NXP Semiconductors
Release Notes

Document Number: MCUXSDKK28FRN
Rev. 0, 05/2018

MCUXpresso SDK Release Notes Supporting FRDM-K28F

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 USB and lwIP, integration with WolfSSL and mbed TLS cryptography libraries, other middleware packages, such as multicore support and FatFs, 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, MCUXpressoSDK is the evolution of Kinetis SDK v2.3.0, 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, a new 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 FSL 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 8.22.2
- MDK-Arm Microcontroller Development Kit (Keil)® 5.24a
- Makefiles support with GCC revision 7-2017-q4-major from Arm Embedded
- MCUXpresso IDE v10.2.0

4 Supported development systems

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

5 Release contents

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

Table 1. 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
USB Documentation	<install_dir>/docs/usb
lwIP Documentation	<install_dir>/docs/lwip
Middleware	<install_dir>/middleware
lwIP stack	<install_dir>/middleware/lwip

Table continues on the next page...

Table 1. Release contents (continued)

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.

6.2 Middleware

6.2.1 USB stack

See the *MCUXpresso SDK USB Stack User's Guide* (document MCUXSDKUSBSUG) for more information.

NOTE

To enable USB High Speed function, add jumper on J14 and remove jumper on J53. To enable USB Full Speed function,add jumper on J53.

6.2.1.1 Peripheral devices tested with the USB Host stack

This table provides a list of USB devices tested with the USB Host stack.

Table 2. Peripheral devices

Device type	Device
USB HUB	BELKIN F5U233
	BELKIN F5U304
	BELKIN F5U307
	BELKIN F4U040
	UNITEK Y-2151
	Z-TEK ZK032A
	HYUNDAI HY-HB608
USB flash drive	ADATA C008 32 GB
	ADATA S102 8 G
	ADATA S102 16 G
	Verbatim STORE N GO USB Device 8 G
	Kingston DataTraveler DT101 G2
	SanDisk Cruzer Blade 8 GB
	Unisplendour 1 G
	Imation 2 GB

Table continues on the next page...

Table 2. Peripheral devices (continued)

	V-mux 2 GB Sanmina-SCI 128 M Corporate Express 1 G TOSHIBA THUHYBS-008G 8 G Transcend JF700 8 G Netac U903 16 G SSK SFD205 8 GB Rex 4 GB SAMSUNG USB3.0 16GB
USB card reader/adapter	SSK TF adapter Kawau Multi Card Reader Kawau TF adapter Kawau SDHC card
USB Mouse	DELL MS111-P DELL M066U0A DELL MUAVDEL8 TARGUS AMU76AP DELL MD56U0 DELL MS111-T RAPOO M110
USB Keyboard	DELL SK8135 DELL SK8115

6.2.2 TCP/IP stack

The lwIP TCP/IP stack is pre-integrated with MCUXpresso SDK and runs on top of the MCUXpresso SDK Ethernet driver with Ethernet-capable devices/boards. For details, see the *lwIP TCPIP Stack and MCUXpresso SDK Integration User's Guide* (document MCUXSDKLWIPUG).

6.2.3 File system

The FatFs file system is integrated with MCUXpresso SDK and can be used to access either the SD card or the USB memory stick when the SD card driver or the USB Mass Storage Device class implementation is used.

6.2.4 RTOS

The MCUXpresso SDK is integrated with FreeRTOS OS.

6.2.5 CMSIS

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

7 MISRA compliance

All MCUXpresso SDK drivers and USB stack comply to MISRA 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.

Table continues on the next page...

Table 3. MISRA exceptions (continued)

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.
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 17.7	The feature of <stdarg.h> shall not be used.

Table continues on the next page...

Table 3. MISRA exceptions (continued)

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.

8.2 USBFS controller issue

Because of the USBFS controller design issues, the USB host suspend/resume demos (usb_suspend_resume_host_hid_mouse) of the full speed controller do not support the low speed device directly.

8.3 USB PID issue

Because the PID of all USB device examples is updated, uninstall the device drivers and then reinstall when the device (with new PID) is plugged in the first time.

MCUXpresso SDK Release Notes Supporting FRDM-K28F

Change Logs

Contents

Contents

Contents

1 Driver Change Log

ADC16

The current ADC16 driver version is 2.0.0.

- 2.0.0
  - Initial version

CMP

The current CMP driver version is 2.0.0.

- 2.0.0
  - Initial version.

CMT

The current CMT driver version is 2.0.1.

- 2.0.1
  - Miscellaneous changes:
    - Added static to global CMT variables.
- 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.

DAC

The current DAC driver version is 2.0.1.

- 2.0.1
  - Bug fix:
    - Moved the default DAC_Enable(..., true) from DAC_Init() to the application code so users

can enable the DAC's output.

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.

DSPI

The current dspi driver version is 2.2.0.

- 2.2.0
  - New features:
    - Added gasket feature for SPI EDMA driver, which reduces one channel used in the EDMA master transfer. With this feature support, only two channels are needed. For example, if the gasket feature is supported, we could use the DSPI_MasterTransfer-CreateHandleEDMA function like below: DSPI_MasterTransferCreateHandleEDMA(E-XAMPLE_DSPI_MASTER_BASEADDR, &g_dspi_edma_m_handle, DSPI_Master-UserCallback, &userData, &dspiEdmaMasterRxRegToRxDataHandle, NULL, &dspi-EdmaMasterIntermediaryToTxRegHandle);
    - Added dummy data setup API to allow users to configure the dummy data to be transferred.
    - Added new APIs for half-duplex transfer function. Users can send and receive data by one API in the polling/interrupt/EDMA way, and users can choose to either transmit first or receive first. Additionally, the PCS pin can be configured as assert status in transmission (between transmit and receive) by setting the isPcsAssertInTransfer to true.
- 2.1.4
  - Bug fix:
    - DSPI EDMA driver: The DSPI instance that has separated so the DMA request source can now transfer up to 32767 Bytes data in one DSPI_MasterTransferEDMA() transfer.
- 2.1.3
  - Bug fix:

        * DSPI EDMA driver can no longer support the case that the transfer data size is odd, but the bitsPerFrame is greater than 8.
    - Optimization:
        * Added #ifndef/#endif to allow users to change the default TX value at compile time.
- 2.1.2
    - Bug fix:
        * DSPI_MasterTransferBlocking function would hang in some corner cases (for example, some cases with bitsPerFrame is 4,6 and kDSPI_MasterPcsContinuous transfer mode).
- 2.1.1
    - Bug fix:
        * Set the EOQ (End Of Queue) bit to TRUE for the last transfer in transactional APIs.
- 2.1.0
    - New features:
        * Added Transfer prefix in transactional APIs.

EDMA

The current eDMA driver version is 2.1.2.

- 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 wrong 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.

FLASH

The current FLASH driver version is 2.3.1.

- 2.3.1
  - Bug fixes:
    - Unified Flash IFR design from K3.
    - New encoding rule for K3 flash size.
- 2.3.0
  - New features:
    - Added support for device with LP flash (K3S/G).
    - Added flash prefetch speculation APIs.
  - Improvements:
    - Refined flash_cache_clear function.
    - Reorganized the member of flash_config_t struct.
- 2.2.0
  - New features:
    - Supports FTFL device in FLASH_Swap API.
    - Supports various pflash start addresses.
    - Added support for KV58 in cache clear function.
    - Added support for device with secondary flash (KW40).
  - Bug fixes:
    - Compiled execute-in-ram functions as PIC binary code for driver use.

        * Added missed flexram properties.
        * Fixed unaligned variable issue for execute-in-ram function code array.
- 2.1.0
  - Improvements:
    * Updated coding style to align with KSDK 2.0.
    * Different alignment size support for pflash and flexnvm.
    * Improved the implementation of execute-in-ram functions.
- 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 cmponents. 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 shoud be 0; for enableInDoze = false, the configuration should be 1.

FLEXIO_UART

The current FLEXIO_UART driver version is 2.1.4.

- 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_TransferGetReceiveCount-DMA, FLEXIO_UART_TransferGetSendCountDMA, FLEXIO_UART_TransferGet-ReceiveCountEDMA and FLEXIO_UART_TransferGetSendCountEDMA
    * Fixed the Dozen mode configuration error in FLEXIO_UART_Init API. For enableIn-Doze = 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 2.1.5.

- 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_MasterTransfer-

CreateHandle 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 shoud 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 2.1.4.

- 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.

FTM

The current FTM driver version is 2.0.4.

- 2.0.4
  - Features:
    - Added to enable DMA transfer with new API:
      - FTM_EnableDmaTransfer()
- 2.0.3
  - Bug fixes:
    - Updated the FTM driver to enable fault input after configuring polarity.
- 2.0.2
  - Features:
    - Added support to Quad Decoder feature with new APIs:
      - FTM_GetQuadDecoderFlags()
      - FTM_SetQuadDecoderModuloValue()
      - FTM_GetQuadDecoderCounterValue()
      - FTM_ClearQuadDecoderCounterValue()
- 2.0.1
  - Bug fixes:
    - Updated the FTM driver to fix write to ELSA and ELSB bits.
    - FTM combine mode: set the COMBINE bit before writing to CnV register.
- 2.0.0
  - Initial version.

GPIO

The current driver version is 2.2.1.

- 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".

I2C

The current I2C driver version is 2.0.5.

- 2.0.5
  - Improvements:
    * Added I2C_WATI_TIMEOUT macro to allow the user to specify the timeout times for waiting flags in functional API and blocking transfer API.
- 2.0.4
  - Bug fixes:
    * Added proper handle for tranfer config flag kI2C_TransferNoStartFlag to support transmit with kI2C_TransferNoStartFlag flag. Only supports write only or write+read with no start flag, does not support read only with no start flag.
- 2.0.3
  - Bug fixes:
    * Removed enableHighDrive member in the master/slave configuration structure because the operation to HDRS bit is useless, user needs to use DSE bit in port register to configure the high drive capability.
    * Added reset registers operation in I2C_MasterInit and I2C_SlaveInit APIs. Fixed issue where I2C could not switch between master and slave mode.
    * Improved slave IRQ handler to handle the corner case that stop flag and address match flag come synchronously.
- 2.0.2
  - Bug fixes:
    * Fixed issue in master receive and slave transmit mode with no stop flag. The master could not succeed to start next transfer because the master could not send out re-start signal.
    * Fixed data transfer out of order issue due to memory barrier
    * Added hold time configuration for slave. By leaving the SCL divider and MULT reset values when configure to slave mode, the setup and hold time of the slave is then reduced outside of spec for lower baudrates. This can cause intermittent arbitration loss on the master side.

  - New features:
    - Added address nak event for master.
    - Added general call event for slave.
- 2.0.1
  - New features:
    - Added double buffer enable configuration for Socs which have the DFEN bit in S2 register.
    - Added flexible transmit/receive buffer size support in I2C_SlaveHandleIRQ.
    - Added start flag clear, address match, and release bus operation in I2C_SlaveWrite/Read-Blocking API.
  - Bug fix:
    - Changed the kI2C_SlaveRepeatedStartEvent to kI2C_SlaveStartEvent.

LLWU

The current LLWU driver version is 2.0.1.

- 2.0.1
  - Miscellaneous changes:
    - Updates for KL8x.
- 2.0.0
  - Initial version.

LMEM

The current LMEM driver version is 2.1.0.

- 2.1.0
  - Removed the write buffer enable from the cache enable API.
  - Added Enable write buffer APIs.
- 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

The current LPUART driver version is 2.2.5.

- 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 seprarte 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.

PDB

The current PDB driver version is 2.0.1.

- 2.0.1
  - Changed PDB register base array to const.
- 2.0.0
  - Initial version.

PIT

The current PIT driver version is 2.0.0.

- 2.0.0
  - Initial version.

PMC

The current PMC driver version is 2.0.0.

- 2.0.0
  - Initial version.

PORT

The current PORT driver version is 2.0.2.

- 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 2.0.2.

- 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_Transfer-

ReceiveBlocking.

- 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.

RCM

The current RCM driver version is 2.0.1.

- 2.0.1
  - [KPSDK-10249] Fixed kRCM_SourceSw bit shift issue.
- 2.0.0
  - Initial version.

RTC

The current RTC driver version is 2.0.0.

- 2.0.0
  - Initial version.

SAI

The current SAI driver version is 2.1.4.

-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:
  - Reduceed code size while not using transactional API.

2.1.0

- API name change:
  - SAI_GetSendRemainingBytes -> SAI_GetSentCount.
  - SAI_GetReceiveRemainingBytes -> SAI_GetReceivedCount.
  - All transcational 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.
  - Fixd MOE enable flow issue, move MOE enable after MICS settings in SAI_TxInit/SAI_RxInit.

2.0.0

- Initial version.

SDHC

The current SDHC driver version is 2.1.7.

- 2.1.7
  - Bug fixes:
    - Fixed ADMA1 descriptor configuration error.
    - Improved set clock function to check the output frequency range.
- 2.1.6 -New features:
  - Added SDHC_CardDetectByData3 API to support detect card through DATA3.
  - Added host base address/user data parameter for all call back function.
- 2.1.5 -New features:
  - Added NON-WORD align data addr transfer support in DMA mode.
- 2.1.4
  - New features:
    - Added response error flag to check response once read from the card.
  - Bug fixes:
    - Fix clock divider calucate not correct issue.
- 2.1.3
  - Modified some definition to be compatible with middleware adapter.
- 2.1.2
  - Bug fix:
    - Used function pointer for interrupt handler to reduce code size.
    - Bad status bit check behaviour when wait for initialization of SD card.
    - Added support NON-WORD aligned data size transfer mode for SDIO card.

- 2.1.1
  - Bug fix:
    - Fixed the compile error when ADMA1 is enabled.
- 2.1.0
  - New features:
    - Added a host descriptor to contain SDHC related attributes.
  - Bug fix:
    - Removed clock auto gated function because of that it is a hardware issue.
  - Other changes:
    - Added more SDIO card related command type.
    - Changed the callback mechanism in the non-blocking transaction API.
    - Merged the two ADMA configuration function to be one.
    - Changed the transaction API's name.

SDRAMC

The current SDRAMC driver version is 2.1.0.

- 2.1.0
  - API change:
    - Changed status_t SDRAMC_SendCommand() to void SDRAMC_SendCommand().
- 2.0.1
  - Miscellaneous changes:
    - Added static to the global sdramc variables.
- 2.0.0
  - Initial version.

SIM

The current SIM driver version is 2.1.0.

- 2.1.0
  - Added new APIs of SIM_GetRfAddr() and SIM_EnableSystickClock().
- 2.0.0
  - Initial version.

SMC

The current SMC driver version is 2.0.3.

- 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.

SYSMPU

The current SYSMPU driver version is 2.2.1.

- 2.2.1
  - Fixed MISRA issue.
- 2.2.0
  - Renamed MPU to SYSMPU.
  - Changed macro definition for slave number and fix the get error status calculation.
- 2.1.1
  - Added the feature file macro definition limitation for the MPU_SetRegionRwMasterAccessRights().
- 2.1.0
  - API changes:
    * Changed the mpu_region_num_t and mpu_master_t to uint32_t.
    * Changed the mpu_low_masters_access_rights_t, mpu_high_masters_access_rights_t to mpu_rwxrights_master_access_control_t, mpu_rwrights_master_access_control_t.
    * Changed the MPU_SetRegionLowMasterAccessRights(), MPU_SetRegionHighMasterAccessRights() to MPU_SetRegionRwxMasterAccessRights(), MPU_SetRegionRwMasterAccessRights().
- 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.

TSI_V2

The current TSI_V2 driver version is 2.1.2.

- 2.1.2
  - Bug fixes:
    * Fixed w1c issues in status handling API.
- 2.1.1
  - New features:
    * Changed void TSI_DeInit(TSI_Type *base) to void TSI_Deinit(TSI_Type *base).
- 2.0.1
  - Other changes:
    * Changed default configuration structure member order.

TSI_V4

The current TSI_V4 driver version is 2.1.2.

- 2.1.2
  - Bug fixes:
    * Fixed w1c issues in status handling API.
    * Fixed register naming error in API "static inline void TSI_EnableEndOfScanDmaTransferOnly(TSI_Type *base, bool enable)".
    * Removed redundant status flags clear APIs when enable interrupts.
- 2.1.1
  - New features:
    * Changed void TSI_DeInit(TSI_Type *base) to void TSI_Deinit(TSI_Type *base).
- 2.0.1
  - Other changes:
    * Changed default configuration structure member order.

TSI_V5

The current TSI_V5 driver version is 2.0.0.

- 2.0.0
  - Initial version.

VREF

The current VREF driver version is 2.1.0.

- 2.1.0
  - Added new functions:
    - Supported L5K board: add VREF_SetTrim2V1Val() and VREF_GetTrim2V1Val() functions to supply 2V1 output mode.
- 2.0.0
  - Initial version.

WDOG

The current WDOG driver version is 2.0.0.

- 2.0.0
  - Initial version.

CLOCK

The current CLOCK driver version is 2.2.0.

- 2.2.0
  - New features:
    - [KPSDK-9157] Update CLOCK_SetFeiMode/CLOCK_SetFbiMode/ CLOCK_BootTo-FeiMode() to support set MCG_C4[DMX32]=1 in FEI/FBI modes.
  - Bug fixes:
    - Updated IP_CLOCKS array, remove unused gates and add missing gates.
- 2.1.0
  - Other changes:
    - Merge fsl_mcg and fsl_osc into fsl_clock.
- 2.0.0
  - Initial version.

2 Middleware Change Log

DMA_MANAGER

The current DMA_MANAGER driver version is 2.1.0.

- 2.1.0
  - Updated DMA manager interface to support dynamic configuration of the managed area. This is used for a platform with multiple cores.
- 2.0.0
  - Initial version.

FatFs for MCUXpresso SDK

Current version is FatFs R0.13a_rev0.

- R0.13a_rev0
  - Upgraded to version 0.13a. Added patch ff_13a_p1.diff.
- R0.12c_rev1
  - Add nand disk support.
- R0.12c_rev0
  - Upgraded to version 0.12c and applied patches ff_12c_p1.diff and ff_12c_p2.diff.
- R0.12b_rev0
  - Upgraded to version 0.12b.
- R0.11a
  - Added glue functions for low-level drivers (SDHC, SDSPI, RAM, MMC). Modified diskio.c.
  - Added RTOS wrappers to make FatFs thread safe. Modified syscall.c.
  - Renamed ffconf.h to ffconf_template.h. Each application should contain its own ffconf.h.
  - Included ffconf.h into diskio.c to enable the selection of physical disk from ffconf.h by macro definition.
  - Conditional compilation of physical disk interfaces in diskio.c.

mbedTLS for MCUXpresso SDK

The current version of mbedTLS is based on mbedTLS 2.6.0 released 2017-Aug-10.

- 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 KSDK.
    - 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 KSDK.
- 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.

MMCAU library

The current version is 2.0.1.

- 2.0.1
  - Bug fixes:
    - KPSDK-17133 fix bug in fsl_mmcau.c when AES key schedule array is not aligned.
- 2.0.0
  - New features:
    - Q4/2013 release of the CAU library.
    - Added fsl_mmcau.h/fsl_mmcau.c optional layer between application and legacy CAU library (cau_api.h). This API has no alignment requirements.

SDMMC

The current driver version is 2.2.4.

- 2.2.4
  - Bug fix:
    - Fixed DDR mode data sequence miss issue, which is caused by NIBBLE_POS.
  - New features:
    - Increased g_sdmmc 512byte to improve the performance when application use a non-word align data buffer address.
    - Used OCR access mode bits to determine the mmccard high capacity flag.
    - Enabled auto cmd12 for SD read/write.
    - Disabled DDR mode frequency multiply by 2.
- 2.2.3
  - Bug fix:
    - Added reponse check for send operation condition command. If not checked, the card may occasionally init fail.
- 2.2.2
  - Moved set card detect priority operation before enable IRQ.
- 2.2.1
  - New features:
    - Improved MMC Boot feature.
    - Keep SD_Init/SDIO_Init function for forward compatibility.
- 2.2.0
  - New features:
    - Separated the SD/MMC/SDIO init API to xxx_CardInit/xxx_HostInit.

        * Allowed user register card detect callback, select card detect type, and determine the card detect timeout value.
        * Allowed user register the power on/off function, and determine the power on/off delay time.
        * SD_Init/SDIO_Init will be deprecated in the next version.
        * Added write complete wait operation for MMC_Write to fix command timeout issue.
- 2.1.6
  - Enhanced SD IO default driver strength.
- 2.1.5
  - Fixed coverity issue.
  - Fixed SD v1.x card write fail issue. It was caused by the block length set error.
  - Improved SDIO card init sequence and add retry option for SDIO_SwitchToHighSpeed function.
- 2.1.4
  - Miscellaneous:
    * Added Host reset function for card re-initialization.
    * Added Go_Idle function for SDIO card.
    * Added Host_ErrorRecovery function for host error recovery procedure.
    * Added cache maintain operation
    * Added HOST_CARD_INSERT_CD_LEVEL to improve compatibility.
  - Bug fix:
    * Fixed card cannot detect dynamically.
- 2.1.3
  - Bug fix:
    * Non high-speed sdcard init fail at switch to high speed.
  - Miscellaneous:
    * Optimized tuning/mmc switch voltage/mmc select power class/mmc select timing function.
    * Added strobe dll for mmc HS400 mode.
    * Added Delay for SDCard power up.
- 2.1.2
  - New features:
    * Added fsl_host.h to provide prototype to adapt different controller IPs(SDHC/SDIF).
    * Added adaptor code in SDMMC/Port folder to adapt different host controller IPs with different. transfer modes(interrupt/polling/freertos). Application includes a different adaptor code to make application more simple.
    * Adaptor code provides HOST_Init/HOST_Deinit/CardInsertDetect. APIs to do host controller initialize and transfer function configuration. SDMMC card stack uses adaptor code inside stack to wait card insert and configure host when calling card init APIs (SD_-Init/MMC_Init/SDIO_Init).
    * This change requires the user to include host adaptor code into the application. If not changed, link errors saying it cannot find the definition of HOST_Init/HOST_Deinit/-CardInsertDetect appear.
  - New features: Improved SDMMC to support SD v3.0 and emmc v5.0.
  - Bug fix:

        - Fixed wrong comparison between count and length in MMC_ReadBlocks/MMC_WriteBlocks.
- 2.1.1
  - Bug fix:
    - Fixed the block range boundary error when transferring data to MMC card.
    - Fixed the bit mask error in the SD card switch to high speed function.
  - Other changes:
    - Added error code to indicate that SDHC ADMA1 transfer type is not supported yet.
    - Optimized the SD card initialization function.
- 2.1.0
  - Bug fix:
    - Change the callback mechanism when sending a command.
    - Fix the performance low issue when transferring data.
  - Other changes:
    - Changed the name of some error codes returned by internal function.
    - Merged all host related attributes to one structure.
    - Optimize the function of setting maximum data bus width for MMC card.

USB stack for MCUXpresso SDK

The current version of USB stack is 2.0.1.

- 2.0.1
  - Bug fix:
    - fixed some USB issues.
    - Change the audio codec interfaces.
- 2.0.0
  - New features:
    - PTN5110N support.
  - Bug fix:
    - Added some comments, fixed some minor USB issues.
- 1.9.0
  - New features:
    - Examples:
      - usb_pd_alt_mode_dp_host
- 1.8.2
  - Updated license.
- 1.8.1
  - Bug fix:
    - Verified some hardware issues, support aruba_flashless.
- 1.8.0
  - New features:
    - Examples:
      - usb_device_composite_cdc_vcom_cdc_vcom

- usb_device_composite_hid_audio_unified
- usb_pd_sink_battery
- Changed usb_pd_battery to usb_pd_charger_battery.

Bug fix:

- Code cleaned up, removed some irrelevant code.

1.7.0

- New features:
  - USB PD stack support.
- Examples
  - usb_pd
  - usb_pd_battery
  - usb_pd_source_charger

1.6.3

- Bug fix: -IP3511_HS driver control transfer sequence issue, enabled 3511 ip cv test.

1.6.2

- New features:
  - Multi instance support.

1.6.1

- New features:
- Changed the struct variable address method for device_video_virtual_camera and host_phdc_manager.

1.6.0

- New features:
  - Supported Device Charger Detect feature on usb_device_hid_mouse.

1.5.0

- New features:
  - Supported controllers
    - OHCI (Full Speed, Host mode)
    - IP3516 (High Speed, Host mode)
    - IP3511 (High Speed, Device mode)
  - Examples:
    - usb_lpm_device_hid_mouse
    - usb_lpm_device_hid_mouse_lite
    - usb_lpm_host_hid_mouse

1.4.0

- New features:
  - Examples:
    - usb_device_hid_mouse/freertos_static

        * usb_suspend_resume_device_hid_mouse_lite

1.3.0

- New features:
  - Supported roles
    * OTG
  - Supported classes
    * CDC RNDIS
  - Examples
    * usb_otg_hid_mouse
    * usb_device_cdc_vnic
    * usb_suspend_resume_device_hid_mouse
    * usb_suspend_resume_host_hid_mouse

1.2.0

- New features:
  - Supported controllers
    * LPC IP3511 (Full Speed, Device mode)

1.1.0

- Bug fix:
  - Fixed some issues in USB certification.
  - Changed VID and Manufacturer string to NXP.
- New features:
  - Supported classes
    * Pinter
  - Examples:
    * usb_device_composite_cdc_msc_sdcard
    * usb_device_printer_virtual_plain_text
    * usb_host_printer_plain_text

1.0.1

- Bug fix:
  - Improved the efficiency of device audio speaker by changing the transfer mode from interrupt to DMA, thus providing the ability to eliminate the periodic noise.

1.0.0

- New features:
  - Supported roles
    * Device
    * Host
  - Supported controllers:
    * KHCI (Full Speed)
    * EHCI (High Speed)
  - Supported classes:
    * AUDIO

        * CCID
        * CDC
        * HID
        * MSC
        * PHDC
        * VIDEO
    - Examples:
        * usb_device_audio_generator
        * usb_device_audio_speaker
        * usb_device_ccid_smart_card
        * usb_device_cdc_vcom
        * usb_device_cdc_vnic
        * usb_device_composite_cdc_msc
        * usb_device_composite_hid_audio
        * usb_device_composite_hid_mouse_hid_keyboard
        * usb_device_hid_generic
        * usb_device_hid_mouse
        * usb_device_msc_ramdisk
        * usb_device_msc_sdcard
        * usb_device_phdc_weighscale
        * usb_device_video_flexio_ov7670
        * usb_device_video_virtual_camera
        * usb_host_audio_speaker
        * usb_host_cdc
        * usb_host_hid_generic
        * usb_host_hid_mouse
        * usb_host_hid_mouse_keyboard
        * usb_host_msd_command
        * usb_host_msd_fatfs
        * usb_host_phdc_manager
        * usb_keyboard2mouse
        * usb_pin_detect_hid_mouse

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 FreeRTOS 9.0.0. Original package is available at freertos.org.

- 9.0.0_rev3
  - New features:
    - Tickless idle mode support for Cortex-A7. Add fsl_tickless_epit.c and fsl_tickless_generic.h in portable/IAR/ARM_CA9 folder.
    - Enabled float context saving in IAR for Cortex-A7. Added configUSE_TASK_FPU_SUPPORT macros. Modified port.c and portmacro.h in portable/IAR/ARM_CA9 folder.
  - Other changes:
    - Transformed ARM_CM core specific tickless low power support into generic form under freertos.
- 9.0.0_rev2
  - New features:
    - Enabled MCUXpresso thread aware debugging. Add freertos_tasks_c_additions.h and configINCLUDE_FREERTOS_TASK_C_ADDITIONS_H and configFRTOS_MEMORY_SCHEME macros.
- 9.0.0_rev1
  - New features:
    - Enabled -flto optimization in GCC by adding **attribute**((used)) for vTaskSwitchContext.
    - Enabled KDS Task Aware Debugger. Apply FreeRTOS patch to enable configRECORD_STACK_HIGH_ADDRESS macro. Modified files are task.c and FreeRTOS.h.
- 9.0.0_rev0
  - New features:
    - Example freertos_sem_static.
    - Static allocation support RTOS driver wrappers.
  - Other changes:
    - Tickless idle rework. Support for different timers is in separated files (fsl_tickless_systick.c, fsl_tickless_lptmr.c).
    - Removed configuration option configSYSTICK_USE_LOW_POWER_TIMER. Low power timer is now selected by linking of apropriate file fsl_tickless_lptmr.c.
    - Removed configOVERRIDE_DEFAULT_TICK_CONFIGURATION in RVDS port. Use of **attribute**((weak)) is preffered solution. Not same as _week!
- 8.2.3
  - New features:
    - Tickles idle mode support.
    - Added template application for Kinetis Expert (KEx) tool (template_application).
  - Other changes:
    - Folder structure reduction. Keep only Kinetis related parts.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number MCUXSDKK28FRN
Revision 0, 05/2018