_multicore_examples_hello_world_cm0plus.axf"; # arm-none-eabi-ob
   text    data     bss     dec     hex filename
   6756       0    8456   15212    3b6c lpcxpresso54114_multicore_examples_hello_world_cm0plus.axf
```

Fig 25. Slave project text size

Fig 26 is the compiled statistics for the master project.

```
Building target: lpcxpresso54114_multicore_examples_hello_world_cm4.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"D:\dualcore_workspace\lpcxpresso54114_multicore_examples_hello_w
Memory region         Used Size  Region Size  %age Used
       MFlash256:       20280 B       256 KB      7.74%
         Ram0_64:        8472 B        64 KB     12.93%
         Ram1_90:        6756 B        90 KB      7.33%
    rpmsg_sh_mem:           0 GB         6 KB      0.00%
         RamX_32:           0 GB        32 KB      0.00%
Finished building target: lpcxpresso54114_multicore_examples_hello_world_cm4.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "lpcxpresso54114_multicore_examples_hello_world_cm4.axf"; # arm-none-eabi-ob
   text    data     bss     dec     hex filename
  20276       4    8468   28748    704c lpcxpresso54114_multicore_examples_hello_world_cm4.axf
```

Fig 26. Master project compiled results

2. Debug the projects

   Connect the LPCXpresso54114 board to the host PC via the debug interface J7 using USB cable. Choose the master project *lpcxpresso54114_multicore_examples_hello_world_cm4* and click the debug command.

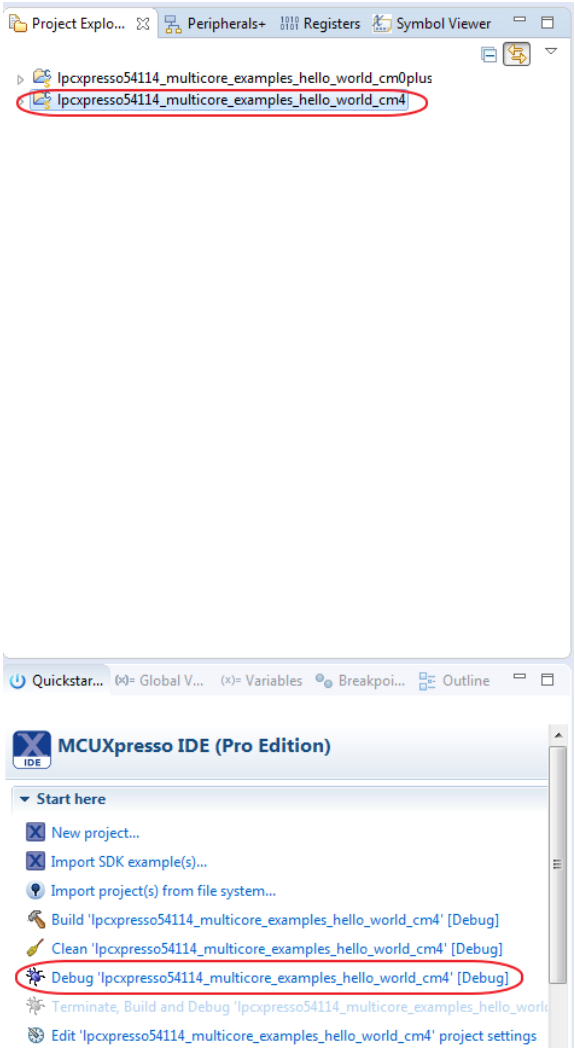

Fig 27. Debug master project

LPC-Link2 debugger will be automatically discovered. Click *OK* to continue.

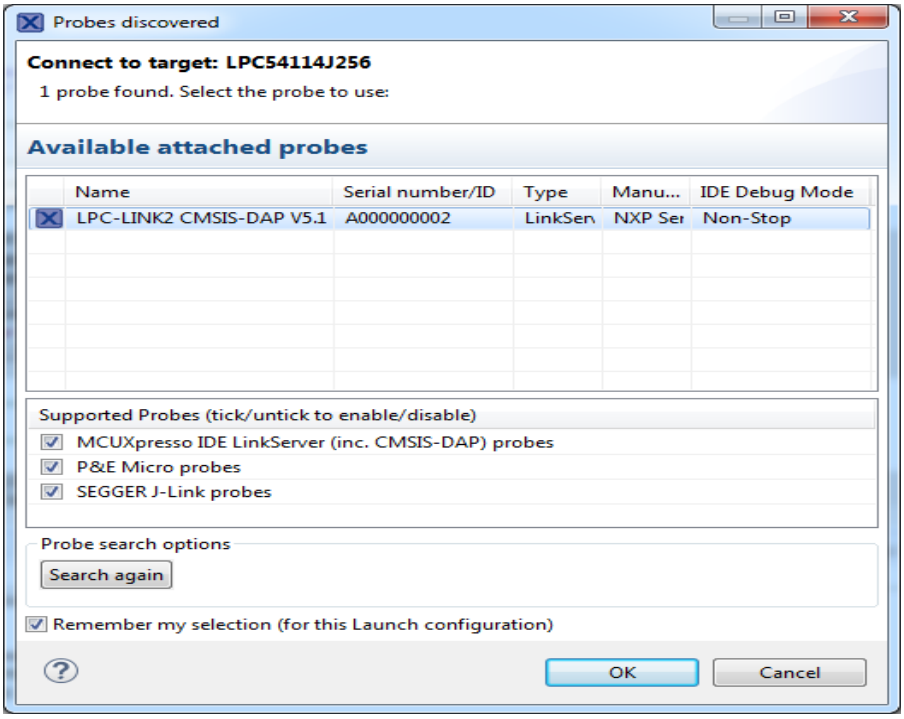

Fig 28. LPC-Link2 debugger discovered

The program will be stopped at the first statement of the main () function.

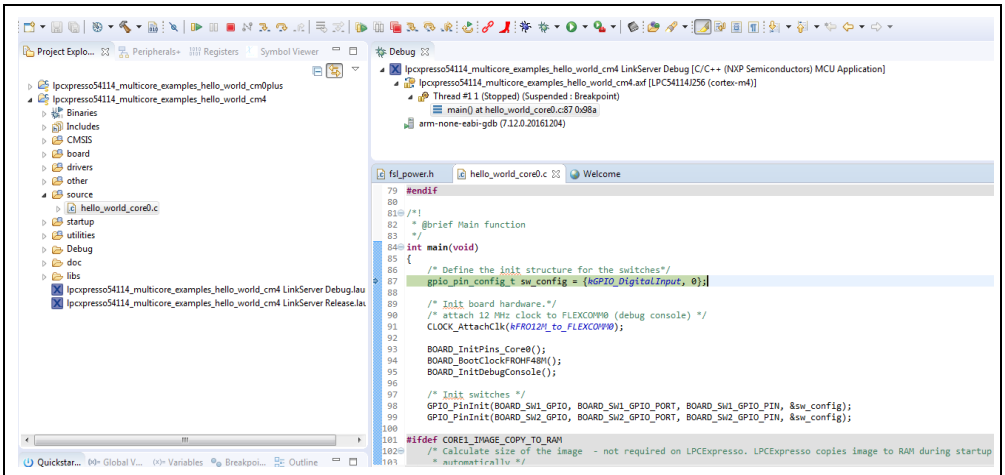

Fig 29. Debugger stopped at main () function

Choose slave project and start debugging, Cortex-M0+ core will be detected. Click *OK* to continue.

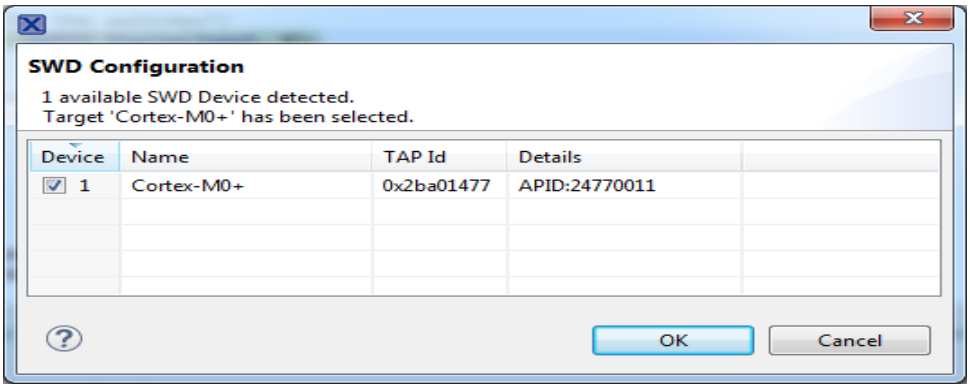

Fig 30. Debug slave project and Cortex-M0+ detected

Both cores are in the debugging state. Since the boot address and initial stack of the slave Cortex-M0+ core is not initialized by the master, it will enter sleep state.

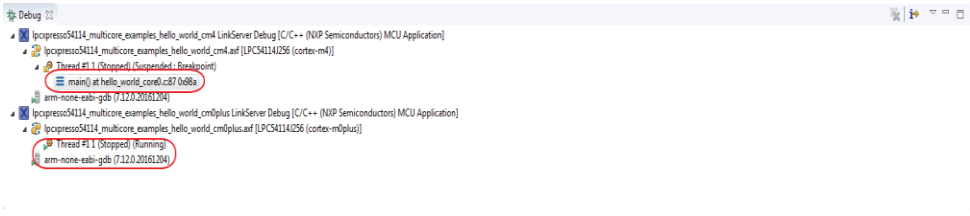

Fig 31. Debug dual core at the same time

When the boot slave core function MCMGR_StartCore () is called in the master project, it will initialize the slave boot address and initial stack, then reset the slave (namely, the M0+ core).

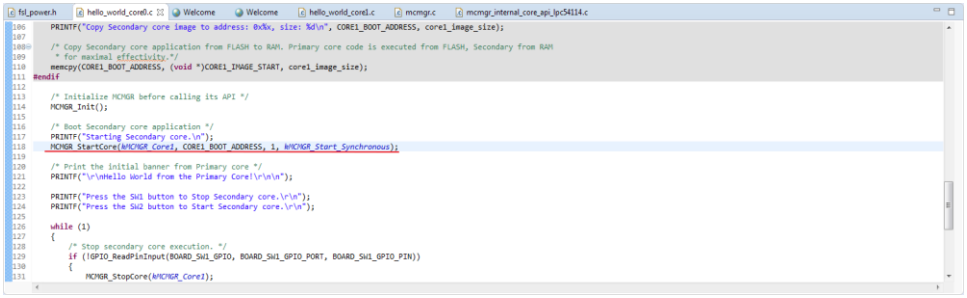

Fig 32. MCMGR_StartCore () function to initialize and start slave core

This time, the slave core will start executing from the boot address initialized, which is the *lpcxpresso54114_multicore_examples_hello_world_cm0plus* image text. The slave project suspends at the first statement of main () function.

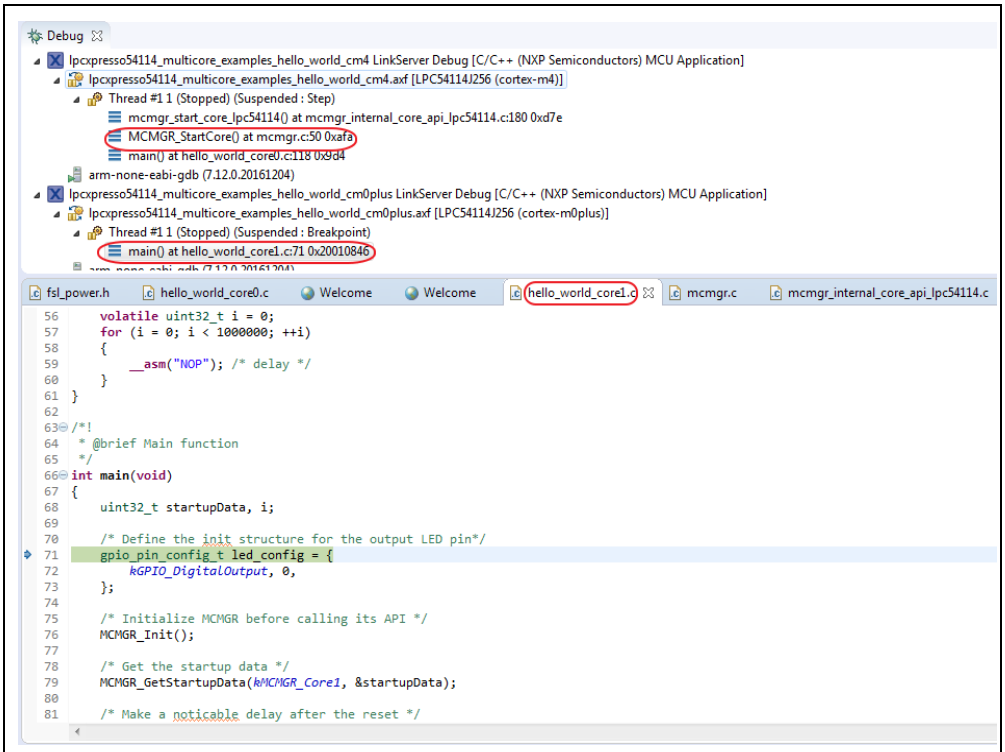

Fig 33. Slave core suspended at main () function

Resume and run the example projects. Detailed description of the project is available in the readme file; see Fig 34.

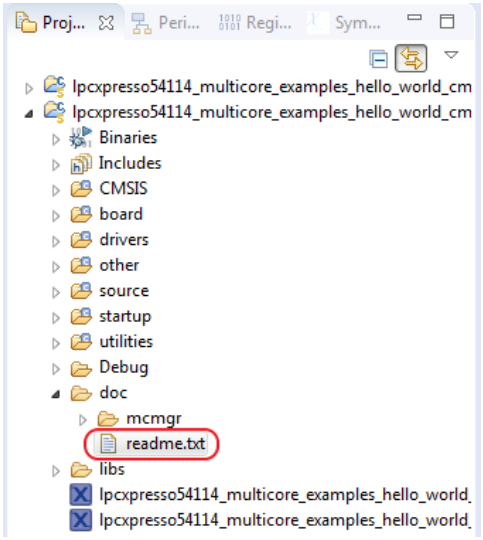

Fig 34. Readme file

5. Conclusion

This application note introduces the basic concept of asymmetric dual core and its implementation in LPC541xx serial MCU. User can have a general idea for designing an application based on this feature. The fundamental process is illustrated using MCUXpresso IDE and SDK examples.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

6.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

6.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP Semiconductors N.V.

7. List of figures

8. Contents

Date of release: 7 March 2018
Document identifier: AN12123