NXP Semiconductors
User's Guide

Document Number: MCUXSDKKW36ARN
Rev 1, 01/2019

MCUXpresso SDK Release Notes Supporting MKW34/35/36 Derivatives

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

MCUXpresso SDK is the evolution of MCUXpresso SDK 1.x into a more optimized software solution. MCUXpresso SDK eliminates the need for a separate HAL and Peripheral Driver, replacing these two layers with a single driver for each peripheral. The single driver provides both the low-level functionality of the HAL and the non-blocking interrupt-based functionality of the Peripheral Driver, enabling customers to select the right level of abstraction for their solution. Peripheral drivers in MCUXpresso SDK also eliminate external software dependencies. The Operating System Abstraction, Power Manager, and Clock Manager are no longer required by the MCUXpresso SDK drivers.

The existing MQX™ RTOS support has been deprecated to focus on support of FreeRTOS OS.

The Real Time Control Embedded Software Library (RTCESL) and motor control examples for PMSM and BLDC are added to the middleware layer.

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:

- Kinetis Design Studio IDE v3.2
- IAR Embedded Workbench for Arm version 8.11.1
- MDK-Arm Microcontroller Development Kit (Keil)® 5.23
- Makefiles support with GCC revision v5-2016-q3 from ARM Embedded
- MCUXpresso IDE v10.3.1

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
FRDM-KW36	MKW36A512VFP4, MKW36Z512VFP4, MKW35A512VFP4, MKW36A512VFT4, MKW35A512VFT4, MKW34A512VFT4, MKW36A512VHT4, MKW35Z512VHT4, **MKW36Z512VHT4**

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
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
RTOS examples	<install_dir>/boards/<board_name>/rtos_examples
Documentation	<install_dir>/docs
Middleware	<install_dir>/middleware
DMA manager	<install_dir>/middleware/dma_manager
SDMMC card driver	<install_dir>/middleware/sdmmc
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
Utilities such as debug console	<install_dir>/devices/<device_name>/utilities
RTOS Kernel Code	<install_dir>/rtos
Tools	<install_dir>/tools

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

6.3 File system

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

6.4 RTOS

The MCUXpresso SDK is integrated with FreeRTOS OS.

6.5 CMSIS

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

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 Flash configuration field programming in IAR

NMI is disabled in the startup code in flash configuration field (FlashConfig section) for a better experience with the FRDM-KW36 board. To enable programming of the flash configuration field in IAR, it may be necessary to add a parameter to the flashloader configuration. The configuration is accessible in the debugger settings of the target. Select the target, then in the menu command 'Project -> Options -> Debugger', select the 'Download' tab, check the 'Override' default .board file, and click the 'Edit' button. Select the flash area starting with address 0x0 and click the 'Edit' button. Add the --enable_config_write parameter to the 'Extra' parameters field.

Chapter 9
Revision history

This table summarizes revisions made to this document.

Table 4. Revision history

Revision number	Date	Substantive changes
0	06/2017	Initial release
1	01/2019	Updates for KW34/35/36 release

10 Change Log - Peripheral drivers

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

- 2.0.0
  - Initial version
- 2.0.1
  - Changes
    - Added static to global CMT variables.

COP

The current COP driver version is 2.0.0

- 2.0.0
  - Initial version

DCDC

The current DCDC driver version is 2.0.1

- 2.0.0
  - Initial version
- 2.0.1
  - Add to support the devices with "1P45" or "1P5" registers' naming for the lower voltage's power output.

DMAMUX

The current DMAMUX driver version is 2.0.2

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

DSPI

The current DSPI driver version is 2.1.4

- 2.1.0
  - New features:
    - Added transfer prefix to transactional APIs
- 2.1.1
  - Bug fix:

        - Set the EOQ (End Of Queue) bit to TRUE for the last transfer in transactional APIs
- 2.1.2
    - Bug fix:
        - The DSPI_MasterTransferBlocking function hangs in corner cases, for example, when bitsPerFrame is 4 and 6 and in the kDSPI_MasterPcsContinuous transfer mode
- 2.1.3
    - Bug Fix:
        - DSPI eDMA driver doesn't support the odd transfer data size and the bitsPerFrame greater than 8.
    - Optimization:
        - Added the #ifndef/#endif to allow users to change the default tx value at compile time.
- 2.1.4
    - Bug fix:
        - DSPI EDMA driver: The DSPI instance that has separated DMA request source can transfer up to 32767 byte data in one DSPI_MasterTransferEDMA() transfer now

eDMA

The current eDMA driver version is 2.1.1

- 2.0.0
    - Initial version
- 2.0.1
    - Bug fix:
        - Fixed the issue where an eDMA callback does not check a valid status in the EDMA_HandleIRQ API
- 2.0.2
    - Bug fix:
        - Fixed the incorrect minorLoopBytes type definition in the _edma_transfer_config structure. Defined the minorLoopBytes as uint32_t instead of uint16_t
- 2.0.3
    - Bug fix:
        - Fixed the incorrect pubweak IRQHandler name issue, which causes re-definition build errors when a client sets his/her own IRQHandler, by changing the 32-channel IRQHandler name to DriverIRQHandler
- 2.0.4
    - Improvement:
        - Added support for SoCs with multiple eDMA instances.
        - Added the pubweak DriverIRQHandler for the KL28T DMA1 and MCIMX7U5_M4.
- 2.0.5
    - Improvement:
        - Added the pubweak DriverIRQHandler for the K32H844P (16 channels shared).
- 2.1.0
    - Improvement:
        - Changed the EDMA_GetRemainingBytes API to EDMA_GetRemainingMajorLoopCount because of the eDMA IP limitation (see API comments/note for details).
- 2.1.1
    - Improvement:
        - Added documentation of the eDMA data flow when scatter/gather is implemented for the EDMA_HandleIRQ API.
        - Updated and corrected comments in the EDMA_HandleIRQ API and edma_handle_t struct.

Flash

The current Flash driver version is 2.3.1

- 2.0.0
    - Initial version
- 2.1.0
    - New features:
        - Support for the FTFL device in FLASH_Swap API

        - Support for various pflash start addresses
        - Added support for KV58 in the cache clear function
    - Bug fix
        - Compiled execute-in-RAM functions as a PIC binary code for driver use
        - Added missed FlexRAM properties
        - Fixed an unaligned variable issue for the execute-in-RAM function code array
- 2.2.0
    - New features:
        - Support FTFL device in FLASH_Swap API
        - Support various pflash start addresses
        - Add support for KV58 in cache clear function
        - Add support for device with secondary flash (KW40)
    - Bug fix
        - 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

- 2.3.0
    - New features:
        - Add support for device with LP flash (K3S/G)
        - Add flash prefetch speculation APIs
    - Improvement
        - Refine flash_cache_clear function
        - Reorganize the member of flash_config_t struct
- 2.3.1
    - Bug fix
        - Unified Flash IFR design from K3
        - New encoding rule for K3 flash size

FlexCAN

The current FlexCAN driver version is 2.2.0

- 2.0.0
    - Initial version
- 2.1.0
    - Bug fix
        - Fix wrong function name spelling: FlEXCAN_XXX() -> FLEXCAN_XXX();
        - Move Freeze Enable/Disable setting from FLEXCAN_Enter/ExitFreezeMode() to FLEXCAN_Init();
        - Fix wrong helper macro values
    - Other changes
        - Hide FLEXCAN_Reset() to user
        - Use NDEBUG macro to wrap FLEXCAN_IsMbOccupied() function instead of DEBUG macro
- 2.2.0
    - Improvement
        - Add FSL_FEATURE_FLEXCAN_HAS_SUPPORT_ENGINE_CLK_SEL_REMOVE feature to support SoCs without CAN Engine Clock selection in FlexCAN module
        - Add FlexCAN Serial Clock Operation to support i.MX SoCs

GPIO

The current GPIO driver version is 2.1.1

- 2.1.0
    - API Interface Change:

        - Added "pins" or "pin" to some API names
        - Renamed the "GPIO_PinConfigure" to "GPIO_PinInit"
- 2.1.1
    - API Interface Change
        - Added API for the check attribute bytes

I2C

The current I2C driver version is 2.0.3

- 2.0.1
    - New features
        - Added a double buffer enable configuration for SoCs which have the DFEN bit in S2 register
        - Added the flexible transmit/receive buffer size support in I2C_SlaveHandleIRQ
        - Added the start flag clear address match and release bus operation in I2C_SlaveWrite/ReadBlocking API
    - Bug fix:
        - Updated the kI2C_SlaveRepeatedStartEvent to kI2C_SlaveStartEvent
- 2.0.2
    - Bug Fix:
        - Fixed the issue that occurs in master receive and slave transmit mode with no stop flag and master can't start a next transfer because it can't send out restart signal
        - Fixed a data transfer out of order issue which occurs because of a memory barrier
    - New Features:
        - Added an address nak event for the master.
        - Added a general call event for the slave.

- 2.0.3
    - Bug fix
        - Remove enableHighDrive member in the master/slave configuration structure because the operation to HDRS bit is useless. The user needs to use DSE bit in port register to configure the high drive capability.
        - Add reset registers operation in I2C_MasterInit and I2C_SlaveInit APIs, and fix the issue that I2C could not switch between master and slave mode.
        - Improve slave IRQ handler to handle the corner case that stop flag and address match flag come synchronously.

LIN

The current LIN driver version is 1.0.0

- 1.0.0
    - Initial version

LLWU

The current LLWU driver version is 2.0.1

- 2.0.0
    - Initial version
- 2.0.1
    - Changes:
        - Updated for KL8x

- 2.1.1
    - Bug fix:
        - Disable the auto stop feature in the EDMA driver. Previously the autostop feature is enabled at transfer when transfer with stop flag. If the previous transfer is without stop flag, then when starting a new transfer

with stop flag, because the auto stop feature is enabled, the stop flag sends before starting the new transfer and the start flag can not successfully sent, so the transfer cannot start.
- Change default slave configuration with address stall false

LPTMR

The current LPTMR driver version is 2.0.1

- 2.0.0
  - Initial version
- 2.0.1
  - Driver update
    - Update the LPTMR driver due to the register LPTMRx_CMR/CNR in some devices that have become 32-bit, so that the updated LPTMR driver supports the 32 bit CNR and CMR register.

LPUART

The current LPUART driver version is 2.2.3

- 2.1.0
  - Updated transactional APIs
- 2.1.1
  - Removed the needless check of event flags and assert in LPUART_RTOS_Receive
  - Wait always for RX event flag in LPUART_RTOS_Receive
- 2.2.0
  - Added seven data bits and MSB support
- 2.2.1
  - Added a separate RX and TX IRQ number support
- 2.2.2
  - Added software reset feature support.
  - Added software reset API to LPUART_Init().
- 2.2.3
  - Changed the parameter type in the LPUART_RTOS_Init() struct rtos_lpuart_config --> lpuart_rtos_config_t.

PIT

The current PIT driver version is 2.0.0

- 2.0.0
  - Initial version

PMC

The current PMC driver version is 2.0.0

- 2.0.0
  - Initial version

PORT

The current PORT driver version is 2.0.2

- 2.0.1
  - Changes:
    - Added "const" in function parameters
    - Updated enumeration variable names
- 2.0.2
  - Changes:
    - Added feature guard macros in the driver

RCM

The current RCM driver version is 2.0.1

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

RTC

The current RTC driver version is 2.0.0

- 2.0.0
  - Initial version

SIM

The current SIM driver version is 2.0.0

- 2.0.0
  - Initial version

SMC

The current SMC driver version is 2.0.3

- 2.0.0
  - Initial version
- 2.0.1
  - Changes:
    - Updated for KL8x
- 2.0.2
  - Bug fix:
    - Added DSB before WFI and ISB after WFI
  - Changes:
    - Updated the SMC_SetPowerModeVlpw implementation
- 2.0.3
  - Add APIs SMC_PreEnterStopModes, SMC_PreEnterWaitModes, SMC_PostExitWaitModes, and SMC_PostExitStopModes

TPM

The current TPM driver version is 2.0.2

- 2.0.0
  - Initial version
- 2.0.1
  - Bug fix:
    - Fixed the TPM_UpdateChnIEdgeLevelSelect ACK wait issue.
    - Fixed the TPM_SetupdualEdgeCapture failure to set the FILTER register.
    - Fixed the TPM_UpdateChnEdgeLevelSelect ACK wait issue.
- 2.0.2
  - Bug fix:
    - Fixed issues in TPM_SetupPwm/TPM_UpdateChnlEdgeLevelSelect /TPM_SetupInputCapture/ TPM_SetupOutputCompare/TPM_SetupDualEdgeCapture functions wait acknowledgement when the channel is disabled.

VREF

The current VREF driver version is 2.1.0

- 2.1.0
  - Added new functions:
    - Added VREF_SetTrim2V1Val() and VREF_GetTrim2V1Val() functions to supply 2V1 output mode.

CLOCK

The current CLOCK driver version is 2.2.1

- 2.0.0
  - Initial version
- 2.1.0
  - Changes:
    - Merged the fsl_mcg and fsl_osc into fsl_clock
- 2.2.0
  - New features:
    - [KPSDK-9157] Updated the CLOCK_SetFeiMode/CLOCK_SetFbiMode/CLOCK_BootToFeiMode() to support set MCG_C4[DMX32]=1 in FEI/FBI modes
  - Bug fix:
    - Updated the IP_CLOCKS array, removed unused gates, and added missing gates
- 2.2.1
  - Bug fix:
    - Fix the issue that MCG could not switch to FEE/FBE/PBE modes when the OSCERCLK clock is not enabled

11 Change Log - Middleware

DMA Manager

The current DMA Manager driver version is 2.1.0

- 2.0.0
  - Initial version
- 2.1.0
  - Update DMA manager interface to support dynamic configure the manage area. This is used for platform with multiple cores

FatFs

The current FatFs driver version is R0.12b

- R0.12b_rev0
- R0.11a
  - Added glue functions for low level drivers (SDHC, SDSPI, RAM, and MMC) and modified the diskio.c file
  - Added RTOS wrappers to make FatFs thread-safe. Modified the syscall.c file
  - Renamed ffconf.h file to ffconf_template.h file. Each application should contain its own ffconf.h file
  - Conditional compilation of physical disk interfaces in diskio.c

SDMMC

The current SDMMC driver version is 2.1.2

- 2.1.0
  - Bug fix:
    - Changed the callback mechanism when sending a command
    - Fixed the performance low issue when transferring data
  - Changes:
    - Changed the name of error codes returned by an internal function
    - Merged all host-related attributes into one structure
    - Optimized the function to set a maximum data bus width for the MMC card
- 2.1.1
  - Bug fix:
    - Fixed the block range boundary error when transferring data to the MMC card
    - Fixed the bit mask error in the SD card when switching to a high-speed function
  - Changes:
    - Added an error code to indicate that SDHC ADMA1 transfer type is not supported
    - Optimized the SD card initialization function

- 2.1.2
  - New feature
    - Add fsl_host.h to provide prototype to adapt different controller IPs(SDHC/SDIF)
    - Add adaptor code in sdmmc/port folder to adapt different host controller IPs with different transfer modes(int/polling/freertos). Application inlude different adpator code to make application simplier
    - Adaptor code provides HOST_Init/HOST_Deinit/CardInsertDetect APIs to do host controller initialize and transfer function configuration. SDMMC card stack uses adaptor code inside stack to wait card insert and configure host when calling card init APIs (SD_Init/MMC_Init/SDIO_Init)
    - So this change requires user to include host adaptor code into application. If not, link errors for cannot find the definition of HOST_Init/HOST_Deinit/CardInsertDetect will appear
  - New feature
    - Improve SDMMC to support SD v3.0 and EMMC v5.0
  - Bug fix:
    - Fix wrong comparison between count and length in MMC_ReadBlocks/MMC_WriteBlocks

12 Change Log - RTOS

FreeRTOS OS

The current version is FreeRTOS OS 9.0.0. The original package is available at freertos.org.

- 9.0.0_rev2
  - New features:
    - Enabled MCUXpresso thread aware debugging. Added freertos_tasks_c_additions.h and configINCLUDE_FREERTOS_TASK_C_ADDITIONS_H and configFRTOS_MEMORY_SCHEME macros
- 9.0.0_rev1
  - New features:
    - Enable -flto optimization in GCC by adding **attribute**((used)) for vTaskSwitchContext
    - Enable KDS Task Aware Debugger. Apply FreeRTOS patch to enable configRECORD_STACK_HIGH_ADDRESS macro. Modified files are task.c and FreeRTOS.h
- 9.0.0_rev0
  - New features:
    - Example freertos_sem_static
    - Static allocation support RTOS driver wrappers

- Other changes:
  - Tickless idle rework. Support for different timers is in separated files (fsl_tickless_systick.c, fsl_tickless_lptmr.c)
  - Remove configuration option configSYSTICK_USE_LOW_POWER_TIMER. Low power timer is now selected by linking of appropriate file fsl_tickless_lptmr.c
  - Remove configOVERRIDE_DEFAULT_TICK_CONFIGURATION in RVDS port. Use of **attribute**((weak)) is preferred solution. Not same as _weak
- 8.2.3
  - New features:
    - Added tickless idle mode support
    - Added a template application for Kinetis Expert (KEx) tool (template_application)
  - Changes:
    - Reduced the folder structure to keep only Kinetis-related information

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:
Document identifier: MCUXSDKKW36ARN

arm