NXP Semiconductors
User's Guide

Document identifier: MCUXSDKMIMXRT106ARN
Rev. 0, 02/2020

MCUXpresso SDK Release Notes for SLN-ALEXA-IOT

Contents

Chapter 1 Overview..... 3

Chapter 2 MCUXpresso SDK..... 4

Chapter 3 Development tools..... 5

Chapter 4 Supported development systems..... 6

Chapter 5 Release contents..... 7

Chapter 6 MCUXpresso SDK release package..... 9

Chapter 7 MISRA compliance..... 11

Chapter 8 Known issues..... 13

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 FatFs, other middleware packages,. In addition to the base enablement, the MCUXpresso SDK is augmented with demo applications, driver example projects, and API documentation to help the customers quickly leverage the support of the MCUXpresso SDK.

For more details about MCUXpresso SDK, see the MCUXpresso SDK homepage MCUXpresso-SDK: Software Development Kit.

Chapter 2
MCUXpresso SDK

As part of the MCUXpresso software and tools, MCUXpresso SDK is the evolution of Kinetis SDK v2.7.0, 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 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 easy cloning of existing SDK examples and demos, allowing users to leverage the existing software examples provided by the SDK for their own projects.

NOTE

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

Chapter 3
Development tools

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

- Makefiles support with GCC revision 8-2019-q3 GCC8 from Arm Embedded
- MCUXpresso IDE v11.1.0

Chapter 4
Supported development systems

This relase supportes boards and devices listed in Table 1. The boards and devices in bold were tested in this release.

Table 1. Supported MCU devices and development boards

Development boards	MCU devices
SLN-ALEXA-IOT	MIMXRT106ADVL6A

Chapter 5
Release contents

Table 2 provides an overview of the MCUXpresso SDK release package contents and locations.

Table 2. Release contents

Deliverable	Location
Amazon Wakeword Library	<install_dir>/middleware/amzn_ww
AVS for AWS IoT	<install_dir>/middleware/aws_ais
AWS IoT	<install_dir>/middleware/aws_iot
Boards	<install_dir>/boards
Boot Applications	<install_dir>/boards/<board_name>/sln_boot_apps
cJSON	<install_dir>/middleware/cjson
CMSIS Arm Cortex®-M header files, DSP library source	<install_dir>/CMSIS
CMSIS drivers	<install_dir>/devices/<device_name>/cmsis_drivers
Cypress BLE examples	<install_dir>/boards/<board_name>/ble_cypress_examples
Cypress Wiced SDK (WiFi, BLE)	<install_dir>/middleware/wiced
Cypress WiFi stack examples	<install_dir>/boards/<board_name>/wifi_cypress_examples
Demo applications	<install_dir>/boards/<board_name>/demo_apps
Documentation	<install_dir>/docs
Driver examples	<install_dir>/boards/<board_name>/driver_examples
Driver, SoC header files, extension header files and feature header files, utilities	<install_dir>/devices/<device_name>
FatFS stack	<install_dir>/middleware/fatfs
jsmn	<install_dir>/middleware/aws_iot/external_libs/jsmn
LittleFS	<install_dir>/middleware/littlefs
lwIP Documentation	<install_dir>/docs/lwip
lwIP stack	<install_dir>/middleware/lwip
mbed TLS	<install_dir>/middleware/mbedtls
MCU Streamer	<install_dir>/middleware/audio_streamer
Peripheral Drivers	<install_dir>/devices/<device_name>/drivers
RTOS Kernel Code	<install_dir>/rtos
SLN Intelligence Toolbox	<install_dir>/middleware/mcu_voice_libs_public
Solutions IoT Common Platform	<install_dir>/middleware/sln_iot_common_platform
Tools	<install_dir>/tools

Table continues on the next page...

Table 2. Release contents (continued)

Deliverable	Location
USB stack	`<install_dir>/middleware/usb`
Utilities such as debug console	`<install_dir>/devices/<device_name>/utilities`
Voice Examples	`<install_dir>/boards/<board_name>/sln_voice_examples`

Chapter 6
MCUXpresso SDK release package

The MCUXpresso SDK release package content is 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 the whole software enablement available for the specific System-on-Chip (SoC) subfamily. This folder includes clock-specific implementation, device register header files, device register feature header files, 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 standard debug console.

The device-specific header files provide a direct access to the microcontroller peripheral registers. The device header file provides an overall SoC memory mapped register definition. The folder also includes the feature header file for each peripheral on the microcontroller.

The toolchain folder contains the startup code and linker files for each supported toolchain. The startup code is a CMSIS compliant startup code 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 USB stack

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

6.2.2 TCP/IP stack

The lwIP TCP/IP stack is pre-integrated with MCUXpresso SDK and runs on top of the MCUXpresso SDK Ethernet driver with Ethernet-capable devices/boards.

6.2.3 Cypress Wiced SDK (WiFi, BLE)

The MCUXpresso SDK provides integration with Cypress Wiced SDK supporting the Murata Type 1DX and Azurewave AW-NM372SM modules based on the CYW4343W and CYW43438 processors.

6.2.4 File system

The FatFs file system is integrated with the 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.

6.2.5 RTOS

The MCUXpresso SDK is integrated with FreeRTOS OS.

6.2.6 CMSIS

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

6.2.7 AVS for AWS IoT

The MCUXpresso SDK is integrated with device side middleware to support AVS for AWS IoT.

Chapter 7
MISRA compliance

All MCUXpresso SDK drivers comply to MISRA 2012 rules with exceptions in Table 3.

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	Typedefs that indicate size and signedness should be used in place of the basic numerical types.
Directive 4.8	If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden.
Directive 4.9	A function should be used in preference to a function-like macro where they are interchangeable.
Directive 4.13	Functions which are designed to provide operations on a resource should be called in an appropriate sequence.
Rule 1.2	Language extensions should not be used.
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 2.6	A function should not contain unused label declarations.
Rule 2.7	There should be no unused parameters in functions.
Rule 4.2	Trigraphs should not be used.
Rule 5.1	External identifiers shall be distinct.
Rule 5.4	Macro identifiers shall be distinct.
Rule 5.9	Identifiers that define objects or functions with internal linkage should be unique.
Rule 8.7	Functions and objects should not be defined with external linkage if they are referenced in only one translation unit.
Rule 8.9	An object should be defined at block scope if its identifier only appears in a single function.
Rule 8.11	When an array with external linkage is declared, its size should be explicitly specified.
Rule 8.13	A pointer should point to a const-qualified type whenever possible.
Rule 10.5	The value of an expression should not be cast to an inappropriate essential type.

Table continues on the next page...

Table 3. MISRA exceptions (continued)

Exception rules	Description
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 12.1	The precedence of operators within expressions should be made explicit.
Rule 12.3	The comma operator should not be used.
Rule 12.4	Evaluation of constant expressions should not lead to unsigned integer wrap-around.
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 15.4	There should be no more than one break or go to statement used to terminate any iteration statement.
Rule 17.5	The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements.
Rule 17.8	A function parameter should not be modified.
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 identifieror reserved macro name.
Rule 21.2	A reserved identifier or macro name shall not be declared.
Rule 21.12	The exception handling features of `<fenv.h>` should not be used.

Chapter 8
Known issues

8.1 Maximum file path length in Windows 7® operating system

The 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 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).

8.3 New Project Wizard compile failure

The following components request the user to manually select other components that they depend upon in order to compile. These components depend on several other components and the New Project Wizard (NPW) is not able to decide which one is needed by the user.

NOTE

`xxx` means core variants, such as, `cm0plus`, `cm33`, `cm4`, `cm33_nodsp`.

Components: Assert, assert_cm0plus, assert_xxx, assert_lite, baremetal, button, codec_i2c, codec_i2c_xxx, debug_console, debug_console_xxx, debug_console_lite, dialog7212, led, misc_utilities, panic, serial_manager, serial_manager_xxx, serial_manager_swo, serial_manager_swo_xxx, serial_manager_uart, serial_manager_uart_xxx, serial_manager_usb_cdc, serial_manager_usb_cdc_xxx, sgtl_adapter, sgtl5000, shell, shell_xxx, timer_manager, wm8904, wm8904_xxx, wm8904_adapter, wm8904_adapter_xxx, wm8960, wm8960_adapter, xip_device.

8.4 RAM targets build issue in CMSIS bsp pack

Because CMSIS pack does not support different macro definitions for different targets, all RAM targets for projects inside CMSIS BSP PACKs for RT10XX boards will get the same macro definitions with Flash targets, resulting in build failure. To pass build for RAM targets, manually update the XIP_EXTERNAL_FLASH and XIP_BOOT_HEADER_ENABLE value to 0 in RTE_Components.h.

8.5 Known SDK issues

Table 4 lists the known issues in v1.1.0 of SLN-ALEXA-IOT SDK.

Table 4. Known SLN-ALEXA-SDK issues

Issue #	Description	Impact	Workaround
VOIS-960	AFE License expiry after 50 hours	SDK Contains evaluation version of Audio Front End.	Request production version of Audio Front End from NXP.
VOIS-961	Secure Boot performance issues running 3 microphones	Microphones become unresponsive when executing from encrypted XIP	Configure board for two microphones using the

Table continues on the next page...

Table 4. Known SLN-ALEXA-SDK issues (continued)

Issue #	Description	Impact	Workaround
			MACROs in `pdm_pcm_definitions.h`
VOIS-954	Failed MSD for wrong bank shows **File to large** error in Windows	Attempting to update the device through USB MSD using the wrong flash bank target results in **File to large** Windows error.	None.
VOIS-963	WiFi loss during OTA results in re-subscribe failure and board reset	If WiFi link is lost during OTA update the device fails to re-subscribe to all topics after WiFi link re-establishes. This causes OTA job to abort and reset the device.	None.
VOIS-962	`INVALID_REQUEST`: OVERRUN exception sent after speaker is closed	Sometimes an OVERRUN can be sent to the service after a speaker session has closed. This can cause out of sequence errors on subsequent speaker interactions.	None
VOIS-971	During UNDERRUN, a short response from Alexa is cut off at the end	Sometimes during an UNDERRUN, speaker is stopped by EOS from streamer until new packets arrive. There is short period when a race between Stop Streamer and checking if playback has ended.	None
VOIS-365	Alexa does not respond to wake word after debugger interaction	In MCUXpresso IDE, setting a break point outside of execution flow can stop wake word detection functionality	Set break points before execution.
VOIS-699	TIMED OUT send msg to streamer failed when using semihosting in `ais_demo`.	Streamer error logs to semihosting cause device to freeze	Avoid using semihosting with applications using MCU Streamer.
VOIS-820	Missing check to image vector table to see if the ResetISR address is correct	Bad image can be stored due to missing check for vector table validity	Always use valid images when performing OTA or MSD updates.
VOIS-898	Secured board encountered distortions while playing **ready sound** and alerts sound	Sound quality issues have been observed on a board running secure boot.	None.
VOIS-908	Push to talk button doing volume down on ETH build	Push to talk button can change volume on ethernet builds	None.
VOIS-917	Missing bootstrap fallback mechanism for situations when both bootloader and application are corrupted	If bootloader and application are corrupted, bootstrap won't be able to roll back	None.
VOIS-202	Alexa responds to speaking own name	When asking Alexa it's own name, it will begin to respond to the **Alexa** in the response.	None.

Table continues on the next page...

Table 4. Known SLN-ALEXA-SDK issues (continued)

Issue #	Description	Impact	Workaround
VOIS-575	RT DMA module does not support 16-byte transfers - gap for supporting 3/4 mics on SAI1	Using more than 3 microphones in array requires using more than one SAI interface.	Default enablement splits 3 mics across SAI1 and SAI2.
VOIS-663	SetAlert doesn't update existing Alert	Updating an existing Alert in companion app won't update on device.	None.
VOIS-1190	Streamer 8 second deadlock	During heavy barge-in the audio streamer can enter deadlock for 8 seconds	Set `IotTaskPoolInfo_t taskPoolInfo =.IOT_TASKPOOL_INFO_INITIALIZER_SMALL` in `rtos/amazon-freertos/libraries/c_sdk/standard/common/iot_init.c`
VOIS-1297	Missing APP_NETWORK_Init for bootloader when using ethernet	Ethernet hardware not initialized correctly	Add APP_NETWORK_Init() to bootloader before calling `APP_MQTT_Connect()`.

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: 02/2020

Document identifier: MCUXSDKMIMXRT106ARN

arm