NXP Semiconductors
Application Note

Document Number: AN5302
Rev. 0, 06/2016

Getting Started with KL28T Dual Core

1. Introduction

KL28T is a new Kinetis L series product with many new features and modules which differ from all previous Kinetis L series. It contains new NXP peripherals, which greatly reduce power consumption, as well as maintaining strong performance. The most exciting change on KL28T is that it has 2 CPUs (Core0 and Core1) that support Asymmetric Multi-core Processing (AMP).

This application note will give you an overview of the KL28T dual core architecture, and also provide some useful information for users to get started with the KL28T dual core system.

Contents

2. Dual Core System Overview

This section provides an overview of the KL28T dual core system. It gives users high-level descriptions of the whole dual core architecture.

The KL28T series has 2 CPUs (called cores) that support Asymmetric Multi-core Processing (AMP):

- Core0 is used as the main customer application core
- Core1 is the second core

All on-chip peripherals are attached on two AIPS bridges. Peripherals attached to the AIPS1 Bridge are part of the Core1 subsystem. Having 2 AIPS bridges enables the Core1 firmware to interface with Core1 peripherals independently from Core0.

Both cores and DMAs have the capability to access peripherals on the AIIPS0 and AIPS1 bridges. All on-chip resources, including all peripherals and memory modules, are accessible for both cores.

If multiple masters request access to peripherals on the same AIPSx Bridge at the same time, then arbitration takes place, as per the AIPSx configuration. The Core0/DMA0 has priority for AIPS0, and the Core1/DMA1 has priority for AIPS1.

Core0 and Core1 share one SCG (System Clock Generator) as input clock source. So Core0 and Core1 have the exact same core clock frequency in normal run mode.

3. Memory Resource Assignment

The following table contains useful and important information to help understand the dual core memory system on KL28T:

Table 1. Dual core system memory map terminology

Name	Descriptions
IMEM0	ROM space for Core0, This is alias to the top of 256KB of flash memory
DMEM0	RAM space for Core0, This is Core0 SRAM
IMEM1	ROM space for Core1, This is alias to the bottom of 256KB of flash memory
DMEM1	RAM space for Core1, This is Core1 SRAM

Most applications will use the memory range IMEM0/DMEM0 for Core0, and IMEM1/DMEM1 for Core1. However, both cores can access all the on chip memory ranges with proper configuration. The core1 boot address is not limited to the start address of IMRM1/DMEM1, you can boot core1 at 0x000_0000 by configuring the MU->CR[BBOOT] field.

The IMEM and DMEM physical address and aliased physical address are shown in the following table:

Table 2. Dual core system memory map[4]

Type	Start address	End Address	Size	Function
ROM	0x0000_0000	0x0003_FFFF[1]	256K	Program Flash0(IMEM0)
ROM	0x0004_0000	0x0007_FFFF	256K	Program Flash1(IMEM1)
RAM	0x1FFF_A000	0x1FFF_FFFF	24K	SRAM0(DMEM0)
RAM	0x2000_0000	0x2001_1FFF	72K	SRAM0(DMEM0)
RAM	0x2D30_0000	0x2D30_7FFF	32K	SRAM1(DMEM1)
ROM	0x2D20_0000	0x2D23_FFFF	256K	IMEM1 aliased in this address range[2]
RAM	0x2D10_0000	0x2D10_7FFF	32K	DMEM1 aliased in this address range[3]

[1] This table is for KL28T which contain 512K flash and does not include BOOTROM, USBRAM. address space

[2] IMEM1 may also be aliased in these address ranges. It is recommended not to use these address spaces.

[3] DMEM1 may also be aliased in these address ranges. It is recommended not to use these address spaces.

[4] For more detail information about dual core memory assignments, refer to the KL28T Reference Manual, Chapter 4 - Memory Map.

4. Dual Core Related Peripherals

This section gives a brief overview of the multicore related peripherals on KL28T which include:

- Message Unit (MU) - multicore status and control, inter processor messaging. and interrupt signal triggering
- SEMA42 - hardware semaphore
- XRDC (Extended Resource Domain Controller) - an integrated, scalable architectural framework for access control, system memory protection. and peripheral isolation

4.1. Message Unit (MU)

The MU is the most important peripheral among the three multicore related peripherals. The MU has the following features:

- enables two cores to communicate and coordinate by passing messages
- provides the status and control function for one core to monitor and control to another
- enables one core to signal to the other core by using interrupts
- it can be used as hardware implementation for a high-level IPC (Inter Processor Communication) components

The MU is connected as a peripheral under the peripheral bus on both sides. On the Core0 side, it is named *MU_A*. On the core1 side, it is named *MU_B*. The status register of one MU side reflects the status of the other MU side.

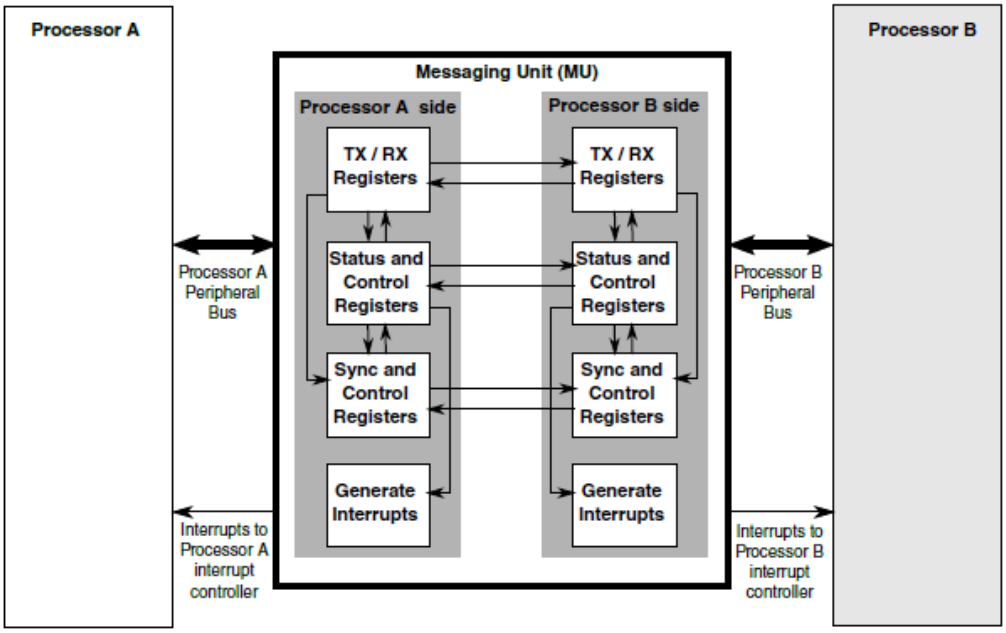

Figure 1. MU block diagram

The key features are:

- Messaging control by interrupts or by polling
- Symmetrical processor interfaces with each side supporting the following:
  - Four general-purpose interrupt requests reflected to the other side
  - Three general-purpose flags reflected to the other side
  - Four receive registers with maskable interrupt
  - Four transmit registers with maskable interrupt

4.2. SEMA42

The SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores. It provides a simple mechanism to achieve "lock and unlock" operations via a single write access.

Each SEMA42 module contains 16 GATEn registers. Each of these SEMA42_GATEn registers implements a 4-bit, 16-state machine. A simplified diagram of the state transitions for each gate is shown in the following figure:

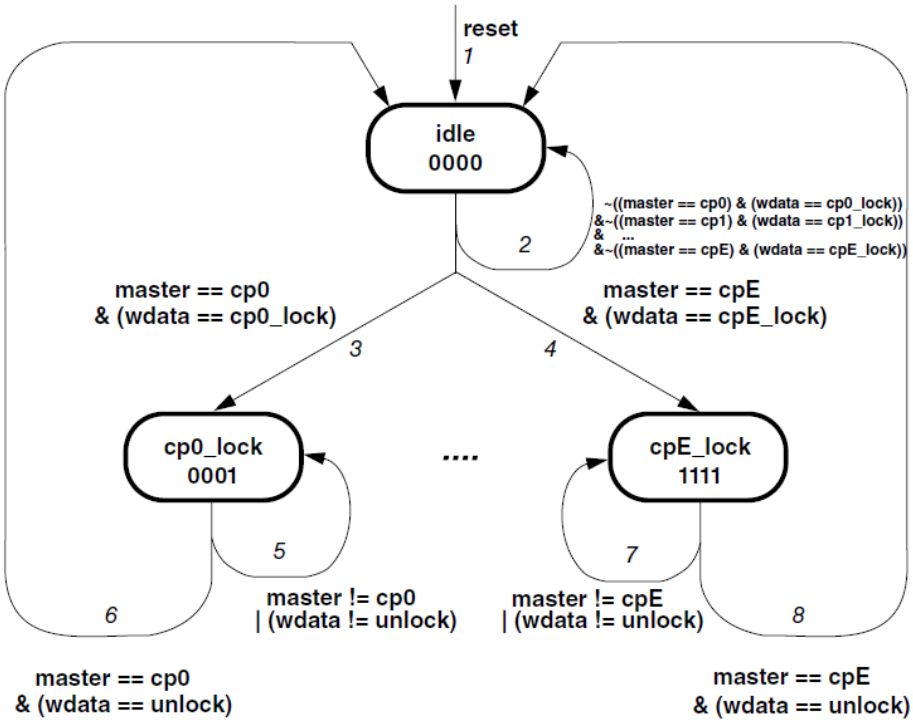

Figure 2. SEMA42 block diagram

A bus master number is used to identify the core X (cpX). The platform passes the AHB bus master number through the peripheral bridge controller and drives a signal to the SEMA42. The bus master number is generated by the XRDC_MDAC register. It can also be generated by first resetting the SEMA42 gate and then reading the SEMA42x_RSTGT register.

4.3. XRDC

The Extended Resource Domain Controller (XRDC) provides an integrated, scalable architectural framework for access control, system memory protection, and peripheral isolation. It enables software to configure processor cores, non-core bus masters to assign chip peripheral such as memory regions and slave peripherals to different access level. This enables users to build strong and robust operational environments.

In the XRDC module, an important concept called “domain” will be used. First, each bus master will be assigned to a domain identifier (Domain, DID). Next, the access control policies for the individual domains are determinate by programming slave memory region descriptors and slave peripheral domain access control registers. Finally, once all domains are configured properly and XRDC is globally enabled, all accesses throughout the device are monitored concurrently to determine the validity of each access. If an access request from a given domain has sufficient access rights, it is allowed to continue, otherwise, the access is aborted and error signal will be issued.

The XRDC has several sub-modules:

- MGR (Manager): coordinates all programming model reads and writes
- MDAC (Master Domain Assignment Controller): handles resource assignments and generation of the domain identifiers

- MRC (Memory Region Controller): implements the access controls for slave memories based on the pre-programmed region descriptor registers
- PAC (The Peripheral Access Controller): implements the access controls for slave peripherals based on the pre-programmed domain access control registers

Each sub-module has several instances for a given microcontroller. The instance number is read from the XRDC_HWCFGn register. The following figure shows the simplified block diagram of XRDC:

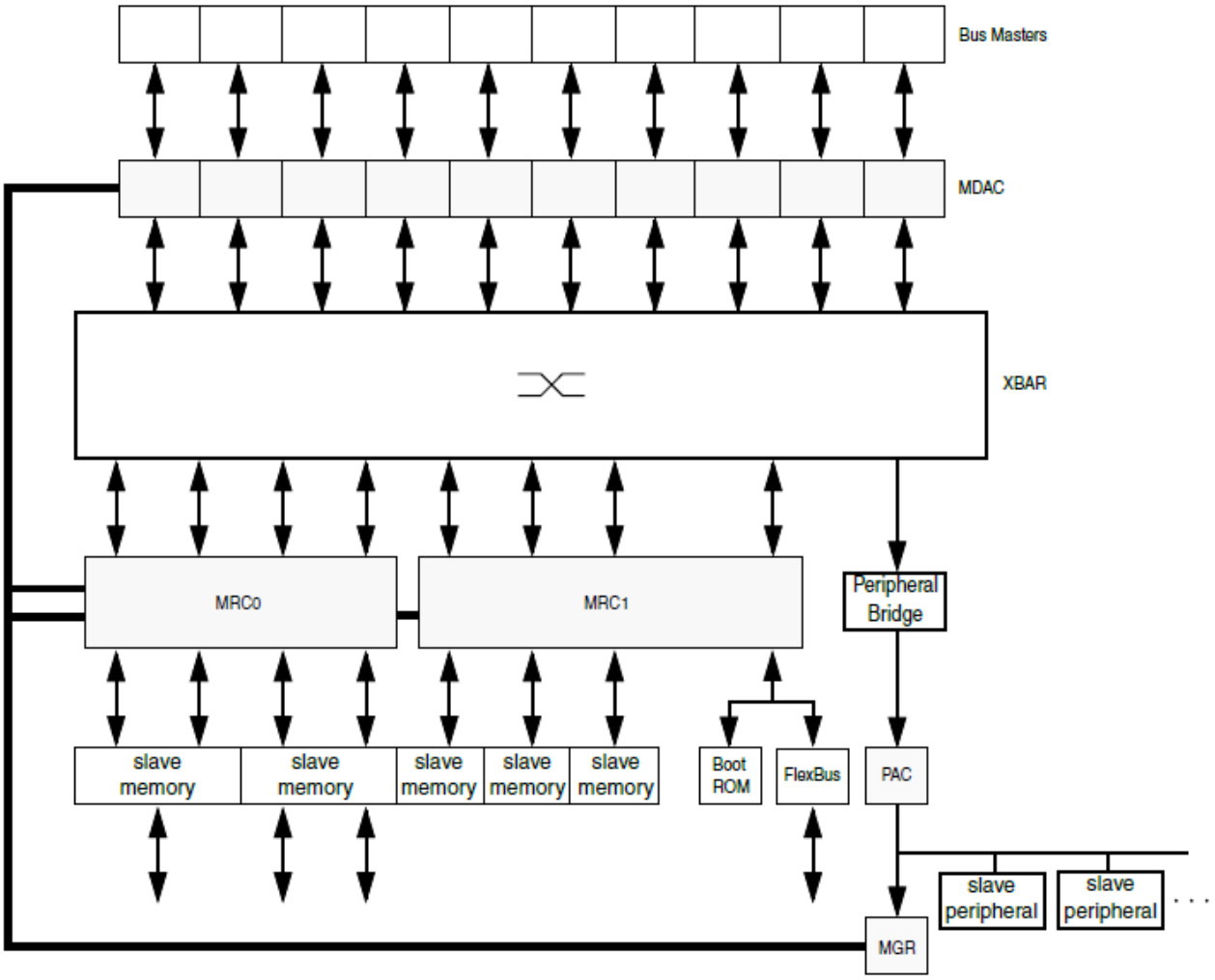

Figure 3. XRDC block diagram

The key steps when configuring XRDC with application code are as follows:

- Configure MRC
  - MRC provides domain-based, hardware access control for all system memory spaces. You must assign correct memory spaces and their associated access rights per domain identifier. This is done by using the KSDK HAL function:

    XRDC_HAL_SetMemAccessConfig

- Configure PAC
  - The peripheral access controller performs an access control evaluation similar to that of the MRC
  - Each slave peripheral slot has one unique PDAC_W [0-1] register. You can program access right and other configuration into those registers. This can be done by using KSDK HAL function: MRC.XRDC_HAL_SetPeriAccessConfig
- Configure MDAC
  - The MDAC submodule is mainly responsible for the generation of the domainID. The domainID are generated and then treated as address attributes and associated with each transaction as it moves through the system. The MDAC is also responsible for configuring several other master attributes. Configuring the MDAC is done by using the SDK HAL function: XRDC_HAL_SetMasterDomainConfig
- Enable XRDC
  - After all configuration is complete, set the XRDC_CR [GVLD] to make the XRDC module valid

5. Software Resources for Dual Core Application

5.1. Kinetis SDK

All dual core demo and dual core related peripheral drivers can be found on Kinetis SDK package, and it also provides the startup file, simple dual-core application, and FreeRTOS porting for KL28.

5.2. MCSDK (Multicore SDK)

In addition to the Kinetis SDK, NXP offers the MCSDK (Multicore SDK) that provides industry-leading ease-of-use for building multicore applications. The MCSDK is a collection of software components that enables development of both generic asymmetric multicore, as well as multicore applications. It is featured as a middleware component in the Kinetis SDK.

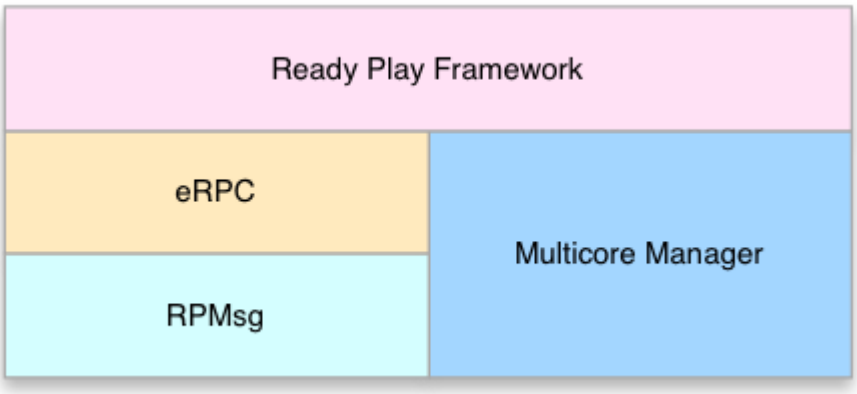

Figure 4. MCSDK architecture

Several software components are included in MCSDK:

1. RPMsg — Inter Processor Communication
2. Embedded Remote Procedure Call (eRPC) implementation
   - provides transparent function call interface to remote services
   - eRPC server and client implementation, SHIM code generator
   - abstracted transport interface: RPmsg is the primary transport for multicore, UART or SPI-based solutions can be used for both multi-chip and multicore
3. Multicore Manager: core startup and configuration
4. Multicore Application Framework: building block and client API specific to RPM apps

KSDK and MCSDK can be downloaded at: nxp.com/ksdk

6. How to Download Dual Core Demo to KL28

This section describes the steps required to configure the IAR Embedded Workbench® development tools and use it to build, run, and debug dual-core applications for the NXP Kinetis MKL28 part.

The multicore debugging in IAR means that there is one master project that contains a link to the slave project. After the debug button is pressed on the master side, the slave project opens (second EWARM instance) and dual-core-specific controls such as *start all cores* or *stop all cores* can be used from both the master and slave sides (multicore toolbar).

6.1. Building SDK library and example applications

After the platform driver libraries are built, the user can focus on building an application. As an example, here is how to build a simple dual-core version of the HelloWorld application. The demo applications workspace files are located in this folder:

<ksdk_install_dir>/examples/<board_name>/demo_apps/multicore/<demo_name>/<toolchain><demo_name>.eww

Using the FRDM-KL28T NXP Freedom Development board as an example, the multicore hello_world IAR workspaces are located in this folder:

<ksdk_install_dir>/examples/frdmkl28t/demo_apps/multicore/hello_world/iar/hello_world_core0.eww
<ksdk_install_dir>/examples/frdmkl28t/demo_apps/multicore/hello_world/iar/hello_world_core1.eww

It is recommended to build the application for the secondary core (core1) first, because the primary core application project has to know the secondary core application binary when running the linker. The primary core debugger handles flashing of both primary and secondary core application into the SoC flash memory.

6.2. Running and debugging the multicore application

To run a demo application, attach the I-Jet debugger to the KL28 SWD connector of the FRDM-KL28T board (J11) and open the Terminal Window application by using Port USB COM, Baud 115200, Parity None, and Bits 8.

In project/Options-Debugger, check the debugger driver settings. The default is I-Jet is currently the only supported debugger probe for multicore debugging.

When applications for both the primary and the secondary core are compiled and linked to KSDK platform libraries, switch to the primary core application project and press the *Download and Debug* button to download applications code to target and start the debugging.

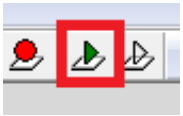

Figure 5. "Download and Debug" button

During this step, the secondary core project is opened in the separate EWARM instance. Both the primary and the secondary image is loaded into the device flash memory, and the primary core application is executed and stops at the default C language entry point in the main() function.

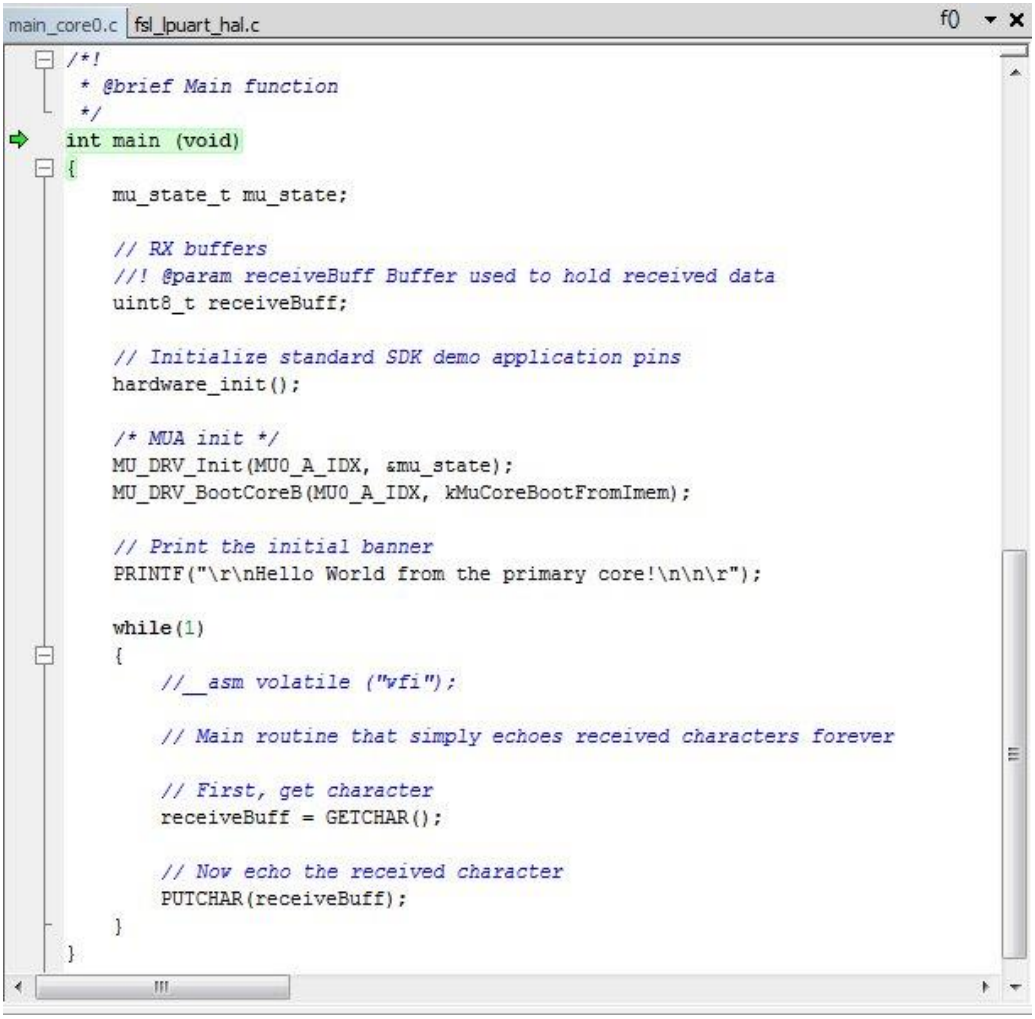

Figure 6. Run the main() function

Normally, the *Start all cores* control button should be used to run both the primary/master and secondary/slave applications at once. It is recommended to click the *Go* control button on the master side first. After the secondary/slave core is released from reset by the primary core application, switch to the slave project Debug window and press the *Go* button again to enable the slave application to run.

A “Hello World from the primary core!” message is printed on the serial console terminal by the primary core.

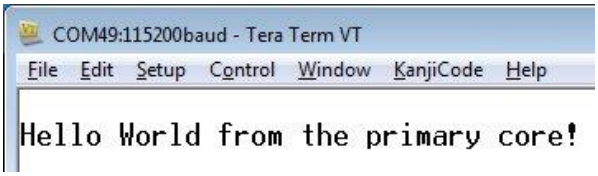

Figure 7. Hello World message

An LED controlled by the secondary core starts flashing, indicating that the secondary core has been released from the reset and running correctly.

When both cores are running, use *Stop all cores* and *Start all cores* control buttons to stop/run both cores simultaneously. 

6.3. Further information

For further information on how to enable multicore in specific IDE and related project settings, refer to the document *Getting_started_with_MKL28_dualcore_in_IAR.pdf*. This PDF is in the installation path of the multicore SDK/doc folder.

7. References

The following references are available on nxp.com:

- KL28 Reference Manual
- KL28 Data Sheet
- KINETIS_SDK: Software Development Kit for Kinetis MCUs
- Getting Started with Kinetis MKL28 Dual-Core and IAR Embedded Workbench

8. Revision History

Table 3. Revision history

Revision number	Date	Substantive changes
0	06/2016	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN5302
Rev. 0
06/2016