MCUXpresso SDK Release Notes Supporting frdmke15z

Change Logs

NXP Semiconductors

Contents

Driver Change Log

Title Page No.

Middleware Change Log

Title **Page No.**

Component Change Log

1 Driver Change Log

CLOCK

Current CLOCK driver version is 2.4.0

- 2.4.0
  - Bug Fixes
    - Removed unimplemented kSCG_AsyncDiv1Clk.
- 2.3.1
  - Bug Fixes
    - Fixed MISRA C-2012 rule 10.1, rule 10.4, rule 10.8, rule 15.5 and so on.
- 2.3.0
  - New Features
    - Moved SDK_DelayAtLeastUs function from clock driver to common driver.
- 2.2.0
  - New Features
    - Added new API CLOCK_DelayAtLeastUs() implemented by DWT to allow users set delay in unit of microsecond.
- 2.1.2
  - Bug Fixes
    - Fixed OSC32 can not use EXTAL32 clock source issue.
- 2.1.1
  - Improvements
    - Changed reserved bit fields in _scg_sys_clk_config struct into unnamed bit fields.
- 2.1.0
  - Other Changes
    - Merged fsl_scg and fsl_osc into fsl_clock.
- 2.0.0
  - Initial version.

LPI2C_CMSIS

Current LPI2C_CMSIS driver version is 2.1

- 2.1
  - Bug Fixes
    - Fixed the MISRA-2012 violations.
      - Fixed rule 8.4, 8.6, 10.1, 10.3, 10.4, 11.1, 11.8, 14.4, 16.1, 16.3, 17.7, 17.3, 17.7, 20.9.
- 2.0
  - Initial version.

LPSPI_CMSIS

Current LPSPI_CMSIS driver version is 2.4

- 2.4
  - Bug Fixes
    - Update driver to fix warnings reported by IAR v9.
- 2.3
  - Bug Fixes
    - Fixed the MISRA-2012 violations.
      - Fixed rule 8.4, 8.6, 10.1, 10.3, 10.4, 11.1, 11.8, 14.4, 16.1, 16.3, 17.7, 17.3, 17.7, 20.9.
- 2.2
  - Bug Fixes
    - Fixed the bug that, the parameter num of APIs ARM_SPI_Transfer, ARM_SPI_Send and ARM_SPI_Receive, and the return value of API ARM_SPI_GetDataCount should be the number of data item defined by datawidth, rather than the number of byte.
- 2.1
  - Bug Fixes
    - Fixed the incorrect clock polarity assignment in the driver. For ARM_SPI_CPOL0_CPHA0 and other frame format parameters, CPOL = 0 means kSPI_ClockPolarityActiveHigh not kSPI_ClockPolarityActiveLow in driver.
  - New features
    - Allowed user to set up the default transmit value by using ARM_SPI_SET_DEFAULT_TX_VALUE. Please note that this is not supported in slave interrupts, because the pin will stay tristated if tX buffer is NULL.
    - Enabled slave select mode. Note this has no effect when user sets any of them because the driver can only support the hardware control function.
    - Enabled 3-Wire mode, user can use ARM_SPI_MODE_MASTER_SIMPLEX/ARM_SPI_MODE_SLAVE_SIMPLEX to enable this feature. For ARM_SPI_MODE_MASTER_SIMPLEX mode, the SOUT pin is selected as the input/output pin, and for ARM_SPI_MODE_SLAVE_SIMPLEX, the SIN pin is selected as the input/output pin.
- 2.0
  - Initial version.

ACMP

The current ACMP driver version is 2.0.6.

- 2.0.6
  - Bug Fixes
    - Fixed the wrong comments, the DAC value should range from 0 to 255.
- 2.0.5
  - Bug Fixes
    - Fixed the out-of-bounds error of Coverity caused by missing an assert sentence to avoid the return value of ACMP_GetInstance() exceeding the array bounds.

        - Fixed the violations of MISRA C-2012 rules:
            - Rule 10.1, 14.4, 16.4, 17.7.
- 2.0.4
    - Bug Fixes
        - Avoided changing w1c bit in ACMP_SetRoundRobinPreState().
- 2.0.3
    - New Features
        - Added feature functions for usage of different power domains(1.8 V and 3 V). These functions are first enabled in ULP1. They are about:
            - ACMP_EnableLinkToDAC()
            - ACMP_SetDiscreteModeConfig()
            - ACMP_GetDefaultDiscreteModeConfig()
- 2.0.2
    - Other Changes
        - Changed coding style of peripheral base address from "s_acmpBases" to "s_acmpBase".
- 2.0.1
    - Bug Fixes
        - Fixed bug regarding the function "ACMP_SetRoundRobinConfig". It will not continue execution but returns directly after disabling round robin mode.

ADC12

The current ADC12 driver version is 2.0.6.

- 2.0.6
    - Improvements
        - Removed useless comments of ADC12_DoAutoCalibration() function.
- 2.0.5
    - Bug Fixes
        - Fixed the violations of MISRA C-2012 rule 10.4.
- 2.0.4
    - Bug Fixes
        - Fixed the violations of MISRA C-2012 rules:
            - Rule 4.7 10.1 10.3 10.4 10.8 12.2 16.4 17.7
- 2.0.3
    - Improvements
        - Used conversion control feature macro instead of that in IO map.
- 2.0.2
    - Bug Fixes
        - Set ADC clock frequency as half of the maximum value for calibration.
- 2.0.1
    - New Features
        - Added a feature to control enablement of DMA.
- 2.0.0

  - Initial version.

COMMON

The current COMMON driver version is 2.3.2.

- 2.3.2
  - Improvements
    - Make driver aarch64 compatible
- 2.3.1
  - Bug Fixes
    - Fixed MAKE_VERSION overflow on 16-bit platforms.
- 2.3.0
  - Improvements
    - Split the driver to common part and CPU architecture related part.
- 2.2.10
  - Bug Fixes
    - Fixed the ATOMIC macros build error in cpp files.
- 2.2.9
  - Bug Fixes
    - Fixed MISRA C-2012 issue, 5.6, 5.8, 8.4, 8.5, 8.6, 10.1, 10.4, 17.7, 21.3.
    - Fixed SDK_Malloc issue that not allocate memory with required size.
- 2.2.8
  - Improvements
    - Included stddef.h header file for MDK tool chain.
  - New Features:
    - Added atomic modification macros.
- 2.2.7
  - Other Change
    - Added MECC status group definition.
- 2.2.6
  - Other Change
    - Added more status group definition.
  - Bug Fixes
    - Undef __VECTOR_TABLE to avoid duplicate definition in cmsis_clang.h
- 2.2.5
  - Bug Fixes
    - Fixed MISRA C-2012 rule-15.5.
- 2.2.4
  - Bug Fixes
    - Fixed MISRA C-2012 rule-10.4.
- 2.2.3
  - New Features
    - Provided better accuracy of SDK_DelayAtLeastUs with DWT, use macro SDK_DELA-

Y_USE_DWT to enable this feature.
    - Modified the Cortex-M7 delay count divisor based on latest tests on RT series boards, this setting lets result be closer to actual delay time.
- 2.2.2
  - New Features
    - Added include RTE_Components.h for CMSIS pack RTE.
- 2.2.1
  - Bug Fixes
    - Fixed violation of MISRA C-2012 Rule 3.1, 10.1, 10.3, 10.4, 11.6, 11.9.
- 2.2.0
  - New Features
    - Moved SDK_DelayAtLeastUs function from clock driver to common driver.
- 2.1.4
  - New Features
    - Added OTFAD into status group.
- 2.1.3
  - Bug Fixes
    - MISRA C-2012 issue fixed.
      - Fixed the rule: rule-10.3.
- 2.1.2
  - Improvements
    - Add SUPPRESS_FALL_THROUGH_WARNING() macro for the usage of suppressing fallthrough warning.
- 2.1.1
  - Bug Fixes
    - Deleted and optimized repeated macro.
- 2.1.0
  - New Features
    - Added IRQ operation for XCC toolchain.
    - Added group IDs for newly supported drivers.
- 2.0.2
  - Bug Fixes
    - MISRA C-2012 issue fixed.
      - Fixed the rule: rule-10.4.
- 2.0.1
  - Improvements
    - Removed the implementation of LPC8XX Enable/DisableDeepSleepIRQ() function.
    - Added new feature macro switch "FSL_FEATURE_HAS_NO_NONCACHEABLE_SECTION" for specific SoCs which have no noncacheable sections, that helps avoid an unnecessary complex in link file and the startup file.
    - Updated the align(x) to **attribute**(aligned(x)) to support MDK v6 armclang compiler.
- 2.0.0
  - Initial version.

CRC

The current CRC driver version is 2.0.3.

- 2.0.3
  - Bug fix:
    - Fix MISRA issues.
- 2.0.2
  - Bug fix:
    - Fix MISRA issues.
- 2.0.1
  - Bug fix:
    - DATA and DATALL macro definition moved from header file to source file.
- 2.0.0
  - Initial version.

DMAMUX

The current DMAMUX driver version is 2.0.5.

- 2.0.5
  - Improvements
    - Added feature FSL_FEATURE_DMAMUX_CHCFG_REGISTER_WIDTH for the difference of CHCFG register width.
- 2.0.4
  - Bug Fixes
    - Fixed violations of MISRA C-2012 rule 10.4.
- 2.0.3
  - Bug Fixes
    - Fixed the issue for MISRA-2012 check.
      - Fixed rule 10.4 and rule 10.3.
- 2.0.2
  - New Features
    - Added an always-on enable feature to a DMA channel for ULP1 DMAMUX support.
- 2.0.1
  - Bug Fixes
    - Fixed the build warning issue by changing the type of parameter source from uint8_t to uint32_t when setting DMA request source in DMAMUX_SetSourceChange.
- 2.0.0
  - Initial version.

EDMA

The current eDMA driver version is 2.4.4.

- 2.4.4
  - Bug Fixes
    - Fixed comments by replacing STCD with TCD
    - Fixed the TCD overwrite issue when submit transfer request in the callback if there is a active TCD in hardware.
    - Fixed violations of MISRA C-2012 rule 10.8,5.6.
- 2.4.3
  - Improvements
    - Added FSL_FEATURE_MEMORY_HAS_ADDRESS_OFFSET to convert the address between system mapped address and dma quick access address.
  - Bug Fixes
    - Fixed the wrong tcd done count calculated in first TCD interrupt for the non scatter gather case.
- 2.4.2
  - Bug Fixes
    - Fixed the wrong tcd done count calculated in first TCD interrupt by correct the initial value of the header.
    - Fixed violations of MISRA C-2012 rule 10.3, 10.4.
- 2.4.1
  - Bug Fixes
    - Added clear CITER and BITER registers in EDMA_AbortTransfer to make sure the TCD registers in a correct state for next calling of EDMA_SubmitTransfer.
    - Removed the clear DONE status for ESG not enabled case to aovid DONE bit cleared unexpectedly.
- 2.4.0
  - Improvements
    - Added api EDMA_EnableContinuousChannelLinkMode to support continuous link mode.
    - Added apis EDMA_SetMajorOffsetConfig/EDMA_TcdSetMajorOffsetConfig to support major loop address offset feature.
    - Added api EDMA_EnableChannelMinorLoopMapping for minor loop offset feature.
    - Removed the reduntant IRQ Handler in edma driver.
- 2.3.2
  - Improvements
    - Fixed HIS ccm issue in function EDMA_PrepareTransferConfig.
    - Fixed violations of MISRA C-2012 rule 11.6, 10.7, 10.3, 18.1.
  - Bug Fixes
    - Added ACTIVE & BITER & CITER bitfields to determine the channel status to fixed the issue of the transfer request cannot submit by function EDMA_SubmitTransfer when channel is idle.
- 2.3.1
  - Improvements
    - Added source/destination address alignment check.
    - Added driver IRQ handler support for multi DMA instance in one SOC.
- 2.3.0

  - Improvements
    - Added new api EDMA_PrepareTransferConfig to allow different configurations of width and offset.
  - Bug Fixes
    - Fixed violations of MISRA C-2012 rule 10.4, 10.1.
    - Fixed the Coverity issue regarding out-of-bounds write.
- 2.2.0
  - Improvements
    - Added peripheral-to-peripheral support in EDMA driver.
- 2.1.9
  - Bug Fixes
    - Fixed MISRA issue: Rule 10.7 and 10.8 in function EDMA_DisableChannelInterrupts and EDMA_SubmitTransfer.
    - Fixed MISRA issue: Rule 10.7 in function EDMA_EnableAsyncRequest.
- 2.1.8
  - Bug Fixes
    - Fixed incorrect channel preemption base address used in EDMA_SetChannelPreemption-Config API which causes incorrect configuration of the channel preemption register.
- 2.1.7
  - Bug Fixes
    - Fixed incorrect transfer size setting.
      - Added 8 bytes transfer configuration and feature for RT series;
      - Added feature to support 16 bytes transfer for Kinetis.
    - Fixed the issue that EDMA_HandleIRQ would go to incorrect branch when TCD was not used and callback function not registered.
- 2.1.6
  - Bug Fixes
    - Fixed KW3X MISRA Issue.
      - Rule 14.4, 10.8, 10.4, 10.7, 10.1, 10.3, 13.5, and 13.2.
  - Improvements
    - Cleared the IRQ handler unavailable for specific platform with macro FSL_FEATURE_-EDMA_MODULE_CHANNEL_IRQ_ENTRY_SHARED_OFFSET.
- 2.1.5
  - Improvements
    - Improved EDMA IRQ handler to support half interrupt feature.
- 2.1.4
  - Bug Fixes
    - Cleared enabled request, status during EDMA_Init for the case that EDMA is halted before reinitialization.
- 2.1.3
  - Bug Fixes
    - Added clear DONE bit in IRQ handler to avoid overwrite TCD issue.
    - Optimized 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 Fixes
    * Fixed the incorrect pubweak IRQHandler name issue, which caused re-definition build errors when client set his/her own IRQHandler, by changing the 32-channel IRQHandler name to DriverIRQHandler.
- 2.0.2
  - Bug Fixes
    * Fixed incorrect minorLoopBytes type definition in _edma_transfer_config struct, and defined minorLoopBytes as uint32_t instead of uint16_t.
- 2.0.1
  - Bug Fixes
    * 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 2.0.3.

- 2.0.3
  - Bug Fixes
    * Fixed violation of MISRA C-2012 rules: 10.1, 10.3.
- 2.0.2
  - Bug Fixes

        - Fixed violation of MISRA C-2012 rules: 10.3, 10.4.
- 2.0.1
    - Bug Fixes
        - Fixed the hard fault in EWM_Deinit.
- 2.0.0
    - Initial version.

FLASH

Current FLASH driver version is 3.1.2

- 3.1.2
    - Bug Fixes — Remove redundant comments.
- 3.1.1
    - Bug Fixes — MISRA C-2012 issue fixed: rule 10.3
- 3.1.0
    - New Feature
        - Support erase flash asynchronously.
- 3.0.2
    - Bug Fixes — MISRA C-2012 issue fixed: rule 8.4, 17.7, 10.4, 16.1, 21.15, 11.3, 10.7 — building warning -Wnull-dereference on arm compiler v6
- 3.0.1
    - New Features
        - Added support FlexNVM alias for (kw37/38/39).
- 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 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 2.0.4.

- 2.0.4
    - Bug Fixes
        * Fixed MISRA 8.4 issues.
- 2.0.3
    - Bug Fixes
        * Fixed MISRA 10.4 issues.
- 2.0.2
    - Improvements
        * Split FLEXIO component which combines all flexio/flexio_uart/flexio_i2c/flexio_i2s drivers into several components: FlexIO component, flexio_uart component, flexio_i2c_master component, and flexio_i2s component.
    - Bug Fixes
        * Fixed MISRA issues
            · Fixed rules 10.1, 10.3, 10.4, 10.7, 11.6, 11.9, 14.4, 17.7.
- 2.0.1
    - Bug Fixes
        * Fixed the dozen mode configuration error in FLEXIO_Init API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.

FLEXIO_UART

The current FLEXIO_UART driver version is 2.4.0.

- 2.4.0
    - Improvements
        * Use separate data for TX and RX in flexio_uart_transfer_t.

  - Bug Fixes
    - Fixed bug that when ring buffer is used, if some data is received in ring buffer first before calling FLEXIO_UART_TransferReceiveNonBlocking, the received data count returned by FLEXIO_UART_TransferGetReceiveCount is wrong.
- 2.3.0
  - Improvements
    - Added check for baud rate's accuracy that returns kStatus_FLEXIO_UART_BaudrateNotSupport when the best achieved baud rate is not within 3% error of configured baud rate.
  - Bug Fixes
    - Added codes in FLEXIO_UART_TransferCreateHandle to clear pending NVIC IRQ before enabling NVIC IRQ, to fix issue of pending IRQ interfering the on-going process.
- 2.2.0
  - Improvements
    - Added timeout mechanism when waiting for certain states in transfer driver.
  - Bug Fixes
    - Fixed MISRA 10.4 issues.
- 2.1.6
  - Bug Fixes
    - Fixed IAR Pa082 warnings.
    - Fixed MISRA issues
      - Fixed rules 10.1, 10.3, 10.4, 10.7, 11.6, 11.9, 14.4, 17.7.
- 2.1.5
  - Improvements
    - Triggered user callback after all the data in ringbuffer were received in FLEXIO_UART_TransferReceiveNonBlocking.
- 2.1.4
  - Improvements
    - Unified 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 configuration instead of disabling module and clock.
      - Updated module Enable APIs to only support enable operation.
- 2.1.2
  - Bug Fixes
    - Fixed the transfer count calculation issue in FLEXIO_UART_TransferGetReceiveCount, FLEXIO_UART_TransferGetSendCount, FLEXIO_UART_TransferGetReceiveCountDMA, FLEXIO_UART_TransferGetSendCountDMA, FLEXIO_UART_TransferGetReceiveCountEDMA and FLEXIO_UART_TransferGetSendCountEDMA.
    - Fixed the Dozen mode configuration error in FLEXIO_UART_Init API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.

        - Added code to report errors if the user sets a too-low-baudrate which FLEXIO cannot reach.
        - Disabled FLEXIO_UART receive interrupt instead of all NVICs when reading data from ring buffer. If ring buffer is used, receive nonblocking will disable all NVIC interrupts to protect the ring buffer. This had negative effects on other IPs 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 an error handle to handle the situation that data count is zero or data buffer is NULL.

FLEXIO_UART_EDMA

The current FLEXIO_UART_EDMA driver version is 2.3.1.

- 2.3.1
  - Bug Fixes
    - Fixed violations of the MISRA C-2012 rules.
- 2.3.0
  - Refer FLEXIO_UART driver change log to 2.3.0

FLEXIO_I2C

The current FLEXIO_I2C driver version is 2.4.0.

- 2.4.0
  - Improvements
    - Added delay of 1 clock cycle in FLEXIO_I2C_MasterTransferRunStateMachine to ensure that bus would be idle before next transfer if master is nacked.
    - Fixed issue that the restart setup time is less than the time in I2C spec by adding delay of 1 clock cycle before restart signal.
- 2.3.0
  - Improvements
    - Used 3 timers instead of 2 to support transfer which is more than 14 bytes in single transfer.
    - Improved FLEXIO_I2C_MasterTransferGetCount so that the API can check whether the transfer is still in progress.
  - Bug Fixes
    - Fixed MISRA 10.4 issues.

- 2.2.0
  - New Features
    - Added timeout mechanism when waiting certain state in transfer API.
    - Added an API for checking bus pin status.
  - Bug Fixes
    - Fixed COVERITY issue of useless call in FLEXIO_I2C_MasterTransferRunStateMachine.
    - Fixed MISRA issues
      - Fixed rules 10.1, 10.3, 10.4, 10.7, 11.6, 11.9, 14.4, 17.7.
    - Added codes in FLEXIO_I2C_MasterTransferCreateHandle to clear pending NVIC IRQ, disable internal IRQs before enabling NVIC IRQ.
    - Modified code so that during master's nonblocking transfer the start and slave address are sent after interrupts being enabled, in order to avoid potential issue of sending the start and slave address twice.
- 2.1.7
  - Bug Fixes
    - Fixed the issue that FLEXIO_I2C_MasterTransferBlocking did not wait for STOP bit sent.
    - Fixed COVERITY issue of useless call in FLEXIO_I2C_MasterTransferRunStateMachine.
    - Fixed the issue that I2C master did not check whether bus was busy before transfer.
- 2.1.6
  - Bug Fixes
    - Fixed the issue that I2C Master transfer APIs(blocking/non-blocking) did not support the situation of master transfer with subaddress and transfer data size being zero, which means no data followed the subaddress.
- 2.1.5
  - Improvements
    - Unified 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 disabling module/clock.
      - Updated module Enable APIs to only support enable operation.
- 2.1.3
  - Improvements
    - Changed the prototype of FLEXIO_I2C_MasterInit to return kStatus_Success if initialized successfully or to return kStatus_InvalidArgument if "(srcClock_Hz / masterConfig->baudRate_Bps) / 2 - 1" exceeds 0xFFU.
- 2.1.2
  - Bug Fixes
    - Fixed the FLEXIO I2C issue where the master could not receive data from I2C slave in high baudrate.

    - Fixed the FLEXIO I2C issue where the master could not receive NAK when master sent non-existent addr.
    - Fixed the FLEXIO I2C issue where the master could not get transfer count successfully.
    - Fixed the FLEXIO I2C issue where the master could not 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 issue that FLEXIO_I2C_MasterTransferBlocking API called FLEXIO_I2C_MasterTransferCreateHandle, which lead to the s_flexioHandle/s_flexioIsr/s_flexioType variable being written. Then, if calling FLEXIO_I2C_MasterTransferBlocking API multiple times, the s_flexioHandle/s_flexioIsr/s_flexioType variable would not be written any more due to it being out of range. This lead to the following situation: NonBlocking transfer APIs could not work due to the fail of register IRQ.
- 2.1.1
  - Bug Fixes
    - Implemented the FLEXIO_I2C_MasterTransferBlocking API which is 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 2.3.0.

- 2.3.0
  - New Features
    - Supported FLEXIO_SPI slave transfer with continuous master CS signal and CPHA=0.
    - Supported FLEXIO_SPI master transfer with continuous CS signal.
    - Support 32 bit transfer width.
  - Bug Fixes
    - Fixed wrong timer compare configuration for dma/edma transfer.
    - Fixed wrong byte order of rx data if transfer width is 16 bit, since the we use shifter buffer bit swapped/byte swapped register to read in received data, so the high byte should be read from the high bits of the register when MSB.
- 2.2.1
  - Bug Fixes
    - Fixed bug in FLEXIO_SPI_MasterTransferAbortEDMA that when aborting EDMA transfer EDMA_AbortTransfer should be used rather than EDMA_StopTransfer.
- 2.2.0
  - Improvements
    - Added timeout mechanism when waiting certain states in transfer driver.

  - Bug Fixes
    - Fixed MISRA 10.4 issues.
    - Added codes in FLEXIO_SPI_MasterTransferCreateHandle and FLEXIO_SPI_SlaveTransferCreateHandle to clear pending NVIC IRQ before enabling NVIC IRQ, to fix issue of pending IRQ interfering the on-going process.
- 2.1.3
  - Improvements
    - Unified component full name to FLEXIO SPI(DMA/EDMA) Driver.
  - Bug Fixes
    - Fixed MISRA issues
      - Fixed rules 10.1, 10.3, 10.4, 10.7, 11.6, 11.9, 14.4, 17.7.
- 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 disabling module/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 when FLEXIO SPI works in eDMA and interrupt mode with 16-bit per frame and Lsbfirst.
    - Fixed the Dozen mode configuration error in FLEXIO_SPI_MasterInit/FLEXIO_SPI_SlaveInit API. For enableInDoze = true, the configuration should be 0; for enableInDoze = false, the configuration should be 1.
  - Improvements
    - Added #ifndef/#endif to allow users 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.

FTM

The current FTM driver version is 2.5.0.

- 2.5.0
  - Improvements

    - Added FTM_CalculateCounterClkDiv to help calculates the counter clock prescaler.
    - Modify FTM_UpdatePwmDutycycle API to make it return pwm duty cycles status.
  - Bug Fixes
    - Fixed TPM_SetupPwm can't configure 100% center align combined PWM issues.
- 2.4.1
  - Bug Fixes
    - Added function macro to determine if FTM instance has only basic features, to prevent access to protected register bits.
- 2.4.0
  - Improvements
    - Added CNTIN register initialization in FTM_SetTimerPeriod API.
    - Added a new API to read the captured value of a FTM channel configured in capture mode:
      - FTM_GetInputCaptureValue()
- 2.3.0
  - Improvements
    - Added support of EdgeAligned/CenterAligned/Asymmetrical combine PWM mode in FTM_SetupPWM() and FTM_SetupPwmMode() APIs.
    - Remove kFTM_ComplementaryPwm from support PWM mode, and add new parameter "enableComplementary" in structure ftm_chnl_pwm_signal_param_t.
    - Rename FTM_SetupFault() API to FTM_SetupFaultInput() to avoid ambiguity.
- 2.2.3
  - Bug Fixes
    - MISRA C-2012 issue fixed: rule 14.4 and 17.7.
- 2.2.2
  - Bug Fixes
    - Fixed the issue that when FTM instance has only TPM features cannot be initialized by FTM_Init() function. By added function macro to assert FTM is TPM only instance.
- 2.2.1
  - Bug Fixes
    - MISRA C-2012 issue fixed: rule 10.1, 10.3, 10.4, 10.6, 10.7 and 11.9.
- 2.2.0
  - Bug Fixes
    - Fixed the issue of comparison between signed and unsigned integer expressions.
  - Improvements
    - Added support of complementary mode in FTM_SetupPWM() and FTM_SetupPwmMode() APIs.
    - Added new parameter "enableDeadtime" in structure ftm_chnl_pwm_signal_param_t.
- 2.1.1
  - Bug Fixes
    - Fixed COVERITY integer handing issue where the right operand of a left bit shift statement should not be a negative value. This appears in FTM_SetReloadPoints().
- 2.1.0
  - Improvements
    - Added a new API FTM_SetupPwmMode() to allow the user to set the channel match

value in units of timer ticks. New configure structure called ftm_chnl_pwm_config_-param_t was added 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
    * Added feature macro to enable/disable the external trigger source configuration.
- 2.0.4
  - Improvements
    * Added a new API to enable DMA transfer:
      · FTM_EnableDmaTransfer()
- 2.0.3
  - Bug Fixes
    * Updated the FTM driver to enable fault input after configuring polarity.
- 2.0.2
  - Improvements
    * 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 2.6.0.

- 2.6.0
  - New Features
    * Added API to get GPIO version information.
    * Added API to control a pin for general purpose input.
    * Added some APIs to control pin in secure and previliege status.
- 2.5.3
  - Bug Fixes
    * Correct the feature macro typo: FSL_FEATURE_GPIO_HAS_NO_INDEP_OUTPUT_-CONTORL.
- 2.5.2
  - Improvements
    * Improved GPIO_PortSet/GPIO_PortClear/GPIO_PortToggle functions to support devices without Set/Clear/Toggle registers.

- 2.5.1
  - Bug Fixes
    - Fixed wrong macro definition.
    - Fixed MISRA C-2012 rule issues in the FGPIO_CheckAttributeBytes() function.
    - Defined the new macro to separate the scene when the width of registers is different.
    - Removed some redundant macros.
  - New Features
    - Added some APIs to get/clear the interrupt status flag when the port doesn't control pins' interrupt.
- 2.4.1
  - Improvements
    - Improved GPIO_CheckAttributeBytes() function to support 8 bits width GACR register.
- 2.4.0
  - Improvements
    - API interface added:
      - New APIs were added to configure the GPIO interrupt clear settings.
- 2.3.2
  - Bug Fixes
    - Fixed the issue for MISRA-2012 check.
      - Fixed rule 3.1, 10.1, 8.6, 10.6, and 10.3.
- 2.3.1
  - Improvements
    - Removed deprecated APIs.
- 2.3.0
  - New Features
    - Updated 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
  - Improvements
    - API interface changes:
      - Refined naming of APIs while keeping all original APIs by marking them as deprecated. The original APIs will be removed in next release. The main change is updating APIs with prefix of _PinXXX() and _PortXXX.
- 2.1.1
  - Improvements
    - API interface changes:
      - Added an API for the check attribute bytes.
- 2.1.0
  - Improvements
    - API interface changes:
      - Added "pins" or "pin" to some APIs' names.
      - Renamed "_PinConfigure" to "GPIO_PinInit".

LPI2C

The current LPI2C driver version is 2.3.1.

- 2.3.1
  - Improvements
    - Updated LPI2C_GetCyclesForWidth to add the parameter of minimum cycle, because for master SDA/SCL filter, master bus idle/pin low timeout and slave SDA/SCL filter configuration, 0 means disabling the feature and cannot be used.
  - Bug Fixes
    - Fixed bug in LPI2C_SlaveTransferHandleIRQ that when restart detect event happens the transfer structure should not be cleared.
    - Fixed bug in LPI2C_RunTransferStateMachine, that when only slave address is transferred or there is still data remaining in tx FIFO the last byte's nack cannot be ignored.
    - Fixed bug in slave filter doze enable, that when FILTDZ is set it means disable rather than enable.
    - Fixed bug in the usage of LPI2C_GetCyclesForWidth. First its return value cannot be used directly to configure the slave FILTSDA, FILTSCL, DATAVD or CLKHOLD, because the real cycle width for them should be FILTSDA+3, FILTSCL+3, FILTSCL+DATAVD+3 and CLKHOLD+3. Second when cycle period is not affected by the prescaler value, prescaler value should be passed as 0 rather than 1.
    - Fixed wrong default setting for LPI2C slave. If enabling the slave tx SCL stall, then the default clock hold time should be set to 250ns according to I2C spec for 100kHz standard mode baudrate.
    - Fixed bug that before pushing command to the tx FIFO the FIFO occupation should be checked first in case FIFO overflow.
- 2.3.0
  - New Features
    - Supported reading more than 256 bytes of data in one transfer as master.
    - Added API LPI2C_GetInstance.
  - Bug Fixes
    - Fixed bug in LPI2C_MasterTransferAbortEDMA, LPI2C_MasterTransferAbort and LPI2C_MasterTransferHandleIRQ that before sending stop signal whether master is active and whether stop signal has been sent should be checked, to make sure no FIFO error or bus error will be caused.
    - Fixed bug in LPI2C master EDMA transactional layer that the bus error cannot be caught and returned by user callback, by monitoring bus error events in interrupt handler.
    - Fixed bug in LPI2C_GetCyclesForWidth that the parameter used to calculate clock cycle should be 2^prescaler rather than prescaler.
    - Fixed bug in LPI2C_MasterInit that timeout value should be configured after baudrate, since the timeout calculation needs prescaler as parameter which is changed during baudrate configuration.
    - Fixed bug in LPI2C_MasterTransferHandleIRQ and LPI2C_RunTransferStateMachine that when master writes with no stop signal, need to first make sure no data remains in the tx FIFO before finishes the transfer.

- 2.2.0
  - Bug Fixes
    - Fixed issue that the SCL high time, start hold time and stop setup time do not meet I2C specification, by changing the configuration of data valid delay, setup hold delay, clock high and low parameters.
    - MISRA C-2012 issue fixed.
      - Fixed rule 8.4, 13.5, 17.7, 20.8.
- 2.1.12
  - Bug Fixes
    - Fixed MISRA advisory 15.5 issues.
- 2.1.11
  - Bug Fixes
    - Fixed the bug that, during master non-blocking transfer, after the last byte is sent/received, the kLPI2C_MasterNackDetectFlag is expected, so master should not check and clear kLPI2C_MasterNackDetectFlag when remainingBytes is zero, in case FIFO is emptied when stop command has not been sent yet.
    - Fixed the bug that, during non-blocking transfer slave may nack master while master is busy filling tx FIFO, and NDF may not be handled properly.
- 2.1.10
  - Bug Fixes
    - MISRA C-2012 issue fixed.
      - Fixed rule 10.3, 14.4, 15.5.
    - Fixed unaligned access issue in LPI2C_RunTransferStateMachine.
    - Fixed uninitialized variable issue in LPI2C_MasterTransferHandleIRQ.
    - Used linked TCD to disable tx and enable rx in read operation to fix the issue that for platform sharing the same DMA request with tx and rx, during LPI2C read operation if interrupt with higher priority happened exactly after command was sent and before tx disabled, potentially both tx and rx could trigger dma and cause trouble.
    - Fixed MISRA issues.
      - Fixed rules 10.1, 10.3, 10.4, 11.6, 11.9, 14.4, 17.7.
    - Fixed the waitTimes variable not re-assignment issue for each byte read.
  - New Features
    - Added the IRQHandler for LPI2C5 and LPI2C6 instances.
  - Improvements
    - Updated the LPI2C_WAIT_TIMEOUT macro to unified name I2C_RETRY_TIMES.
- 2.1.9
  - Bug Fixes
    - Fixed Coverity issue of unchecked return value in I2C_RTOS_Transfer.
    - Fixed Coverity issue of operands did not affect the result in LPI2C_SlaveReceive and LPI2C_SlaveSend.
    - Removed STOP signal wait when NAK detected.
    - Cleared slave repeat start flag before transmission started in LPI2C_SlaveSend/LPI2C_SlaveReceive. The issue was that LPI2C_SlaveSend/LPI2C_SlaveReceive did not handle with the reserved repeat start flag. This caused the next slave to send a break, and the master was always in the receive data status, but could not receive data.

- 2.1.8
  - Bug Fixes
    - Fixed the transfer issue with LPI2C_MasterTransferNonBlocking, kLPI2C_TransferNoStopFlag, with the wait transfer done through callback in a way of not doing a blocking transfer.
    - Fixed the issue that STOP signal did not appear in the bus when NAK event occurred.
- 2.1.7
  - Bug Fixes
    - Cleared the stopflag before transmission started in LPI2C_SlaveSend/LPI2C_SlaveReceive. The issue was that LPI2C_SlaveSend/LPI2C_SlaveReceive did not handle with the reserved stop flag and caused the next slave to send a break, and the master always stayed in the receive data status but could not receive data.
- 2.1.6
  - Bug Fixes
    - Fixed driver MISRA build error and C++ build error in LPI2C_MasterSend and LPI2C_SlaveSend.
    - Reset FIFO in LPI2C Master Transfer functions to avoid any byte still remaining in FIFO during last transfer.
    - Fixed the issue that LPI2C_MasterStop did not return the correct NAK status in the bus for second transfer to the non-existing slave address.
- 2.1.5
  - Bug Fixes
    - Extended the Driver IRQ handler to support LPI2C4.
    - Changed to use ARRAY_SIZE(kLpi2cBases) instead of FEATURE COUNT to decide the array size for handle pointer array.
- 2.1.4
  - Bug Fixes
    - Fixed the LPI2C_MasterTransferEDMA receive issue when LPI2C shared same request source with TX/RX DMA request. Previously, the API used scatter-gather method, which handled the command transfer first, then the linked TCD which was pre-set with the receive data transfer. The issue was that the TX DMA request and the RX DMA request were both enabled, so when the DMA finished the first command TCD transfer and handled the receive data TCD, the TX DMA request still happened due to empty TX FIFO. The result was that the RX DMA transfer would start without waiting on the expected RX DMA request.
    - Fixed the issue by enabling IntMajor interrupt for the command TCD and checking if there was a linked TCD to disable the TX DMA request in LPI2C_MasterEDMACallback API.
- 2.1.3
  - Improvements
    - Added LPI2C_WATI_TIMEOUT macro to allow the user to specify the timeout times for waiting flags in functional API and blocking transfer API.
    - Added LPI2C_MasterTransferBlocking API.
- 2.1.2
  - Bug Fixes

    - In LPI2C_SlaveTransferHandleIRQ, reset the slave status to idle when stop flag was detected.
- 2.1.1
  - Bug Fixes
    - Disabled the auto-stop feature in eDMA driver. Previously, the auto-stop feature was enabled at transfer when transferring with stop flag. Since transfer was without stop flag and the auto-stop feature was enabled, when starting a new transfer with stop flag, the stop flag would be sent before the new transfer started, causing unsuccesful sending of the start flag, so the transfer could not start.
    - Changed default slave configuration with address stall false.
- 2.1.0
  - Improvements
    - API name changed:
      - LPI2C_MasterTransferCreateHandle -> LPI2C_MasterCreateHandle.
      - LPI2C_MasterTransferGetCount -> LPI2C_MasterGetTransferCount.
      - LPI2C_MasterTransferAbort -> LPI2C_MasterAbortTransfer.
      - LPI2C_MasterTransferHandleIRQ -> LPI2C_MasterHandleInterrupt.
      - LPI2C_SlaveTransferCreateHandle -> LPI2C_SlaveCreateHandle.
      - LPI2C_SlaveTransferGetCount -> LPI2C_SlaveGetTransferCount.
      - LPI2C_SlaveTransferAbort -> LPI2C_SlaveAbortTransfer.
      - LPI2C_SlaveTransferHandleIRQ -> LPI2C_SlaveHandleInterrupt.
- 2.0.0
  - Initial version.

LPIT

The current LPIT driver version is 2.0.2.

- 2.0.2
  - Improvements
    - Improved LPIT_SetTimerPeriod implementation, configure timeout value with LPIT ticks minus 1 generate more correct interval.
    - Added timeout value configuration check for LPIT_SetTimerPeriod, at least input 3 ticks for calling LPIT_SetTimerPeriod.
  - Bug Fixes
    - Fixed MISRA C-2012 rule 17.7 violations.
- 2.0.1
  - Bug Fixes
    - MISRA C-2012 issue fixed.
      - Fixed rules, containing: rule-10.3, rule-14.4, rule-15.5.
- 2.0.0
  - Initial version.

LPSPI

The current LPSPI driver version is 2.2.1.

- 2.2.1
  - Bug Fixes
    - Fixed bug in LPSPI_SetPCSContinous when disabling PCS continous mode.
- 2.2.0
  - Bug Fixes
    - Fixed bug in 3-wire polling and interrupt transfer that the received data is not correct and the PCS continous mode is not working.
- 2.1.0
  - Improvements
    - Improved LPSPI_SlaveTransferHandleIRQ to fill up TX FIFO instead of write one data to TX register which improves the slave transmit performance.
    - Added new functional APIs LPSPI_SelectTransferPCS and LPSPI_SetPCSContinous to support changing PCS selection and PCS continous mode.
  - Bug Fixes
    - Fixed bug in non-blocking and EDMA transfer APIs that kStatus_InvalidArgument is returned if user configures 3-wire mode and full-duplex transfer at the same time, but transfer state is already set to kLPSPI_Busy by mistake causing following transfer can not start.
    - Fixed bug when LPSPI slave using EDMA way to transfer, tx should be masked when tx data is null, otherwise in 3-wire mode which tx/rx use the same pin, the received data will be interfered.
- 2.0.5
  - Improvements
    - Added timeout mechanism when waiting certain states in transfer driver.
  - Bug Fixes
    - Fixed the bug that LPSPI can not transfer large data using EDMA.
    - Fixed MISRA 17.7 issues.
    - Fixed variable overflow issue introduced by MISRA fix.
    - Fixed issue that rxFifoMaxBytes should be calculated according to transfer width rather than FIFO width.
    - Fixed issue that completion flag was not cleared after transfer completed.
- 2.0.4
  - Bug Fixes
    - Fixed in LPSPI_MasterTransferBlocking that master rxfifo may overflow in stall condition.
    - Eliminated IAR Pa082 warnings.
    - Fixed MISRA issues.
      - Fixed rules 10.1, 10.3, 10.4, 10.6, 11.9, 14.2, 14.4, 15.7, 17.7.
- 2.0.3
  - Bug Fixes
    - Removed LPSPI_Reset from LPSPI_MasterInit and LPSPI_SlaveInit, because this API may glitch the slave select line. If needed, call this function manually.

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

LPTMR

The current LPTMR driver version is 2.1.1.

- 2.1.1
  - Improvements
    - Updated the characters from "PTMR" to "LPTMR" in "FSL_FEATURE_PTMR_HAS_NO_PRESCALER_CLOCK_SOURCE_1_SUPPORT" feature definition.
- 2.1.0
  - Improvements
    - Implement for some special devices' not supporting for all clock sources.
  - Bug Fixes
    - Fixed issue when accessing CMR register.
- 2.0.2
  - Bug Fixes
    - Fixed MISRA-2012 issues.
      - Rule 10.1.
- 2.0.1
  - Improvements
    - 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 2.5.3.

- 2.5.3

  - Bug Fixes
    - Fixed comments by replacing unused status flags kLPUART_NoiseErrorInRxDataRegFlag and kLPUART_ParityErrorInRxDataRegFlag with kLPUART_NoiseErrorFlag and kLPUART_ParityErrorFlag.
- 2.5.2
  - Bug Fixes
    - Fixed bug that when setting watermark for TX or RX FIFO, the value may exceed the maximum limit.
  - Improvements
    - Added check in LPUART_TransferDMAHandleIRQ and LPUART_TransferEdmaHandleIRQ to ensure if user enables any interrupts other than transfer complete interrupt, the dma transfer is not terminated by mistake.
- 2.5.1
  - Improvements
    - Use separate data for TX and RX in lpuart_transfer_t.
  - Bug Fixes
    - Fixed bug that when ring buffer is used, if some data is received in ring buffer first before calling LPUART_TransferReceiveNonBlocking, the received data count returned by LPUART_TransferGetReceiveCount is wrong.
- 2.5.0
  - Bug Fixes
    - Added missing interrupt enable masks kLPUART_Match1InterruptEnable and kLPUART_Match2InterruptEnable.
    - Fixed bug in LPUART_EnableInterrupts, LPUART_DisableInterrupts and LPUART_GetEnabledInterrupts that the BAUD[LBKDIE] bit field should be soc specific.
    - Fixed bug in LPUART_TransferHandleIRQ that idle line interrupt should be disabled when rx data size is zero.
    - Deleted unused status flags kLPUART_NoiseErrorInRxDataRegFlag and kLPUART_ParityErrorInRxDataRegFlag, since firstly their function are the same as kLPUART_NoiseErrorFlag and kLPUART_ParityErrorFlag, secondly to obtain them one data word must be read out thus interfering with the receiving process.
    - Fixed bug in LPUART_GetStatusFlags that the STAT[LBKDIF], STAT[MA1F] and STAT[MA2F] should be soc specific.
    - Fixed bug in LPUART_ClearStatusFlags that tx/rx FIFO is reset by mistake when clearing flags.
    - Fixed bug in LPUART_TransferHandleIRQ that while clearing idle line flag the other bits should be masked in case other status bits be cleared by accident.
    - Fixed bug of race condition during LPUART transfer using transactional APIs, by disabling and re-enabling the global interrupt before and after critical operations on interrupt enable register.
    - Fixed DMA/eDMA transfer blocking issue by enabling tx idle interrupt after DMA/eDMA transmission finishes.
  - New Features
    - Added APIs LPUART_GetRxFifoCount/LPUART_GetTxFifoCount to get rx/tx FIFO data count.

    - Added APIs LPUART_SetRxFifoWatermark/LPUART_SetTxFifoWatermark to set rx/tx FIFO water mark.
- 2.4.1
  - Bug Fixes
    - Fixed MISRA advisory 17.7 issues.
- 2.4.0
  - New Features
    - Added APIs to configure 9-bit data mode, set slave address and send address.
- 2.3.1
  - Bug Fixes
    - Fixed MISRA advisory 15.5 issues.
- 2.3.0
  - Improvements
    - Modified LPUART_TransferHandleIRQ so that txState will be set to idle only when all data has been sent out to bus.
    - Modified LPUART_TransferGetSendCount so that this API returns the real byte count that LPUART has sent out rather than the software buffer status.
    - Added timeout mechanism when waiting for certain states in transfer driver.
- 2.2.8
  - Bug Fixes
    - Fixed issue for MISRA-2012 check.
      - Fixed rule-10.3, rule-14.4, rule-15.5.
    - Eliminated Pa082 warnings by assigning volatile variables to local variables and using local variables instead.
    - Fixed MISRA issues.
      - Fixed rules 10.1, 10.3, 10.4, 10.8, 14.4, 11.6, 17.7.
  - Improvements
    - Added check for kLPUART_TransmissionCompleteFlag in LPUART_WriteBlocking, LPUART_TransferHandleIRQ, LPUART_TransferSendDMACallback and LPUART_-SendEDMACallback to ensure all the data would be sent out to bus.
    - Rounded up the calculated sbr value in LPUART_SetBaudRate and LPUART_Init to achieve more acurate baudrate setting. Changed osr from uint32_t to uint8_t since osr's bigest value is 31.
    - Modified LPUART_ReadBlocking so that if more than one receiver errors occur, all status flags will be cleared and the most severe error status will be returned.
- 2.2.7
  - Bug Fixes
    - Fixed issue for MISRA-2012 check.
      - Fixed rule-12.1, rule-17.7, rule-14.4, rule-13.3, rule-14.4, rule-10.4, rule-10.8, rule-10.3, rule-10.7, rule-10.1, rule-11.6, rule-13.5, rule-11.3, rule-13.2, rule-8.3.
- 2.2.6
  - Bug Fixes
    - Fixed the issue of register's being in repeated reading status while dealing with the IRQ routine.
- 2.2.5

  - Bug Fixes
    - Do not set or clear the TIE/RIE bits when using LPUART_EnableTxDMA and LPUART_EnableRxDMA.
- 2.2.4
  - Improvements
    - Added hardware flow control function support.
    - Added idle-line-detecting feature in LPUART_TransferNonBlocking function. If an idle line is 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 received data size. Before triggering the callback, data in the FIFO (if has FIFO) is read out, and no interrupt will be disabled, except for that the receive data size reaches 0.
    - Enabled the RX FIFO watermark function. With the idle-line-detecting feature enabled, users can set the watermark value to whatever you want (should be less than the RX FIFO size). Data is received and a callback will be triggered when data receive ends.
- 2.2.3
  - Improvements
    - Changed parameter type in LPUART_RTOS_Init struct from rtos_lpuart_config to lpuart_rtos_config_t.
  - Bug Fixes
    - Disabled LPUART receive interrupt instead of all NVICs when reading data from ring buffer. Otherwise when the ring buffer is used, receive nonblocking method will disable all NVICs to protect the ring buffer. This may has a negative effect on other IPs that are using the interrupt.
- 2.2.2
  - Improvements
    - Added software reset feature support.
    - Added software reset API in LPUART_Init.
- 2.2.1
  - Improvements
    - Added separate RX/TX IRQ number support.
- 2.2.0
  - Improvements
    - Added support of 7 data bits and MSB.
- 2.1.1
  - Improvements
    - Removed unnecessary check of event flags and assert in LPUART_RTOS_Receive.
    - Added code to always wait for RX event flag in LPUART_RTOS_Receive.
- 2.1.0
  - Improvements
    - Update transactional APIs.

LPUART_EDMA

The current LPUART_EDMA driver version is 2.4.0.

- 2.4.0
  - Refer LPUART driver change log 2.1.0 to 2.4.0

LPUART_FREERTOS

The current LPUART_FREERTOS driver version is 2.4.0.

- 2.4.0
  - Refer LPUART driver change log 2.1.0 to 2.4.0

MMDVSQ

The current MMDVSQ driver version is 2.0.3.

- 2.0.3
  - Bug Fixes
    - MISRA C-2012 issue fixed: rule 10.3, 10.4, and 14.4.
- 2.0.2
  - Bug fix:
    - Fixed MMDVSQ_GetExecutionStatus function get execution status wrong.
- 2.0.1
  - Other changes:
    - Changed name of MMDVSQ_GetDivideRemainder and MMDVSQ_GetDivideQuotient functions.
- 2.0.0
  - Initial version.

PDB

The current PDB driver version is 2.0.4.

- 2.0.4
  - Bug Fixes
    - Fixed violations of MISRA C-2012 rule 10.1 and 10.4.
- 2.0.3
  - Bug Fixes
    - Fixed violations of MISRA C-2012 rule 17.7.
- 2.0.2
  - Improvement:
    - Used macros in feature file instead of that in IO map.
- 2.0.1
  - Changed PDB register base array to const.
- 2.0.0
  - Initial version.

PMC

The current PMC driver version is 2.0.3.

- 2.0.3
  - Bug Fixes
    - Fixed the violation of MISRA C-2012 rule 11.3.
- 2.0.2
  - Bug Fixes
    - Fixed the violations of MISRA 2012 rules:
      - Rule 10.3.
- 2.0.1
  - Bug Fixes
    - Fixed MISRA issues.
      - Rule 10.8, Rule 10.3.
- 2.0.0
  - Initial version.

PORT

The current PORT driver version is 2.3.0.

- 2.3.0
  - New Features
    - Added new APIs for Electrical Fast Transient(EFT) detect.
    - Added new API to configure port voltage range.
- 2.2.0
  - New Features
    - Added new api PORT_EnablePinDoubleDriveStrength.
- 2.1.1
  - Bug Fixes
    - Fixed the violations of MISRA C-2012 rules: 10.1, 10.411.311.8, 14.4.
- 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 GPIO module.
- 2.0.2
  - Other Changes
    - Added feature guard macros in the driver.
- 2.0.1
  - Other Changes
    - Added "const" in function parameter.
    - Updated some enumeration variables' names.

PWT

The current PWT driver version is 2.0.1.

- 2.0.1
  - Bug Fixes
    - Fixed violations of MISRA C-2012 rules: 10.8, 10.3, 10.6.
- 2.0.0
  - Initial version.

RCM

The current RCM driver version is 2.0.4.

- 2.0.4
  - Bug Fixes
    - Fixed violation of MISRA C-2012 rule 10.3
- 2.0.3
  - Bug Fixes
    - Fixed violation of MISRA C-2012 rules.
- 2.0.2
  - Bug Fixes
    - Fixed MISRA issue.
      - Rule 10.8, rule 10.1, rule 13.2, rule 3.1.
- 2.0.1
  - Bug Fixes
    - Fixed kRCM_SourceSw bit shift issue.
- 2.0.0
  - Initial version.

RTC

The current RTC driver version is 2.2.1.

- 2.2.1
  - Bug Fixes
    - Fixed the issue of Pa082 warning.
    - Fixed the issue of bit field mask checking.
    - Fixed the issue of hard code in RTC_Init.
- 2.2.0
  - Bug Fixes
    - Fixed MISRA C-2012 issue.
      - Fixed rule contain: rule-17.7, rule-14.4, rule-10.4, rule-10.7, rule-10.1, rule-10.3.
    - Fixed central repository code formatting issue.
  - Improvements

        * Added an API for enabling wakeup pin.
- 2.1.0
  - Improvements
    * Added feature macro check for many features.
- 2.0.0
  - Initial version.

SIM

The current SIM driver version is 2.1.3.

- 2.1.3
  - Improvements
    * Updated function SIM_GetUniqueId to support different register names.
- 2.1.2
  - Bug Fixes
    * Fixed SIM_GetUniqueId bug that could not get UIDH.
- 2.1.1
  - Bug Fixes
    * Fixed violations of the MISRA C-2012 rules 10.1, 10.4
- 2.1.0
  - Improvements
    * Added new APIs: SIM_GetRfAddr() and SIM_EnableSystickClock().
- 2.0.0
  - Initial version.

SMC

The current SMC driver version is 2.0.7.

- 2.0.7
  - Bug Fixes
    * Fixed MISRA-2012 issue 10.3.
- 2.0.6
  - Bug Fixes
    * Fixed issue for MISRA-2012 check.
      · Fixed rule 10.3, rule 11.3.
- 2.0.5
  - Bug Fixes
    * Fixed issue for MISRA-2012 check.
      · Fixed rule 15.7, rule 14.4, rule 10.3, rule 10.1, rule 10.4.
- 2.0.4
  - Bug Fixes
    * When entering stop modes, used RAM function for the flash synchronization issue.

Application should make sure that, the RW data of fsl_smc.c is located in memory region which is not powered off in stop modes.

- 2.0.3
  - Improvements
    - Added APIs SMC_PreEnterStopModes, SMC_PreEnterWaitModes, SMC_PostExitWaitModes, and SMC_PostExitStopModes.
- 2.0.2
  - Bug Fixes
    - Added DSB before WFI while ISB after WFI.
  - Other Changes
    - Updated SMC_SetPowerModeVlpw implementation.
- 2.0.1
  - Other Changes
    - Updated for KL8x.
- 2.0.0
  - Initial version.

TRGMUX

The current TRGMUX driver version is 2.0.1.

- 2.0.1
  - Bug Fixes
    - Fixed violations of the MISRA C-2012 rules 10.1, 10.3, 10.8.
- 2.0.0
  - Initial version.

TSI_V5

The current TSI_V5 driver version is 2.3.0.

- 2.3.0
  - Other Changes
    - Changed the TSI SINC cutoff divider number.
- 2.2.0
  - Improvements
    - Extended enableShield items from tsi_selfCap_config_t structure to cover three shields in the ke17z series.
    - Added interface for getting instance from TSI base address and apply it for clock and IRQ enable/disable.
- 2.1.2
  - Bug Fixes
    - Fixed the violations of MISRA C-2012 rules: 10.1, 10.8.
- 2.1.1

  - Improvements
    - Improved the module's noise immunity in mutual cap mode by setting M_TRIM2[0] to 1 in TSI_MUL1 register.
  - Bug Fixes
    - Fixed the violations of MISRA C-2012 rules:
      - Rule 10.1, 10.3, 10.4, 10.8, 12.2, 14.4, 17.7.
- 2.1.0
  - Bug Fixes
    - Fixed incorrect TSI SSC clock calculation.
- 2.0.1
  - Improvements
    - Added functions for M_TX_USED bitfield for ke16z only (Unused TX mutual pins can work as GPIO).
- 2.0.0
  - Initial version.

WDOG32

The current WDOG32 driver version is 2.0.4.

- 2.0.4
  - Improvements
    - To ensure that the reconfiguration is inside 128 bus clocks unlock window, put all reconfiguration APIs in quick access code section.
- 2.0.3
  - Bug Fixes
    - Fixed the noncompliance issue of the reference document.
      - Waited until for new configuration to take effect by checking the RCS bit field.
      - Waited until for registers to be unlocked by checking the ULK bit field.
  - Improvements
    - Added 128 bus clocks delay ensures a smooth transition before restarting the counter with the new configuration when there is no RCS status bit.
- 2.0.2
  - Bug Fixes
    - MISRA C-2012 issue fixed.
      - Fixed rules, containing: rule-10.3, rule-14.4, rule-15.5.
    - Fixed the issue of the inseparable process interrupted by other interrupt source.
      - WDOG32_Refresh
- 2.0.1
  - Bug Fixes
    - WDOG must be configured within its configuration time period.
      - Added WDOG32_Init API to quick access section.
      - Defined register variable in WDOG32_Init API.
- 2.0.0

- Initial version.

2 Middleware Change Log

FatFs for MCUXpresso SDK

Current version is FatFs R0.14b_rev0.

- R0.14b_rev0
  - Upgraded to version 0.14b
- R0.14a_rev0
  - Upgraded to version 0.14a
  - Applied patch ff14a_p1.diff and ff14a_p2.diff
- R0.14_rev0
  - Upgraded to version 0.14
  - Applied patch ff14_p1.diff and ff14_p2.diff
- R0.13c_rev0
  - Upgraded to version 0.13c
  - Applied patches ff_13c_p1.diff,ff_13c_p2.diff, ff_13c_p3.diff and ff_13c_p4.diff.
- R0.13b_rev0
  - Upgraded to version 0.13b
- 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.

FreeMASTER Communication Driver

Current version is 3.0.4. Visit https://www.nxp.com/freemaster for more information. Reach out for a support at https://community.nxp.com/community/freemaster.

- 3.0.0
  - Initial version of FreeMASTER driver reworked from a standalone package to MCUXpresso SDK middleware.
  - This driver version supports new version V4 of FreeMASTER serial communication protocol.

  - Supports UART, LPUART, USART, MINIUSART, FlexCAN, USB-CDC and JTAG/BDM communication.
  - Initial version was tested with the following boards: evkmimxrt1060, frdmk64f, frdmke15z, frdmkl28z, lpcxpresso54628 lpcxpresso55s69, lpcxpresso845max and twrk64f120m.
  - Use with FreeMASTER PC Host tool version 2.5 or later.
- 3.0.1
  - FreeMASTER driver extended to support wide range of Kinetis, LPC and i.MX-RT platforms.
  - Low-level communication drivers also available for few non-SDK NXP platforms like S12Z, S32x and more.
  - Use with FreeMASTER PC Host tool version 3.0 or later.
- 3.0.2
  - FreeMASTER driver support of DSC56F800EX and S12 platforms extended.
  - Removed dependency on C99 compiler features.
  - Use with FreeMASTER PC Host tool version 3.0.2 or later.
- 3.0.3
  - General update for SDK 2.9.0
  - fmstr_any demo added to selected platforms - use with MCUXpresso SDK and FreeMASTER peripheral configuration tool.
  - New example.pmp project file embedded into application flash storage.
  - USB-CDC implementation fixed, new JTAG EOnCE communication interface added to DSC 56F800E family.
  - Use with FreeMASTER PC Host tool version 3.0.3 or later. Version 3.1.x is recommended.
- 3.0.4
  - Fixed component dependency logic of FreeMASTER driver.
  - Use with FreeMASTER PC Host tool version 3.1.x
- 3.0.5
  - General update for SDK 2.11 and 2.12
  - New TCP and UDP support with lwIP stack
  - New communication over Segger RTT interface
  - Add fmstr_net and fmstr_wifi examples for selected i.MX-RT platforms
  - Add fmstr_rtt example for selected platforms
  - Fixed negative recorder threshold trigger processing

MOTOR_CONTROL for KSDK

Current version is 1.1.0

- 1.1.0
  - Initial version.

RTCESL for KSDK

Current version is 4.3

- 4.3
  - Initial version.

TOUCH_SENSING for KSDK

Current version is 1.1.0

- 1.1.0
  - Initial version.

3 Component Change Log

SERIAL_MANAGER

The current Serial_Manager component version is 1.0.2.

- 1.0.2
  - Add SerialManager_WriteTimeDelay()/SerialManager_ReadTimeDelay() for serial manager's read/write non-blocking mode.
- 1.0.1
  - Add prefixing fsl_component_xxx/fsl_adapter_xxx.
- 1.0.0
  - Initial version

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support