NXP Semiconductors
User's Guide

Document Number: MCUXSDKK66FRN
Rev. 0, December 2018

MCUXpresso SDK Release Notes Supporting TWR-K65F180M and FRDM-K66F

Contents

Chapter 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.

Chapter 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, 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 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.

Chapter 3
Development tools

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

- IAR Embedded Workbench for Arm version 8.32.1
- MDK-Arm Microcontroller Development Kit (Keil)® 5.26
- Makefiles support with GCC revision 7-2018-q2-update from Arm Embedded
- MCUXpresso IDE v10.3.0

Chapter 4
Supported development systems

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

Table 1. Supported MCU devices and development boards

Development boards	MCU devices
TWR-K65F180M, FRDM-K66F	MK65FN2M0VMI18WS, MK65FN2M0CAC18WS, MK65FX1M0CAC18WS, MK65FX1M0VMI18WS, MK26FN2M0VLQ18, MK26FN2M0VMD18, MK26FN2M0CAC18, MK26FN2M0VMI18, **MK65FN2M0VMI18**, MK65FN2M0CAC18, MK65FX1M0CAC18, MK65FX1M0VMI18, **MK66FN2M0VMD18**, MK66FN2M0VLQ18, MK66FX1M0VMD18, MK66FX1M0VLQ18

Chapter 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
TinyCBOR	<install_dir>/rtos/amazon-freertos/lib/third_party/tinycbor
Demo applications	<install_dir>/boards/<board_name>/demo_apps
Driver examples	<install_dir>/boards/<board_name>/driver_examples
CMSIS driver examples	<install_dir>/boards/<board_name>/cmsis_driver_examples
FatFS examples	<install_dir>/boards/<board_name>/fatfs_examples
Azure IoT SDK examples	<install_dir>/boards/<board_name>/azure_examples
AWS IoT SDK examples	<install_dir>/boards/<board_name>/aws_examples
emWin examples	<install_dir>/boards/<board_name>/emwin_examples
Multiprocessor examples	<install_dir>/boards/<board_name>/ multiprocessor_examples
RTOS examples	<install_dir>/boards/<board_name>/rtos_examples
WolfSSL examples	<install_dir>/boards/<board_name>/wolfssl_examples
mbed TLS examples	<install_dir>/boards/<board_name>/mbedtls_examples
Documentation	<install_dir>/docs
Middleware	<install_dir>/middleware
mbed TLS	<install_dir>/middleware/mbedtls
Azure IoT	<install_dir>/middleware/azure_iot
jsmn	<install_dir>/middleware/aws_iot/external_libs/jsmn
AWS IoT	<install_dir>/middleware/aws_iot
FatFS stack	<install_dir>/middleware/fatfs
mmCAU	<install_dir>/middleware/mmcau
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
CMSIS drivers	<install_dir>/devices/<device_name>/cmsis_drivers

Table continues on the next page...

Table 2. Release contents (continued)

Utilities such as debug console	<install_dir>/devices/<device_name>/utilities
RTOS Kernel Code	<install_dir>/rtos
Tools	<install_dir>/tools
segger_systemview	<install_dir>/boards/<board>/rtos_examples/visualization/freertos_segger_sysview
percepio_snapshot	<install_dir>/boards/<board>/rtos_examples/visualization/freertos_percepio_snapshot
gradle	<install_dir>/boards/<board>/aws_examples/remote_control_android/gradle, boards/<board>/aws_examples/led_wifi_android/gradle, boards/<board>/aws_examples/device_configuration_android/gradle
AWS Remote Control	<install_dir>/boards/<board>/aws_examples/remote_control_android/AwsRemoteControl.apk
AWS LED WiFi	<install_dir>/boards/<board>/aws_examples/led_wifi_android/AwsLedWifi.apk
AWS Device Configuration	<install_dir>/boards/<board>/aws_examples/device_configuration_android/AwsDeviceConfiguration.apk

Chapter 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 CMSIScompliant 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.

6.2 Middleware

6.2.1 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.

6.2.2 RTOS

The MCUXpresso SDK is integrated with FreeRTOS OS.

6.2.3 CMSIS

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

6.2.4 emWin

The MCUXpresso SDK is pre-integrated with the SEGGER emWin GUIBuilder.

6.2.5 Other middleware

Optional middleware packages can be included in the release based on the user selection. See *<install_dir>/SW-Content-Register.txt* for a list of components and associated licenses.

Chapter 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.

Table continues on the next page...

Table 3. MISRA exceptions (continued)

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.

Chapter 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 USB HUB power supply

The external power supply of the USB HUB must be provided before it can be used. The development board power is not enough to supply multi-level USB HUBs and connected devices. Therefore, the external USB HUB that is connected to the development board should have its own power supply.

8.3 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.

8.4 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.

8.5 No sound issue

Under EHCI of the TWRK65F180M, if the USB device audio speaker connects to a high-speed hub along with an HID device, such as a keyboard or mouse, there is high possibility the sound will not work.

8.6 Create new project without board template

The following components should be selected at the same time when creating a new project without using a board template, including serial_manager, serial_manager_uart, debug_console, and one UART adapter (lpuart_adapter for LPUART IP, uart_adapter for UART IP, lpsci_adapter for LPSCI IP, etc).

MCUXpresso SDK Release Notes Supporting TWR-K65F180M and FRDM-K66F

Change Logs

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_SetSourceChange 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_MasterTransferCreateHandleEDMA function like below: DSPI_MasterTransferCreateHandleEDMA(EXAMPLE_DSPI_MASTER_BASEADDR, &g_dspi_edma_m_handle, DSPI_MasterUserCallback, &userData, &dspiEdmaMasterRxRegToRxDataHandle, NULL, &dspiEdmaMasterIntermediaryToTxRegHandle);
    - 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 2.1.4.

- 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.

ENET

The current ENET driver version is 2.2.3.

- 2.2.3
  - Improved data buffer cache maintainence in the ENET driver.
- 2.2.2
  - Added the APIs for extended multi-ring support.
  - Added the AVB configure API for extended AVB feature support.
- 2.2.1
  - Changed the input data pointer attribute to const in ENET_SendFrame().
- 2.1.1
  - Added the extended MDIO IEEE802.3 Clause 45 MDIO format SMI command APIs.
  - Added the extended interrupt coalescing feature.
  - Combined all storage operations in the ENET_Init to ENET_SetHandler API.
- 2.0.1
  - Bug fix:
    * Used direct transmit busy check when doing data transmit.
  - Miscellaneous changes:
    * Updated IRQ handler work flow.
    * Changed the TX/RX interrupt macro from kENET_RxByteInterrupt to kENET_RxBufferInterrupt, from kENET_TxByteInterrupt to kENET_TxBufferInterrupt.
    * Deleted unnecessary parameters in ENET handler.
- 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 3.0.0.

- 3.0.0
  - Improvements:
    - Reorganized FTFx Flash driver source file.
    - Extracted Flash cache driver from FTFx driver.
    - Extracted FLEXNVM flash driver from FTFx driver.
- 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:
    - Supported FTFL device in FLASH_Swap API.
    - Supported various pflash start addresses.
    - Added support for KV58 in cache clear function.
    - Added support for devices 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.

FLEXCAN

The current FLEXCAN driver version is 2.3.2.

- 2.3.2
  - Improvements:
    * Implementation for ERR005959.
    * Implementation for ERR005829.
    * Implementation for ERR006032.
- 2.3.1
  - Bug fixes:
    * Adding correct handle when kStatus_FLEXCAN_TxSwitchToRx comming.
- 2.3.0
  - Improvements:
    * Added self wake support from STOP mode in the interrupt handling.
- 2.2.3
  - Bug fix:
    * Fixed CANFD data phase's bit rate not set as expected.
- 2.2.2
  - Improvements:
    * Added time stamp feature and enabled in interrupt_transfer example.
- 2.2.1
  - Improvements:
    * Separated CANFD initialization API.
    * In the interrupt handling, fixed issue that user cannot use normal CAN API when FD is present.
- 2.2.0
  - Improvements:
    * Added FSL_FEATURE_FLEXCAN_HAS_SUPPORT_ENGINE_CLK_SEL_REMOVE feature to support SoCs without CAN Engine Clock selection in FlexCAN module.
    * Added FlexCAN Serial Clock Operation to support i.MX SoCs.
- 2.1.0
  - Bug fixes:
    * Fixed wrong function name spelling: FLEXCAN_XXX() -> FLEXCAN_XXX();
    * Moved Freeze Enable/Disable setting from FLEXCAN_Enter/ExitFreezeMode() to FLEXCAN_Init();
    * Fixed wrong helper macro values.
  - Other changes:
    * Hided FLEXCAN_Reset() to user.
    * Used NDEBUG macro to wrap FLEXCAN_IsMbOccupied() function instead of DEBUG macro.
- 2.0.0

  - Initial version.

FTM

The current FTM driver version is 2.1.0.

- 2.1.0
  - New feature:
    - Add a new API FTM_SetupPwmMode() to allow user set the channel match value in unit of timer ticks. New configure structure called ftm_chnl_pwm_config_param_t was offerred to configure the channel's PWM parameters. This API is similar with FTM_SetupPwm() API, but the new API will not set the timer period(MOD value), it will be useful for users to set the PWM parameters without changing the timer period.
  - Bug fixes:
    - Add feature macro to enable/disable the external trigger source configuration.
- 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 2.3.0.

- 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".

I2C

The current I2C driver version is 2.0.6.

- 2.0.6
  - 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.
- 2.0.5
  - 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 2.0.2.

- 2.0.2
  - Optimization:
    * Correct driver function LLWU_SetResetPinMode parameter name.
- 2.0.1
  - Miscellaneous changes:
    * Updates for KL8x.
- 2.0.0
  - Initial version.

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 2.2.6.

- 2.2.6
  - Fixed the repeated 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
  - Updated 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 2.0.1.

- 2.0.1
  - Bug fix:
    * Cleared timer enable bit for all channels in function PIT_Init() to make sure all channels stay in disable status before setting other configurations.
- 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 2.1.0.

- 2.1.0
  - New features:
    * Updated the driver code to adapt the case of the interrupt configurations in GPIO module. Will move the pin configuration APIs to the 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.

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 2.1.0.

- 2.1.0
  - Added feature macro check for many features.
- 2.0.0
  - Initial version.

SAI

The current SAI driver version is 2.1.7.

- 2.1.7
  - Improvements:
    - Added feature macro test for the mclkSource member in sai_config_t.
  - Bug fix:
    - Fixed the build error caused by feature macro test for mclkSource.
- 2.1.6
  - Improvements:
    - Added feature macro test for mclkSourceClockHz check.
    - Added bit clock source name for general devices.
  - Bug fix:
    - Fixed incorrect channel numbers setting while call RX/TX set format together.
- 2.1.5
  - Bug fixes:
    - Corrected SAI3 driver IRQ handler name.
    - Added I2S4/5/6 IRQ handler.
    - Added base in handler structure to support different instances share one IRQ number.
  - New features:

        * Updated 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 features:
    * 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 changes:
    * 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 fixes:
    * Fixed WLC bug while reading TCSR/RCSR registers.
    * Fixed MOE enable flow issue, moved MOE enable after MICS settings in SAI_TxInit/SAI_RxInit.
- 2.0.0
  - Initial version.

SDHC

The current SDHC driver version is 2.1.8.

- 2.1.8
  - Disable useless interrupt while DMA is used.
  - Fix MDK 66-D warning.
- 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 2.0.4.

- 2.0.4
  - When entering stop modes, use the RAM function for the flash synchronize issue. The application should make sure that the RW data of fsl_smc.c is located in the 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.

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 2.0.1.

- 2.0.0
  - Initial version.
- 2.0.1
  - Added functions for M_TX_USED bitfield for ke16z only (Unused TX mutual pins can work as GPIO).

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: December 2018
Document identifier: MCUXSDKK66FRN

arm