MCUXSDKMIMXRT105XRN

MCUXpresso SDK Release Notes for EVKB-IMXRT1050

Rev. 12.0.0 — 11 July 2022 **Release notes**

Document information

Information	Content
Keywords	Arm Cortex-M, MCUXpresso SDK, IAR, KEIL, GCC, Cmake
Abstract	This document describes the MCUXpresso SDK release notes for EVKB-IMXRT1050.

1 Overview

The MCUXpresso SDK is a comprehensive software enablement package designed to simplify and accelerate application development with Arm® Cortex®-M-based devices from NXP, including its general purpose, crossover and Bluetooth™-enabled MCUs. MCUXpresso SW and Tools for DSC further extends the SDK support to current 32-bit Digital Signal Controllers. The MCUXpresso SDK includes production-grade software with integrated RTOS (optional), integrated enabling software technologies (stacks and middleware), reference software, and more.

In addition to working seamlessly with the MCUXpresso IDE, the MCUXpresso SDK also supports and provides example projects for IAR, KEIL, and GCC with Cmake. 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.

Underscoring our commitment to high quality, the MCUXpresso SDK is MISRA compliant and checked with Coverity® static analysis tools. For details on MCUXpresso SDK, see MCUXpresso-SDK: Software Development Kit for MCUXpresso.

2 MCUXpresso SDK

As part of the MCUXpresso software and tools, MCUXpresso SDK is the evolution of Kinetis SDK, includes support for LPC, DSC, and i.MX System-on-Chip (SoC). The same drivers, APIs, and middleware are still available with support for Kinetis, LPC, DSC, 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.

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.

3 Development tools

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

- Keil MDK, version is 5.37
- MCUXpresso IDE, version is 11.6.0
- GCC Arm Embedded, version is 10.3-2021.10
- IAR Embedded Workbench for Arm, version is 9.30.1

4 Supported development system

This release supports boards and devices listed in table below. The boards and devices in bold were tested in this release.

Table 1. Supported boards and devices

Development boards	MCU devices
EVKB-IMXRT1050	**MIMXRT1052DVL6B**, MIMXRT1052DVL6B, MIMXRT1052CVJ5B, MIMXRT1051DVJ6B, MIMXRT1051CVL5B, MIMXRT1052DVJ6B, MIMXRT1051DVL6B, MIMXRT1052CVL5B, MIMXRT1051CVJ5B

5 MCUXpresso SDK release package

The MCUXpresso SDK release package content is aligned with the silicon subfamily it supports. This includes the boards, devices, documentation, and middleware.

5.1 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, 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 efficiently transfers the code execution to the main() function.

5.1.1 Board support

The boards folder provides the board-specific demo applications, driver examples, and middleware examples.

5.1.2 Demo application 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.

5.2 CMSIS DSP Library

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

5.3 Operating System

5.3.1 FreeRTOS

Real-time operating system for microcontrollers from Amazon.

5.3.2 Azure RTOS

Azure RTOS middleware (FileX, GUIX, LevelX, NetX Duo, USBX) and the ThreadX RTOS are optional components in MCUXpresso SDK. Source code for these items is included in the *rtos/azure-rtos* directory, and examples for the target board are located in *boards/evkbimxrt1050/azure_rtos_examples* .

For more information, see the *MCUXpresso SDK Azure RTOS Release Notes* and *Getting Started with MCUXpresso SDK for Azure RTOS*.

5.4 Middleware

5.4.1 AWS IoT

Amazon Web Service (AWS) IoT Core SDK.

5.4.2 Azure RTOS FileX

A file system based on Azure RTOS.

5.4.3 Azure RTOS GUIX

A GUI library based on Azure RTOS.

5.4.4 Azure RTOS IoT

A software package that connects to the IoT Hub through Azure RTOS.

5.4.5 Azure RTOS LevelX

NOR/NAND Flash wear leveling component

5.4.6 Azure RTOS NetX Duo

A network protocol stack based on Azure RTOS.

5.4.7 Azure RTOS ThreadX

Azure RTOS ThreadX.

5.4.8 Azure RTOS USBX

A USB library based on Azure RTOS.

5.4.9 canopen

MicroCANopen Stack from Embedded Solutions Academy

5.4.10 cjson

Ultralightweight JSON parser in ANSI C.

5.4.11 Crank Storyboard GUI

Crank Storyboard GUI Engine from Crank Software.

5.4.12 Essential Audio Processing Library

Audio processing blocks for enhancing the tonal and spatial perception of sound in audio applications.

5.4.13 eIQ

eIQ machine learning SDK containing:

- Arm CMSIS-NN library (neural network kernels optimized for Cortex-M cores)
- Inference engines:
  - TensorFlow Lite Micro
  - DeepView^{TM} RT
- Example code for TensorFlow Lite Micro, Glow, and DeepView RT

5.4.14 Embedded Wizard GUI

The MCUXpresso SDK is pre-integrated with the Embedded Wizard GUI framework. The integration demonstrates state-of-the-art GUI applications by using hardware accelerated graphic operations (where applicable, otherwise pure software). Embedded Wizard GUI from TARA Systems.

5.4.15 emWin

The MCUXpresso SDK is pre-integrated with the SEGGER emWin GUI middleware. The AppWizard provides developers and designers with a flexible tool to create stunning user interface applications, without writing any code.

5.4.16 Fatfs

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.

5.4.17 FreeMASTER

FreeMASTER communication driver for 32-bit platforms.

5.4.18 IEC60730B Safety Library

NXP IEC60730B Safety Library

5.4.19 IoT Sensing Software Development Kit (ISSDK)

The IoT Sensing Software Development Kit (ISSDK) is the embedded software framework enabling the NXP digital and analog sensors for IoT applications. ISSDK combines a set of robust sensor drivers and algorithms along with example applications that allow users to get started with using NXP IoT motion & pressure sensors. ISSDK is being offered as a middleware component in MCUXpresso SDK.

5.4.20 JPEG library

JPEG library

5.4.21 littlefs

LittleFS filesystem stack

5.4.22 LVGL

LVGL Open Source Graphics Library

5.4.23 lwIP

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

For details, see the *lwIP TCPIP Stack and MCUXpresso SDK Integration User's Guide* (document MCUXSDKLWIPUG).

5.4.24 Maestro Audio Framework for MCU

Maestro Audio Framework library for MCU

5.4.25 Motor Control Software (ACIM, BLDC, PMSM)

Motor control examples.

5.4.26 mbedTLS

mbedtls SSL/TLS library

5.4.27 MCU Boot

MCU Bootloader source code.

5.4.28 Nghttp2 HTTP/2 C Library

C library implementation of HTTP/2

5.4.29 NXP Touch Library

NXP Touch Library

5.4.30 NXP Wi-Fi

The MCUXpresso SDK provides driver for NXP Wi-Fi external modules. The Wi-Fi driver is integrated with LWIP TCPIP stack and demonstrated with several network applications (iperf and AWS IoT).

For more information, see Getting Started with NXP based Wireless Modules and i.MX RT Platform Running on RTOS (document: UM11441).

5.4.31 Openh264

H.264 Codec Library

5.4.32 sdmmc stack

The SDMMC software is integrated with MCUXpresso SDK to support SD/MMC/SDIO standard specification. This also includes a host adapter layer for bare-metal/RTOS applications.

5.4.33 USB Host, Device, OTG Stack

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

5.4.34 USB Type-C Power Delivery Authentication

USB Type-C Power Delivery Authentication

5.4.35 USB Type-C PD Stack

See the *MCUXpresso SDK USB Type-C PD Stack User's Guide* (document MCUXSDKUSBPDUG) for more information.

Note: *The USB TYPE-C PD stack supports IAR only.*

6 Release contents

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

Table 2. MCUXpresso SDK release package contents and locations

Deliverable	Location
Boards	INSTALL_DIR/boards
Demo Applications	INSTALL_DIR/boards/<board_name>/demo_apps
Driver Examples	INSTALL_DIR/boards/<board_name>/driver_examples
Board Project Template for MCUXpresso IDE NPW	INSTALL_DIR/boards/<board_name>/project_template
Driver, SoC header files, extension header files and feature header files, utilities	INSTALL_DIR/devices/<device_name>
Peripheral drivers	INSTALL_DIR/devices/<device_name>/drivers
Toolchain linker files and startup code	INSTALL_DIR/devices/<device_name>/<toolchain_name>
Utilities such as debug console	INSTALL_DIR/devices/<device_name>/utilities
Device Project Template for MCUXpresso IDE NPW	INSTALL_DIR/devices/<device_name>/project_template
CMSIS Arm Cortex-M header files, DSP library source	INSTALL_DIR/CMSIS

Table 2. MCUXpresso SDK release package contents and locations*...continued*

Deliverable	Location
Components and board device drivers	INSTALL_DIR/components
Documents	INSTALL_DIR/docs
RTOS	INSTALL_DIR/rtos
Release Notes, Getting Started Document and other documents	INSTALL_DIR/docs
Tools such as shared cmake files	INSTALL_DIR/tools

7 MISRA compliance

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

Table 3. MISRA exception rules

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.

Table 3. MISRA exception rules*...continued*

Exception rules	Description
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.
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 identifier or 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.

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:\<folder>.

8.2 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:issdk_mag3110, issdk_host, systick, gpio_kinetis, gpio_lpc, issdk_mpl3115, sensor_fusion_agm01, sensor_fusion_agm01_lpc, issdk_mma845x, issdk_mma8491q, issdk_mma865x, issdk_mma9553, and CMSIS_RTOS2.

Also for low-level adapter components, currently the different types of the same adapter cannot be selected at the same time.

For example, if there are two types of timer adapters, gpt_adapter and pit_adapter, only one can be selected as timer adapter

in one project at a time. Duplicate implementation of the function results in an error.

Note: *Most of middleware components have complex dependencies and are not fully supported in new project wizard. Adding a middleware component may result in compile failure.*

8.3 CMSIS PACK new project compile failure

The generated configuration cannot be applied globally. The components, serial_manager_usb_cdc_virtual and serial_manager_usb_cdc_virtual_xxx (xxx means core variants like cm0plus, cm33, cm4, and cm33_nodsp) are unsupported for new project wizard of CMSIS pack and will lead to compile failure if selected while creating new project(s).

8.4 Cannot add SDK components into FreeRTOS projects

It is not possible to add any SDK components into FreeRTOS project using the MCUXpresso IDE New Project wizard.

8.5 Maestro_playback mono EAP crossover issue

When playing mono files with maestro_playback demo and using the EAP presets, when switching to crossover (“file set 9” or 10), the output signal can be distorted (missing samples).

There is a possible solution to this problem described in the following steps:

1. Increasing AUDIO_SINK_BUFFER_NUM to 3 in the audio_sink_pcmrtos.h file
2. Add following line to the streamer_pcm_setparams function in the streamer_pcm.c file: **SAI_TransferTerminateSendEDMA**(DEMO_SAI, &pcm->saiTxHandle); right before the **SAI_GetClassicI2SConfig** call.

8.6 safety_iec60730b cloned project fails to build

When you use the MCUXpresso Config Tool to clone the "safety_iec60730b" project in MCUXpresso SDK package, the created project fails to build.

The build fails because the post-build setup for CRC is incorrect. Therefore, It is recommended to use the "safety_iec60730b" project in MCUXpresso SDK package.

8.7 Some Azure examples cannot finish PHY Initializing on MCUXpresso

The following Azure examples cannot finish PHY initializing on MCUXpresso when running in Arm GCC, IAR, or Keil environment.

Examples: `azure_amqp`, `azure_http`, `azure_mqtt`, `azure_amqp_rc`, `azure_http_rc`, and `azure_mqtt_rc`.

9 Change Log

Change log of software components included in the package, see MCUXpresso SDK ChangeLog_MIMXRT1052.pdf.

10 Legal information

10.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

10.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 11 July 2022
Document identifier: MCUXSDKMIMXRT105XRN