**NXP Semiconductors**
Release Notes

Document Number: MCUXSDKIMX7ULPRN
Rev. 0, 01/2019

# MCUXpresso SDK Release Notes Supporting EVK-MCIMX7ULP

**Contents**



## 1 Overview

The MCUXpresso Software Development Kit (SDK) is a collection of software enablement for Microcontrollers that includes peripheral drivers, high-level stacks including integration with WolfSSL and mbed TLS cryptography libraries, other middleware packages, such as multicore support, and integrated RTOS support for FreeRTOS™ OS. In addition to the base enablement, the MCUXpresso SDK is augmented with demo applications and driver example projects, and API documentation to help the customers quickly leverage the support of the MCUXpresso SDK.

For the latest version of this and other MCUXpresso SDK documents, see the MCUXpresso SDK homepage MCUXpresso-SDK: Software Development Kit.

**NOTE**

See the attached Change Logs section at the end of this document to reference the device-specific driver logs, middleware logs, and RTOS log.

## 2 MCUXpresso SDK

As part of the MCUXpresso software and tools, MCUXpresso SDK is the evolution of Kinetis SDK v2.4.0, includes support for both LPC and i.MX System-on-Chips (SoC). The same drivers, APIs, and middleware are still available with support for Kinetis, LPC, and i.MX silicon. The MCUXpresso SDK adds support for the MCUXpresso IDE, an Eclipse-based toolchain that works with all MCUXpresso SDKs. Easily import your SDK into the new toolchain to have access to all of the available components, examples, and demos for your target silicon. In addition to the MCUXpresso IDE, support for the MCUXpresso Config Tools allows for easy cloning of existing SDK examples and demos, allowing users to easily leverage the existing software examples provided by the SDK for their own projects.

**NOTE**

In order to maintain compatibility with legacy Freescale code, the filenames and source code in MCUXpresso SDK containing the legacy Freescale prefix 'FSL' has been left as is. The 'FSL' prefix has been redefined as the NXP Foundation Software Library.

# 3 Development Tools

The MCUXpresso SDK was compiled and tested with these development tools:

- IAR Embedded Workbench for ARM version 8.32.1
- Makefiles support with GCC revision 7-2018-q2-update from Arm Embedded

# 4 Supported Development Systems

This release supports boards and devices listed in this table. The boards and devices in boldface were tested in this release.

**Table 1. Supported MCU devices and development boards**

| Development boards | MCU devices |
|---|---|
| **MCIMX7ULP-EVK RevB1** | **MCIMX7U5DVP08**, MCIMX7U5DVK08, MCIMX7U5CVP05, MCIMX7U3DVK08, MCIMX7U3CVP05 |

# 5 Release contents

This table provides an overview of the MCUXpresso SDK release package contents and locations.

**Table 2. Release contents**

| Deliverable | Location |
|---|---|
| Boards | <install_dir>/boards |
| Demo applications | <install_dir>/boards/<board_name>/demo_apps |
| USB demo applications | <install_dir>/boards/<board_name>/usb_examples |
| Driver examples | <install_dir>/boards/<board_name>/driver_examples |
| RTOS examples | <install_dir>/boards/<board_name>/rtos_examples |
| Multicore examples | <install_dir>/boards/<board_name>/multiprocessor_examples |
| Documentation | <install_dir>/docs |

*Table continues on the next page...*

**Table 2. Release contents (continued)**

| | |
|---|---|
| USB Documentation | <install_dir>/docs/usb |
| lwIP Documentation | <install_dir>/docs/lwip |
| Middleware | <install_dir>/middleware |
| lwIP stack | <install_dir>/middleware/lwip |
| DMA manager | <install_dir>/middleware/dma_manager |
| EMV stack | <install_dir>/middleware/emv |
| FatFS stack | <install_dir>/middleware/fatfs |
| mmCAU | <install_dir>/middleware/mmcau |
| Motor Control libraries | <install_dir>/middleware/motor_control |
| Multicore stack | <install_dir>/middleware/multicore |
| RTCESL libraries | <install_dir>/middleware/rtcesl |
| SDMMC card driver | <install_dir>/middleware/sdmmc |
| USB stack | <install_dir>/middleware/usb |
| WolfSSL stack | <install_dir>/middleware/wolfssl |
| Driver, SoC header files, extension header files and feature header files, utilities | <install_dir>/devices/<device_name> |
| Cortex Microcontroller Software Interface Standard (CMSIS) ARM Cortex®-M header files, DSP library source | <install_dir>/CMSIS |
| Peripheral Drivers | <install_dir>/devices/<device_name>/drivers |
| Utilities such as debug console | <install_dir>/devices/<device_name>/utilities |
| RTOS Kernel Code | <install_dir>/rtos |
| Tools | <install_dir>/tools |

# 6 MCUXpresso SDK release package

The MCUXpresso SDK release package contents are aligned with the silicon subfamily it supports. This includes the boards, CMSIS, devices, documentation, middleware, and RTOS support.

## 6.1 Device support

The device folder contains all available software enablement for the specific System-on-Chip (SoC) subfamily. This folder includes clock-specific implementation, device register header file, device register feature header file, CMSIS derived device SVD, and the system configuration source files. Included with the standard SoC support are folders containing peripheral drivers, toolchain support, and a simple debug console.

The device-specific header files provide a direct access to the MCU peripheral registers. The device header file provides an overall SoC memory mapped register definition. In addition to the overall device memory mapped header file, the MCUXpresso SDK also includes the feature header file for each peripheral instantiated on the SoC.

The toolchain folder contains the startup code and linker files for each supported toolchain. The startup code is a CMSIS-compliant startup that efficiently transfers the code execution to the main() function.

### 6.1.1 Board support

The boards folder provides the board-specific demo applications, driver examples, RTOS, and middleware examples.

### 6.1.2 Demo applications and other examples

The demo applications demonstrate the usage of the peripheral drivers to achieve a system level solution. Each demo application contains a readme file that describes the operation of the demo and required setup steps.

The driver examples demonstrate the capabilities of the peripheral drivers. Each example implements a common use case to help demonstrate the driver functionality.

The RTOS and middleware folders each contain examples demonstrating the use of the included source.

**NOTE**

Most demo applications and all driver examples are intended for a single ARM Cortex-M4 application reference. They cannot support running with the Linux BSP, which requires additional service protocol implementation. See the readme file for the specific application to know whether it supports running with the Linux BSP.

## 6.2 Middleware

## 6.3 RTOS

The MCUXpresso SDK is integrated with FreeRTOS OS.

## 6.4 CMSIS

The MCUXpresso SDK is shipped with the standard CMSIS development pack, including the prebuilt libraries.

## 6.5 Security libraries

The MCUXpresso SDK is pre-integrated with mbedTLS and wolfSSL libraries. The integration demonstrates hardware acceleration of various cryptography algorithms and random number generation.

# 7 MISRA compliance

All MCUXpresso SDK drivers and USB stack comply to MISRA C 2012 rules with the following exceptions.

## Table 3. MISRA exceptions

| Exception Rules | Description |
|---|---|
| Directive 4.4 | Sections of code should not be commented out. |
| Directive 4.5 | Identifiers in the same name space with overlapping visibility should be typographically unambiguous. |
| Directive 4.6 | Typedef that indicate size and signedness should be used in place of the basic numerical type. |
| Directive 4.8 | If a pointer to a structure or union is never dereferenced within a transaction unit then the implementation of the object should hidden. |
| Directive 4.9 | A function should be used in preference to a function like macro where they are interchangeable. |
| Directive 4.10 | Precautions shall be taken in order to prevent the contents of a header file being included more than once. |
| Directive 4.11 | The validity of values passed to library functions shall be checked. |
| Rule 2.3 | A project should not contain unused type declarations. |
| Rule 2.4 | A project should not contain unused tag declarations. |
| Rule 2.5 | A project should not contain unused macro declarations. |
| Rule 2.7 | There should be no unused parameters in functions. |
| Rule 3.1 | The character sequences /* and // shall not be used within a comment. |
| Rule 5.1 | External identifiers shall distinct. |
| Rule 5.3 | A identifier declared in an inner scope shall not hide an identifier declared in an outer scope. |
| Rule 5.7 | A tag name shall be a unique identifier. |
| Rule 5.9 | Identifiers that define objects or functions with external linkage shall be unique. |
| Rule 8.13 | A pointer should point to a const-qualified type whenever possible. |
| Rule 8.3 | All declarations of an object or function shall use the same names and type qualifiers. |
| Rule 8.6 | An identifier with external linage shall have exactly one external definition. |
| Rule 8.7 | Octal constants shall not be used. |
| Rule 8.9 | A object should be defined at block scope if its identified only appears in a single function. |
| Rule 10.1 | Operands shall not be of an inappropriate essential type. |
| Rule 10.3 | The value of an expression shall not be assigned to an object with a narrower essential type of a different essential type category. |
| Rule 10.4 | Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category. |
| Rule 10.5 | The value of an expression should not be cast to an inappropriate essential type. |

*Table continues on the next page...*

## Table 3. MISRA exceptions (continued)

| | |
|---|---|
| Rule 10.6 | The value of a composite expression shall not be assigned to an object with wider essential type. |
| Rule 10.7 | If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type. |
| Rule 10.8 | The value of a composite expression shall not be cast to a different essential type category or a wider essential type. |
| Rule 11.1 | Conversions shall not be performed between a pointer to a function and any other type. |
| Rule 11.3 | A case shall not be performed between a pointer to object type and a pointer to a different object type. |
| Rule 11.4 | A conversion should not be performed between a pointer to object and an integer type. |
| Rule 11.5 | A conversion should not be performed from pointer to void into pointer to object. |
| Rule 11.6 | A cast shall not be performed between pointer to void and an arithmetic type. |
| Rule 12.1 | The precedence of operators within expressions should be made explicit. |
| Rule 12.2 | The right hand operator of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand. |
| Rule 13.3 | A full expression containing an increment(++) or decrement(--) operator should have no other potential side effects other than that caused by the increment or decrement operator. |
| Rule 13.5 | The right hand operand of a logical && or \|\| operator shall not contain persistent side effects. |
| Rule 14.2 | A for loop shall be well formed. |
| Rule 14.4 | The controlling expressions of an statement and the controlling expression of an iteration-statement shall have essentially Boolean type. |
| Rule 15.5 | A function should have a single point of exit at the end. |
| Rule 16.1 | All switch statements shall be well-formed. |
| Rule 17.1 | The feature of <stdarg.h> shall not be used. |
| Rule 18.4 | The +,-,+=and -=operators should not be applied to an expression of pointer type. |
| Rule 19.2 | The union keyword should not be used. |
| Rule 20.1 | #include directives should only be preceded by preprocessor directives or comments. |
| Rule 20.10 | The #and ## preprocessor operators should not be used. |
| Rule 21.1 | #define and #undef shall not be used on a reserved identifier or reserved macro name. |

# 8 Known issues

## 8.1 Maximum file path length in Windows® 7 Operating System

Windows 7 operating system imposes a 260 character maximum length for file paths. When installing the MCUXpresso SDK, place it in a directory close to the root to prevent file paths from exceeding the maximum character length specified by the Windows operating system. The recommended location is the `C:\nxp` folder.

# MCUXpresso SDK Release Notes Supporting EVK-MCIMX7ULP

## Change Logs

# Contents

# Contents

# Contents

# 1 Driver Change Log

## ACMP

The current ACMP driver version is 2.0.4.

- 2.0.4
  - Bug fix:
    - Avoid change w1c bit in ACMP_SetRoundRobinPreState();
- 2.0.3
  - Added feature functions for different power domain's usage (1.8 V and 3 V). These functions are first enabled in ULP1. They are about:
    - ACMP_EnableLinkToDAC()
    - ACMP_SetDiscreteModeConfig()
    - ACMP_GetDefaultDiscreteModeConfig()
- 2.0.2
  - Coding style changes:
    - Changed coding style of peripheral base address from "s_acmpBases" to "s_acmpBase";
- 2.0.1
  - Bug fix:
    - Fixed bug regarding the function "ACMP_SetRoundRobinConfig". It will not continue execution but returns directly after disabling round robin mode;

## CACHE

The current CACHE driver version is 2.0.1.

- 2.0.1
  - Fixed the over 4 KB size maintenance issue in invalidate/clean/clean&invalidate by range APIs.
- 2.0.0
  - Initial version.

## CRC

The current CRC driver version is 2.0.1.

- 2.0.1
  - Bug fix:
    - DATA and DATALL macro definition moved from header file to source file.
- 2.0.0
  - Initial version.

## DAC12

The current DAC12 driver version is 2.0.0.

- 2.0.0
  - Initial version.

## DMAMUX

The current DMAMUX driver version is 2.0.2.

- 2.0.2
  - New feature:
    - Added an always-on enable feature to a DMA channel for ULP1 DMAMUX support.
- 2.0.1
  - Bug fix:
    - Fixed build warning while setting the DMA request source in DMAMUX_SetSource-Change issue by changing the type of the parameter source from uint8_t to uint32_t.
- 2.0.0
  - Initial version.

## EDMA

The current eDMA driver version is 2.1.5.

- 2.1.5
  - Improvements:
    - Improve EDMA IRQ handler to support half interrupt feature.
- 2.1.4
  - Bug fix:
    - Clear enabled request, status during EDMA_Init for the case that EDMA is halted before reinitialization.
- 2.1.3
  - Bug fix:
    - Add clear DONE bit in IRQ handler to avoid overwrite TCD issue.
    - Optimize above solution for the case that transfer request occurs in callback.
- 2.1.2
  - Improvements:
    - Added interface to get next TCD address.
    - Added interface to get the unused TCD number.
- 2.1.1
  - Improvements:
    - Added documentation for eDMA data flow when scatter/gather is implemented for the EDMA_HandleIRQ API.
    - Updated and corrected some related comments in the EDMA_HandleIRQ API and edma-

_handle_t struct.

- 2.1.0
  - Improvements:
    - Changed the EDMA_GetRemainingBytes API into EDMA_GetRemainingMajorLoopCount due to eDMA IP limitation (see API comments/note for further details).
- 2.0.5
  - Improvements:
    - Added pubweak DriverIRQHandler for K32H844P (16 channels shared).
- 2.0.4
  - Improvements:
    - Added support for SoCs with multiple eDMA instances.
    - Added pubweak DriverIRQHandler for KL28T DMA1 and MCIMX7U5_M4.
- 2.0.3
  - Bug fix:
    - Fixed the incorrect pubweak IRQHandler name issue, which causes re-definition build errors when client sets his/her own IRQHandler, by changing the 32-channel IRQHandler name to DriverIRQHandler.
- 2.0.2
  - Bug fix:
    - Fixed incorrect minorLoopBytes type definition in _edma_transfer_config struct, and defined minorLoopBytes as uint32_t instead of uint16_t.
- 2.0.1
  - Bug fix:
    - Fixed the eDMA callback issue (which did not check valid status) in EDMA_HandleIRQ API.
- 2.0.0
  - Initial version.

## EWM

The current EWM driver version is 2.0.1.

- 2.0.1
  - Fixed EWM_Deinit hardfault issue.
- 2.0.0
  - Initial version.

## FLEXIO

The current FLEXIO driver version is 2.0.2.

- 2.0.2:
  - Improvements:
    - Split FlexIO component which combines all flexio/flexio_uart/flexio_i2c/flexio_i2s

drivers into several components. FlexIO component, flexio_uart component, flexio_i2c_master component, and flexio_i2s component.

- 2.0.1
  - Bug fix:
    - Fix the Dozen mode configuration error in FLEXIO_Init API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.

## FLEXIO_UART

The current FLEXIO_UART driver version is 2.1.5.

- 2.1.5
  - Trigger user callback when all data are in ringbuffer are received when calling FLEXIO_UART_TransferReceiveNonBlocking.
- 2.1.4
  - Unify component full name to FLEXIO UART(DMA/EDMA) Driver
- 2.1.3
  - Bug fixes: The following modifications support FlexIO using multiple instances.
    - Removed FLEXIO_Reset API in module Init APIs.
    - Updated module Deinit APIs to reset the shifter/timer config instead of disable module and disable clock.
    - Updated module Enable APIs to only support enable operation.
- 2.1.2
  - Bug fixes:
    - Fixed the transfer count calculation issue in FLEXIO_UART_TransferGetReceiveCount, FLEXIO_UART_TransferGetSendCount, FLEXIO_UART_TransferGetReceiveCountDMA, FLEXIO_UART_TransferGetSendCountDMA, FLEXIO_UART_TransferGetReceiveCountEDMA and FLEXIO_UART_TransferGetSendCountEDMA
    - Fixed the Dozen mode configuration error in FLEXIO_UART_Init API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.
    - Reported error when set baudrate too low and FLEXIO cannot reach that baudrate.
    - Disabled FLEXIO_UART receive interrupt instead of disable all NVIC when read data from ring buffer. Because ring buffer is used, receive nonblocking disables all NVIC interrupts to protect the ring buffer. This has negative effects on other IPS which are using interrupt.
- 2.1.1
  - Bug fixes:
    - Changed the API name FLEXIO_UART_StopRingBuffer to FLEXIO_UART_TransferStopRingBuffer to align with the definition in C file.
- 2.1.0
  - New features:
    - Added Transfer prefix in transactional APIs.
    - Added txSize/rxSize in handle structure to record the transfer size.

  - Bug fixes:
    * Added error handle to handle the data count is zero or data buffer is NULL situation.

## FLEXIO_I2C

The current FLEXIO_I2C driver version is 2.1.6.

- 2.1.6
  - Bug fix:
    * Fixed the issue that I2C Master transfer APIs(blocking/non-blocking) does not support the situation that master transfer with subaddress and transfer data size zero, which means no data follows by the subaddress.
- 2.1.5
  - Unify component full name to FLEXIO I2C Driver
- 2.1.4
  - Bug fixes: The following modifications support FlexIO using multiple instances.
    * Removed FLEXIO_Reset API in module Init APIs.
    * Updated module Deinit APIs to reset the shifter/timer config instead of disable module and disable clock.
    * Updated module Enable APIs to only support enable operation.
- 2.1.3
  - Changed the prototype of FLEXIO_I2C_MasterInit to return kStatus_Success if initialization successfully and return kStatus_InvalidArgument if "(srcClock_Hz / masterConfig->baudRate_Bps) / 2 - 1" exceeds 0xFFU.
- 2.1.2
  - Fixed the FLEXIO I2C issue where the master cannot receive data from I2C slave in high baudrate.
  - Fixed the FLEXIO I2C issue where the master cannot receive NAK when master sends non-existent addr.
  - Fixed the FLEXIO I2C issue where the master cannot get transfer count successfully.
  - Fixed the FLEXIO I2C issue where the master cannot receive data successfully when sending data first.
  - Fixed the Dozen mode configuration error in FLEXIO_I2C_MasterInit API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.
  - Fixed the FLEXIO_I2C_MasterTransferBlocking API calls FLEXIO_I2C_MasterTransferCreateHandle issue. This leads the s_flexioHandle/s_flexioIsr/s_flexioType variable written. Then, if calling FLEXIO_I2C_MasterTransferBlocking API multiple times, the s_flexioHandle/s_flexioIsr/s_flexioType variable cannot be written anymore due to it being out of range. This leads to the following: NonBlocking transfer APIs cannot work due to register IRQ failed.
- 2.1.1
  - Bug fixes:
    * Implemented the FLEXIO_I2C_MasterTransferBlocking API which defined in header file but has no implementation in the C file.

- 2.1.0
  - New features:
    - Added Transfer prefix in transactional APIs.
    - Added transferSize in handle structure to record the transfer size.

## FLEXIO_SPI

The current FLEXIO_SPI driver version is 2.1.3.

- 2.1.3
  - Unify component full name to FLEXIO SPI(DMA/EDMA) Driver
- 2.1.2
  - Bug fixes: The following modification support FlexIO using multiple instances.
    - Removed FLEXIO_Reset API in module Init APIs.
    - Updated module Deinit APIs to reset the shifter/timer config instead of disable module and disable clock.
    - Updated module Enable APIs to only support enable operation.
- 2.1.1
  - Bug fixes:
    - Fixed bug where FLEXIO SPI transfer data is in 16 bit per frame mode with eDMA.
    - Fixed bug where FLEXIO SPI transfer data is in 16 bit per frame and direction is Lsbfirst mode with eDMA and interrupt.
    - Fixed the Dozen mode configuration error in FLEXIO_SPI_MasterInit/FLEXIO_SPI_SlaveInit API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.
  - Optimization:
    - Added #ifndef/#endif to allow user to change the default tx value at compile time.
- 2.1.0
  - New features:
    - Added Transfer prefix in transactional APIs.
    - Added transferSize in handle structure to record the transfer size.
  - Bug fixes:
    - Fixed the error register address return for 16-bit data write in FLEXIO_SPI_GetTxDataRegisterAddress.
    - Provided independent IRQHandler/transfer APIs for Master and slave to fix the baudrate limit issue.

## FLEXIO_I2S

The current FLEXIO_I2S driver version is 2.1.6.

- 2.1.6
  - Bug fix:
    - Add reset flexio before flexio i2s init to make sure flexio status is normal.

- 2.1.5
  - Bug fix:
    - Fix i2s driver use hard code for bitwidth setting.
- 2.1.4
  - Unify component full name to FLEXIO I2S(DMA/EDMA) Driver
- 2.1.3
  - Bug fixes: The following modifications support FlexIO using multiple instances.
    - Removed FLEXIO_Reset API in module Init APIs.
    - Updated module Deinit APIs to reset the shifter/timer config instead of disable module and disable clock.
    - Updated module Enable APIs to only support enable operation.
- 2.1.2
  - New features:
    - Added configure items for all pin polarity and data valid polarity.
    - Added default configure for pin polarity and data valid polarity.
- 2.1.1
  - Bug fixes:
    - Fixed FlexIO I2S RX data read error and eDMA address error.
    - Fix FlexIO I2S slave timer compare setting error.
- 2.1.0
  - New features:
    - Added Transfer prefix in transactional APIs.
    - Added transferSize in handle structure to record the transfer size.

## FLEXIO_MCU_LCD

The current FLEXIO_MCU_LCD driver version is 2.0.2.

- 2.0.2
  - Unify component full name to FLEXIO_MCU_LCD(EDMA) Driver
- 2.0.1
  - Bug fixes: The following modification to support FlexIO using multiple instances.
    - Removed FLEXIO_Reset API in module Init APIs.
    - Updated module Deinit APIs to reset the shifter/timer config instead of disable module and disable clock.
    - Updated module Enable APIs to only support enable operation.
- 2.0.0
  - Initial version.

## FLEXIO_CAMERA

The current FLEXIO_CAMERA driver version is 2.1.2.

- 2.1.2

  - Unify component full name to FLEXIO CAMERA(EDMA) Driver
- 2.1.1
  - Bug fixes: The following modifications support FlexIO using multiple instances.
    - Removed FLEXIO_Reset API in module Init APIs.
    - Updated module Deinit APIs to reset the shifter/timer config instead of disable module and disable clock.
    - Updated module Enable APIs to only support enable operation.
- 2.1.0
  - New features:
    - Added Transfer prefix in transactional APIs.

## GPIO

The current driver version is 2.3.1.

- 2.3.1:
  - Remove deprecated APIs.
- 2.3.0:
  - New feature:
    - Update the driver code to adapt the case of interrupt configurations in GPIO module. New APIs were added to configure the GPIO interrupt settings if the module has this feature on it.
- 2.2.1:
  - API interface changes:
    - Refined naming of API while keep all original APIs by marking them as deprecated. Original API will be removed in next release. The main change is update API with prefix of _PinXXX() and _PortXXX.
- 2.1.1:
  - API interface changes:
    - Added API for the check attribute bytes.
- 2.1.0:
  - API interface changes:
    - Added "pins" or "pin" to some APIs' names.
    - Renamed "_PinConfigure" to "GPIO_PinInit".

## LLWU

The current LLWU driver version is 2.0.2.

- 2.0.2
  - Optimization
    - correct driver function LLWU_SetResetPinMode parameter name.
- 2.0.1
  - Miscellaneous changes:

        * Updates for KL8x.
- 2.0.0
  - Initial version.

## LPADC

The current LPADC driver version is 2.0.1.

- 2.0.1
  - Ensure the API LPADC_SetConvCommandConfig configure related registers correctly.
- 2.0.0
  - Initial version.

## LPI2C CMSIS

The current LPI2C driver version is 2.1.5.

- 2.1.5
  - Bug fix:
    * Extended the Driver IRQ handler to support LPI2C4 and change to use ARRAY_SIZE(k-Lpi2cBases) instead of FEATURE COUNT to decide the array size for handle pointer array.
    * 2.1.4
  - Bug fix:
    * Fixed the LPI2C_MasterTransferEDMA receive issue when LPI2C share same request source for TX/RX DMA request. In the previous way the API uses scatter gather method, handle command transfer first, then handles the linked TCD which preset with the receive data transfer. The issue is that TX DMA request and RX DMA request are both enabled, when DMA finished the first command TCD transfer and handled the receive data TCD, the TX DMA request still happens due to TX FIFO empty. This results the RX DMA transfer starts, without waiting on the expected RX DMA request. Fix the issue by enabling IntMajor interrupt for the command TCD and checking if there is a linked TCD to disable the TX DMA request in LPI2C_MasterEDMACallback API.
- 2.1.3
  - Improvement:
    * Added LPI2C_WATI_TIMEOUT macro to allow the user to specify the timeout times for waiting flags in functional API and blocking transfer API.
    * Added LPI2C_MasterTransferBlocking API.
- 2.1.2
  - Bug fix:
    * In LPI2C_SlaveTransferHandleIRQ, reset the slave status to idle when stop flag is detected.
- 2.1.1
  - Bug fix:

        * Disabled auto stop feature in eDMA driver. Previously, the autostop feature was enabled at transfer when transferring with stop flag. If the previous transfer was without stop flag, because the auto stop feature is enabled, then when starting a new transfer with stop flag, the stop flag sends before starting the new transfer, and the start flag cannot successfully send, so the transfer cannot start.
        * Changed default slave configuration with address stall false.
- 2.1.0
  - API name change:
    * LPI2C_MasterTransferCreateHandle -> LPI2C_MasterCreateHandle.
    * LPI2C_MasterTransferGetCount -> LPI2C_MasterGetTransferCount.
    * LPI2C_MasterTransferAbort -> LPI2C_MasterAbortTransfer.
    * LPI2C_MasterTransferHandleIRQ -> LPI2C_MasterHandleInterrupt.
    * LPI2C_SlaveTransferCreateHandle -> LPI2C_SlaveCreateHandle.
    * LPI2C_SlaveTransferGetCount -> LPI2C_SlaveGetTransferCount.
    * LPI2C_SlaveTransferAbort -> LPI2C_SlaveAbortTransfer.
    * LPI2C_SlaveTransferHandleIRQ -> LPI2C_SlaveHandleInterrupt.
- 2.0.0
  - Initial version.

## LPIT

The current LPIT driver version is 2.0.0.

- 2.0.0
  - Initial version.

## LPSPI

The current LPSPI driver version is 2.0.2.

- 2.0.2
  - New feature:
    * Added dummy data setup API to allow users to configure the dummy data to be transferred.
    * Enabled the 3-wire mode, SIN and SOUT pins can be configured as input/output pin.
- 2.0.1
  - Bug fix:
    * The clock source should divided by PRESCALE setting in LPSPI_MasterSetDelayTimes function.
    * Fixed the bug that LPSPI_MasterTransferBlocking function would hang in some corner cases.
  - Optimization:
    * Added #ifndef/#endif to allow user to change the default TX value at compile time.
- 2.0.0

  - Initial version.

## LPTMR

The current LPTMR driver version is 2.0.1.

- 2.0.1
  - Driver update:
    - Updated the LPTMR driver to support 32-bit CNR and CMR registers in some devices.
- 2.0.0
  - Initial version.

## LPUART CMSIS

The current LPUART driver version is 2.2.6.

- 2.2.6
  - Fix the repeatedly reading status register issue while dealing with the IRQ routine.
- 2.2.5
  - Do not set or clear the TIE/RIE bits when using LPUART_EnableTxDMA() and LPUART_-EnableRxDMA().
- 2.2.4
  - Added hardware flow control function support.
  - Added idle line detected feature in LPUART_TransferNonBlocking function. If an idle line was detected, a callback is triggered with status kStatus_LPUART_IdleLineDetected returned. This feature may be useful when the received Bytes is less than the expected receive data size. Before triggering the callback, data in the FIFO (if has FIFO) is read out, and all interrupts will not be disabled, except if the receive data size reaches 0.
  - Enabled the RX FIFO watermark function. With the idle line detected feature enabled, you can set the watermark value to whatever you want (should be less than the RX FIFO size). Data is received and a callback is triggered when data receive is end.
- 2.2.3
  - Changed parameter type in LPUART_RTOS_Init() struct rtos_lpuart_config –> lpuart_rtos_-config_t.
  - Bug fix:
    - Disabled LPUART receive interrupt instead of disabling all NVIC when read data from ring buffer. Because the ring buffer is used, receive nonblocking disables all NVIC interrupts to protect the ring buffer. This has a negative effect to other IPS which are using the interrupt.
- 2.2.2
  - Added software reset feature support.
  - Added software reset API to LPUART_Init().
- 2.2.1
  - Added separate RX,TX IRQ number support.

- 2.2.0
  - Added 7 data bits and MSB support.
- 2.1.1
  - Removed needless check of event flags and assert in LPUART_RTOS_Receive.
  - Always wait for RX event flag in LPUART_RTOS_Receive.
- 2.1.0
  - Update transactional APIs.

## MU

The Current MU driver version is 2.0.2.

- 2.0.2
  - Added support for MIMX8MQx.
- 2.0.1
  - Added support for MCIMX7Ux_M4.
- 2.0.0
  - Initial version.

## MSMC

The current MSMC driver version is 2.1.0.

- 2.1.0
  - Added new APIs with FEATURE macros support: SMC_GetStopEntryStatus() SMC_ClearStopEntryStatus() SMC_SetForceBootOptionConfig() SMC_SRAMEnableLowPowerMode() SMC_SRAMEnableDeepSleepMode()
  - Updated APIs with FEATURE macros support: SMC_SetPowerModeStop() SMC_SetPowerModeVlpr() SMC_SetPowerModeLls() SMC_SetPowerModeVlls() SMC_ConfigureResetPinFilter()
- 2.0.0
  - Initial version.

## PMC0

The current PMC0 driver version is 2.0.0.

- 2.0.0
  - Initial version.

## PORT

The current PORT driver version is 2.1.0.

- 2.1.0
  - New feature:
    - Update the driver code to adapt the case of the interrupt configurations in GPIO module. will move the pin configuration APIs to GPIO module.
- 2.0.2
  - Miscellaneous changes:
    - Added feature guard macros in the driver.
- 2.0.1
  - Miscellaneous changes:
    - Added "const" in function parameter.
    - Updated some enumeration variables' names.

## QSPI

The current QSPI driver version is 2.0.3.

- 2.0.3
  - Add new API QSPI_SetDqsConfig for DQS configuration.
- 2.0.2
  - New Macro function:
    - Added QSPI_LUT_SEQ() function for users to set LUT table easily.
    - Added LUT command macros for users to easy use.
  - Comment update:
    - Added the comments for the limitation of QSPI_ReadBlocking and QSPI_TransferReceiveBlocking.
- 2.0.1
  - New API:
    - QSPI_SetReadArea to set the read area.
  - Bug fix:
    - Fixed QSPI_UpdateLUT function only update first LUT issue.
    - Fixed issue that some function that hardcode QSPI0 as base.
- 2.0.0
  - Initial version.

## SAI

The current SAI driver version is 2.1.7.

- 2.1.7
  - Improvement:
    - Add feature macro test for the mclkSource member in sai_config_t.
    - Change "FSL_FEATURE_SAI5_SAI6_SHARE_IRQ" to "FSL_FEATURE_SAI_SAI5_SAI6_SHARE_IRQ".
    - Add #ifndef #endif check for SAI_XFER_QUEUE_SIZE to allow redefinition.

  - Bug fix:
    * Fix the build error caused by feature macro test for mclkSource.

-2.1.6

- Improvement:
  - Add feature macro test for mclkSourceClockHz check.
  - Add bit clock source name for general devices.
- Bug fix:
  - Fix incorrect channel numbers setting while call RX/TX set format together.

-2.1.5

- Bug fix:
  - Correct SAI3 driver IRQ handler name.
  - Add I2S4/5/6 IRQ handler.
  - Add base in handler structure to support different instances share one IRQ number.
- New feature:
  - Update SAI driver for MCR bit MICS.
  - Added 192KHZ/384KHZ in the sample rate enumeration.
  - Added multi FIFO interrupt/sdma transfer support for TX/RX.
  - Added API to read/write multi FIFO data in a blocking method.
  - Added bclk bypass support when bclk is same with mclk.

2.1.4

- New feature:
  - Added API to enable/disable auto FIFO error recovery in platforms that support this feature.
  - Added API to set data packing feature in platform which support this feature.

2.1.3

- New feature:
  - Added feature to make I2S frame sync length configurable according to bitWidth.

2.1.2

- Bug fix:
  - Added 24-bit support for SAI eDMA transfer. All data shall be 32 bits for send/receive, as eDMA cannot directly handle 3 Byte transfer.

2.1.1

- Optimization:
  - Reduced code size while not using transactional API.

2.1.0

- API name change:
  - SAI_GetSendRemainingBytes -> SAI_GetSentCount.
  - SAI_GetReceiveRemainingBytes -> SAI_GetReceivedCount.
  - All transactional API name add "Transfer" prefix.
  - All transactional API use base and handle as input parameter.

  - Unify the parameter names.
- Bug fix:
  - Fixed WLC bug while reading TCSR/RCSR registers.
  - Fixed MOE enable flow issue, move MOE enable after MICS settings in SAI_TxInit/SAI_RxInit.

2.0.0

- Initial version.

## SEMA42

The current SEMA42 driver version is 2.0.0.

- 2.0.0
  - Initial version.

## SMC

The current SMC driver version is 2.0.4.

- 2.0.4
  - When entering stop modes, use RAM function for the flash synchronize issue. Application should make sure that, the rw data of fsl_smc.c is located in memory region which is not powered off in stop modes.
- 2.0.3
  - Added APIs SMC_PreEnterStopModes, SMC_PreEnterWaitModes, SMC_PostExitWaitModes, and SMC_PostExitStopModes.
- 2.0.2
  - Bug fix:
    - Added DSB before WFI, add ISB after WFI.
  - Miscellaneous changes:
    - Updated SMC_SetPowerModeVlpw implementation.
- 2.0.1
  - Miscellaneous changes:
    - Updated for KL8x.
- 2.0.0
  - Initial version.

## TPM

The current TPM driver version is 2.0.2.

- 2.0.2
  - Bug fixes:

        * Fixed issues in functions TPM_SetupPwm/TPM_UpdateChnlEdgeLevelSelect /TPM_SetupInputCapture/TPM_SetupOutputCompare/TPM_SetupDualEdgeCapture, wait acknowledgement when channel disabled.
- 2.0.1
  - Bug fixes:
    * Fix TPM_UpdateChnIEdgeLevelSelect ACK wait issue.
    * Fix TPM_SetupdualEdgeCapture can not set FILTER register issue.
    * Fix TPM_UpdateChnEdgeLevelSelect ACK wait issue.
- 2.0.0
  - Initial version.

## TRGMUX

The current TRGMUX driver version is 2.0.0.

- 2.0.0
  - Initial version.

## TSTMR

The current TSTMR driver version is 2.0.0.

- 2.0.0
  - Initial version.

## WDOG32

The current WDOG32 driver version is 2.0.1.

- 2.0.1
  - Bug fixes:
    * WDOG must be configured within its configuration time period
      · Added WDOG32_Init API to quick access section.
      · Defined register variable in WDOG32_Init API.
- 2.0.0
  - Initial version.

## XRDC

The current XRDC driver version is 2.0.3.

- 2.0.3
  - Updates:

        * Added necessary driver supports for K32H844P.
        * Added new APIs concerning new features of Exclusive Access Lock and domain programmable access flags configurations.
- 2.0.2
  - Bug fixes:
    * Fixed wrong assert of assignIndex input check in the xRDC driver.
  - Improvements:
    * Added master input CPU/non-CPU check in XRDC_SetNonProcessorDomainAssignment and XRDC_SetProcessorDomainAssignment API.
    * Added necessary assert checks for several config inputs.
- 2.0.1
  - Improvements:
    * Changed reserved bit fields in the structs into unnamed-identifier bit fields.
- 2.0.0
  - Initial version.

## CLOCK

Current CLOCK driver version is 2.1.1

- 2.1.1
  - Improvement:
    * Change reserved bit fields in _scg_sys_clk_config struct into unnamed bit fields.
- 2.1.0
  - Other Changes:
    * Merge fsl_scg and fsl_osc into fsl_clock.
- 2.0.0
  - Initial version.

## IOMUXC

The current IOMUXC driver version is 2.0.0.

- 2.0.0
  - initial version.

## LPI2C CMSIS

Current LPI2C CMSIS driver version is 2.0

- 2.0
  - Initial version.

## LPSPI_CMSIS

Current lpspi_cmsis driver version is 2.1

- 2.1
  - Bug Fix:
    - Fix the incorrect clock polarity assignment in the driver. For ARM_SPI_CPOL0_CPHA0 and other frame format parameters, CPOL = 0 means kSPI_ClockPolarityActiveHigh not kSPI_ClockPolarityActiveLow in driver.
  - New feature:
    - Allow user to set up the default Transmit value by using ARM_SPI_SET_DEFAULT_TX_VALUE. Please note that set default value is not supported in slave interrupts, because the pin will stay tristated if TX buffer is NULL.
    - Enable slave select mode in the new driver, but this has no effect when users set any of them because the driver can only support the hardware control function.
    - Enable 3-Wire mode, users can use ARM_SPI_MODE_MASTER_SIMPLEX/ARM_SPI_MODE_SLAVE_SIMPLEX to enable this feature. For ARM_SPI_MODE_MASTER_SIMPLEX mode, select SOUT pin as the input/output pin, and for ARM_SPI_MODE_SLAVE_SIMPLEX, the SIN pin is selected as the input/output pin.
- 2.0
  - - Initial version.

## LPUART CMSIS

Current LPUART CMSIS driver version is 2.0

- 2.0
  - Initial version.

# 2 Middleware Change Log

## mbedTLS for MCUXpresso SDK

The current version of mbedTLS is based on mbedTLS 2.13.1 released 2018-09-06

- 2.13.1_rev1
  - Bug fixes:
    - Fixed incorrect macro check when skipping AES-192 or AES-256
- 2.13.1
  - New features:
    - Ported mbedTLS 2.13.1 to MCUXpresso SDK.
- 2.12.0_rev1
  - New features:
    - Added support for NIST P-256 elliptic curve with CASPER driver.
- 2.12.0
  - New features:
    - Ported mbedTLS 2.12.0 to MCUXpresso SDK.
- 2.9.0_rev2
  - New features:
    - Added support for Hashcrypt driver.
- 2.9.0_rev1
  - New features:
    - Added support for CASPER driver.
- 2.9.0
  - New features:
    - Ported mbedTLS 2.9.0 to MCUXpresso SDK.
- 2.6.0_rev2
  - Bug fixes:
    - ssl_cookie.c now uses SHA256 for COOKIE_MD (instead of original SHA224). Some hw crypto acceleration (such as CAU3) don't support SHA224 but all support SHA256.
- 2.6.0_rev1
  - Bug fixes:
    - ksdk_mbedtls.c bignum functions now read sign of input mbedtls_mpi at beginning of functions to properly support in place computations (when output bignum is the same as one of input bignums). Affected functions: mbedtls_mpi_mul_mpi(), mbedtls_mpi_mod_mpi(), ecp_mul_comb().
- 2.6.0
  - New features:
    - Ported mbedTLS 2.6.0 to MCUXpresso SDK.
    - Added MBEDTLS_FREESCALE_FREERTOS_CALLOC_ALT to allow alternate implementation of pvPortCalloc() when using .c.
- 2.5.1_rev1
  - New features:

        - Added support for DCP driver.
- 2.5.1
  - New features:
    - Ported mbedTLS 2.5.1 to MCUXpresso SDK.
- 2.4.2_rev2
  - New features:
    - Added Curve25519 support for CAU3.
    - Added MBEDTLS_ECP_MUL_MXZ_ALT configuration parameter enabling overloading of ecp_mul_mxz().
- 2.4.2_rev1
  - New features:
    - Added support for CAU3 driver.
    - Added new files:
    - .c - contains regular software implementation of DES algorithm with added MBEDTLS_DES3_SETKEY_DEC_ALT and MBEDTLS_DES3_SETKEY_ENC_ALT config parameters.
    - .h - contains modified mbedtls_des_context and mbedtls_des3_context structures.
    - Added MBEDTLS_DES3_SETKEY_DEC_ALT configuration parameter enabling reloading of mbedtls_des3_set2key_dec() and mbedtls_des3_set3key_dec().
    - Added MBEDTLS_DES3_SETKEY_ENC_ALT configuration parameter enabling reloading of mbedtls_des3_set2key_enc() and mbedtls_des3_set3key_enc().
- 2.4.2
  - New features:
    - Ported mbedTLS 2.4.2 to KSDK 2.0.0.
    - Added CRYPTO_InitHardware() function.
    - Added new file:
      - .h - contains declaration of CRYPTO_InitHardware() function and should be included in applications.
- 2.3.0_rev1
  - New features:
    - Added support for CAAM driver.
    - In LTC-specific wrapper, allocate temporary integers from heap in one large block.
- 2.3.0
  - New features:
    - Ported mbedTLS 2.3.0 to KSDK 2.0.0.

2.2.1

- New features:
  - Ported mbedTLS 2.2.1 to KSDK 2.0.0.
  - Added support of MMCAU cryptographic acceleration module. Accelerated MD5, SHA, AES, and DES.
  - Added support of LTC cryptographic acceleration module. Accelerated AES, DES, and PKHA.
  - Added new files:
  - .c - alternative implementation of cryptographic algorithm functions using LTC and MMCAU

module drivers.
- .h - configuration settings used by mbedTLS KSDK bare metal examples.
- Added mbedTLS KSDK bare-metal examples:
  - <board name> - KSDK mbedTLS benchmark application.
  - <board name> - KSDK mbedTLS self-test application.
- Added MBEDTLS_GCM_CRYPT_ALT configuration parameter enabling reloading of mbedtls_gcm_crypt_and_tag().
- Added MBEDTLS_ECP_MUL_COMB_ALT to enable alternate implementation of ecp_mul-_comb().
- Added MBEDTLS_ECP_ADD_ALT configuration parameter enabling reloading of ecp_-add().
- Added MBEDTLS_DES_SETKEY_DEC_ALT configuration parameter enabling reloading of mbedtls_des_setkey_dec(), mbedtls_des3_set2key_dec() and mbedtls_des3_set3key_dec().
- Added MBEDTLS_DES_SETKEY_ENC_ALT configuration parameter enabling reloading of mbedtls_des_setkey_enc(), mbedtls_des3_set2key_enc() and mbedtls_des3_set3key_enc().
- Added MBEDTLS_DES_CRYPT_CBC_ALT configuration parameter enabling reloading of mbedtls_des_crypt_cbc().
- Added MBEDTLS_DES3_CRYPT_CBC_ALT configuration parameter enabling reloading of mbedtls_des3_crypt_cbc().
- Added MBEDTLS_AES_CRYPT_CBC_ALT configuration parameter enabling reloading of mbedtls_aes_crypt_cbc().
- Added MBEDTLS_AES_CRYPT_CTR_ALT configuration parameter enabling reloading of mbedtls_aes_crypt_ctr().
- Added MBEDTLS_CCM_CRYPT_ALT configuration parameter enabling reloading of mbedtls_ccm_encrypt_and_tag() and mbedtls_ccm_auth_decrypt().
- Added MBEDTLS_MPI_ADD_ABS_ALT configuration parameter enabling reloading of mbedtls_mpi_add_abs().
- Added MBEDTLS_MPI_SUB_ABS_ALT configuration parameter enabling reloading of mbedtls_mpi_sub_abs().
- Added MBEDTLS_MPI_EXP_MOD_ALT configuration parameter enabling reloading of mbedtls_mpi_exp_mod().
- Added MBEDTLS_MPI_MUL_MPI_ALT configuration parameter enabling reloading of mbedtls_mpi_mul_mpi().
- Added MBEDTLS_MPI_MOD_MPI_ALT configuration parameter enabling reloading of mbedtls_mpi_mod_mpi().
- Added MBEDTLS_MPI_GCD_ALT configuration parameter enabling reloading of mbedtls-_mpi_gcd().
- Added MBEDTLS_MPI_INV_MOD_ALT configuration parameter enabling reloading of mbedtls_mpi_inv_mod().
- Added MBEDTLS_MPI_IS_PRIME_ALT configuration parameter enabling reloading of mbedtls_mpi_is_prime().
- Added encrypt/decrypt mode to mbedtls_des_context and mbedtls_des3_context structure.
- Added carriage return ” for mbedtls_printf() in self test functions.

## Multicore SDK

The current version of Multicore SDK is 2.5.0.

- 2.5.0
  - Multicore SDK component versions:
    - embedded Remote Procedure Call (eRPC) v1.7.1
    - eRPC generator (erpcgen) v.1.7.1
    - Multicore Manager (MCMgr) v4.0.2
    - RPMsg-Lite v2.0.2
  - New features:
    - RPMsg-Lite, MCMgr: Align porting layers to the updated MCUXpressoSDK feature files.
    - eRPC: Fixed semaphore in static message buffer factory.
    - erpcgen: Fixed MU received error flag.
    - erpcgen: Fixed tcp transport.
- 2.4.0
  - Multicore SDK component versions:
    - embedded Remote Procedure Call (eRPC) v1.7.0
    - eRPC generator (erpcgen) v.1.7.0
    - Multicore Manager (MCMgr) v4.0.1
    - RPMsg-Lite v2.0.1
  - New features:
    - eRPC: Improved code size of generated code.
    - eRPC: Generating CRC value is optional.
    - eRPC: Fixed CMSIS UART driver. Removed dependency on MCUXpresso SDK.
    - eRPC: List names are based on their types. Names are more deterministic.
    - eRPC: Service objects are as a default created as global static objects.
    - eRPC: Added missing doxygen comments.
    - eRPC: Forbid users use reserved words.
    - eRPC: Removed outByref for function parameters.
    - eRPC: Added support for 64bit numbers.
    - eRPC: Added support of program language specific annotations.
    - eRPC: Optimized code style of callback functions.
    - RPMsg-Lite: New API rpmsg_queue_get_current_size()
    - RPMsg-Lite: Fixed bug in interrupt handling for lpc5411x, lpc5410x
    - RPMsg-Lite: Code adjustments based on static analysis tool findings
- 2.3.1
  - Multicore SDK component versions:
    - embedded Remote Procedure Call (eRPC) v1.6.0
    - eRPC generator (erpcgen) v.1.6.0
    - Multicore Manager (MCMgr) v4.0.0
    - RPMsg-Lite v1.2.0
  - New features:
    - eRPC: Improved code size of generated code.
    - eRPC: Improved eRPC nested calls.
    - eRPC: Improved eRPC list length variable serialization.

        - eRPC: Added support for scalar types.
        - MCMgr: Added new MCMGR_TriggerEventForce() API.
- 2.3.0
  - Multicore SDK component versions:
    - embedded Remote Procedure Call (eRPC) v1.5.0
    - eRPC generator (erpcgen) v.1.5.0
    - Multicore Manager (MCMgr) v3.0.0
    - RPMsg-Lite v1.2.0
  - New features:
    - eRPC: Added support for unions type non-wrapped by structure.
    - eRPC: Added callbacks support.
    - eRPC: Added support  annotation for functions.
    - eRPC: Added support

## wolfSSL

The current version is 3.9.8_rev3, based on Release 3.9.8 of wolfSSL.

- 3.9.8_rev3
  - New features:
    - Added support for DCP driver.
- 3.9.8_rev2
  - New features:
    - Added support for CAU3 driver.
- 3.9.8_rev1
  - New features:
    - Added support for CAAM driver.
    - Added FREESCALE_ALT macros.
- 3.9.8
  - New features:
    - Added support for AES and SHA acceleration modules of LPC devices. Accelerates AES and SHA wolfSSL modules.
    - LTC acceleration for AES CBC now updates IV.
  - Bug fixes:
    - Fixed K8x/KL8x LTC RSA sign when FREESCALE_LTC_TFM_RSA_4096_ENABLE macro is enabled.
- 3.9.0
  - New features:
    - Added more LTC public key acceleration (curve25519, ed25519 and RSA4096).
    - FREESCALE_LTC_TFM_RSA_4096_ENABLE macro added to enable RSA4096 on K8x/KL8x LTC.
    - LTC_MAX_ECC_BITS increased to 384 to enable ECC-384 curve acceleration on LTC.
    - FREESCALE_LTC_SHA added for KL8x SHA-1 and SHA-256 hardware acceleration.
  - Other changes:

    - wolfSSL/wolfcrypt/settings.h is changed to remove unused macros and add support for KSDK 2.0.
    - LTC public key acceleration is implemented in separate source file ksdk_port.h and ksdk_port.c
- 3.8.0
  - New features:
    - Added support for LTC hardware acceleration module. Accelerates AES, 3DES, TFM module (modular integer arithmetic) and ECC wolfSSL modules.
    - Added support for random number generator modules TRNG and RNGA.
  - Other changes:
    - The MMCAU acceleration now uses "fsl_mmcau.h" instead of "cau_api.h".
    - In DSA, wc_dsaSign() changed to repeate wc_RNG_GenerateBlock() until k is less than q.
    - wolfSSL/wolfcrypt/settings.h is changed to remove unused macros and add support for KSDK 2.0.
    - In wolfcrypt/src/asn.c, ksdk_time(time_t) changed to extern, to be defined by application.

# 3 RTOS Change Log

## FreeRTOS for MCUXpresso SDK.

The current version is Amazon-FreeRTOS 1.4.0 Original package is available at github.-com/aws/amazon-freertos.

- 1.4.0
  - New features:
    - Support for CM33, CM33F architectures based on CM3, CM4F ports
    - Support for pkcs11 for several platforms, secure element host library under pkcs11/portable/nxp folder
    - Lwip, wifi_qca support for secure_sockets in secure_sockets/portable/nxp folder
    - Flash driver support for several platforms in third_party/mcu_vendor/nxp folder
    - Generic support for aws_wifi under wifi/portable/nxp/common folder
  - Other changes:
    - Fix several build warnings, errors

***How to Reach Us:***

**Home Page:**
nxp.com

**Web Support:**
nxp.com/support



Document Number MCUXSDKIMX7ULPRN
Revision 0, 01/2019