# MCUXSDKMIMXRT116XRN
## MCUXpresso SDK Release Notes for MIMXRT1160-EVK

## Contents

## 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 5.34
- MCUXpresso IDE, version is 11.4.0
- GCC ARM Embedded, version is 10-2020-q4-major
- IAR Embedded Workbench for ARM, version is 9.10.2

### NOTE

There are some issues when using Segger J-Link Version 7.00 to debug CM4/CM7 flash related targets. NXP is working with Segger to get this issue fixed in the future. However, if you want to use the JLink probe to debug flash related targets before the issue is fixed, contact NXP for a temporary patch.

# 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 |
|---|---|
| MIMXRT1160-EVK | **MIMXRT1166DVM6A**, MIMXRT1166CVM5A, MIMXRT1166XVM5A, MIMXRT1165DVM6A, MIMXRT1165CVM5A, MIMXRT1165XVM5A |

# 5 MCUXpresso SDK release package

The MCUXpresso DSC 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 Middleware

### 5.3.1 AWS IoT

Amazon Web Service (AWS) IoT Core SDK.

### 5.3.2 Azure RTOS FileX

A file system based on Azure RTOS.

### 5.3.3 Azure RTOS GUIX

A GUI library based on Azure RTOS.

### 5.3.4  Azure RTOS IoT

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

### 5.3.5  Azure RTOS LevelX

NOR/NAND Flash wear leveling component

### 5.3.6  Azure RTOS NetX Duo

A network protocol stack based on Azure RTOS.

### 5.3.7  Azure RTOS ThreadX

Azure RTOS ThreadX.

### 5.3.8  Azure RTOS USBX

A USB Library Based on Azure RTOS.

### 5.3.9  canopen

MicroCANopen Stack from Embedded Solutions Academy

### 5.3.10  eIQ

eIQ machine learning SDK containing:

- Arm CMSIS-NN library (neural network kernels optimized for Cortex-M cores)
- inference engine for running TensorFlow Lite models
- project examples for the Glow NN Compiler

### 5.3.11  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.

### 5.3.12  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.

### 5.3.13  FreeMASTER

FreeMASTER communication driver for 32-bit platforms.

### 5.3.14  IoT Sensing Software Development Kit (ISSDK)

The IoT Sensing Software Development Kit (ISSDK) is the embedded software framework enabling NXP\u2019s digital and analog sensors for IoT applications. ISSDK combines a set of robust sensor drivers and algorithms along with example applications tallow users to get started with using NXP IoT motion & pressure sensors. ISSDK is being offered as \u201cmiddleware\u201d component in MCUXpresso SDK.

### 5.3.15  JPEG library

JPEG library

### 5.3.16   littlefs

LittleFS filesystem stack

### 5.3.17   LVGL

LVGL Open Source Graphics Library

### 5.3.18   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).

### 5.3.19   Motor Control Software (ACIM, BLDC, PMSM)

Motor control examples.

### 5.3.20   mbedTLS

mbedtls SSL/TLS library

### 5.3.21   MCU Boot

MCU Bootloader source code.

### 5.3.22   MMCAU

MMCAU

### 5.3.23   multicore

Multicore Software Development Kit

### 5.3.24   NAND Flash Management Stack

NAND Flash Management Stack.

### 5.3.25   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).

### 5.3.26   Openh264

H.264 Codec Library

### 5.3.27   openvg

Openvg library

### 5.3.28   USB Host, Device, OTG Stack

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

### 5.3.28.1 Peripheral devices tested with USB Host stack

Table 2 provides a list of USB devices tested with the USB Host stack.

Table 2. Peripheral devices

| Device type | Device |
|---|---|
| USB HUB | BELKIN F5U233BELKIN F5U304BELKIN F5U307BELKIN F4U040UNITEK Y-2151Z-TEK ZK032AHYUNDAI HY-HB608 |
| USB flash drive | ADATA C008 32 GBADATA S102 8 GADATA S102 16 GVerbatim STORE N GO USB Device 8 GKingston DataTraveler DT101 G2SanDisk Cruzer Blade 8 GBUnisplendour 1 GImation 2 GBV-mux 2 GBSanmina-SCI 128 MCorporate Express 1 GTOSHIBA THUHYBS-008G 8 GTranscend JF700 8 GNetac U903 16 GSSK SFD205 8 GBRex 4 GBSAMSUNG USB3.0 16GB |
| USB card reader/adapter | SSK TF adapterKawau Multi Card ReaderKawau TF adapterKawau SDHC cardUSB Mouse DELL MS111-PDELL M066U0ADELL MUAVDEL8TARGUS AMU76APDELL MD56U0DELL MS111-TRAPOO M110 |
| USB Keyboard | DELL SK8135DELL SK8115 |

### 5.3.29 USB Type-C Power Delivery Authentication

USB Type-C Power Delivery Authentication

### 5.3.30 VG-Lite GPU Library

vglite lib.

### 5.3.31 Voice intelligent technology library

Voice Intelligent Technology (VIT) Library.

### 5.3.32 Wireless EdgeFast Bluetooth PAL

For more information, see the MCUXpresso SDK EdgeFast Bluetooth Protocol Abstraction Layer User's Guide.

# 6 Release contents

Table 3 provides an overview of the MCUXpresso SDK release package contents and locations.

Table 3. 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 |

*Table continues on the next page...*

Table 3.  MCUXpresso SDK release package contents and locations (continued)

| Deliverable | Location |
|---|---|
| Board Project Template for MCUXpresso IDE NPW | INSTALL_DIR/boards/<board_name>/utilities |
| Driver, SoC header files, extension header files and feature header files, utilities | INSTALL_DIR/devices/<device_name> |
| CMSIS drivers | INSTALL_DIR/devices/<device_name>/cmsis_drivers |
| 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>/utilities |
| CMSIS Arm Cortex-M header files, DSP library source | INSTALL_DIR/CMSIS |
| 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 4.

Table 4.  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. |

*Table continues on the next page...*

Table 4. MISRA exception rules (continued)

| Exception rules | Description |
|---|---|
| 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. |
| 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. |

*Table continues on the next page...*

Table 4. MISRA exception rules (continued)

| Exception rules | Description |
|---|---|
| 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, CMSIS_RTOS2, and components which include cache driver, such as enet_qos.

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 svd issue

CMSIS-PACK DFP installation takes a while. When installing cmsis-pack DFP, Keil MDK processes the MCU SVD file. The large size of SVD file takes considerable time to finish this conversion. During the installation, the progress appears stalled. However, it finishes after approximately 20 minutes.

## 8.4 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).

## 8.5 MCUXpresso IDE limitation

- **Cannot debug cm4 sdram related demos with CMSIS-DAP**

  MCUXpresso IDE does not support initialization of sdram when debugging.

- **Cannot debug cm4 demos with JLINK**

  Text section in the MCUXpresso IDE projects is linked to RAM. Currently, SEGGER does not support debugging under cache enabled area.

## 8.6 IAR debug limitation

CM4 flash target demos cannot be debugged on IAR with JLINK.

## 8.7 Extra option required when using CMSIS-DAP to debug

When using CMSIS-DAP to debug CM4 sdram related target in IAR, such as flexspi_nor_sdram and sdram_txt, an extra option must be specified in the debugger settings. For details, see IAR debug limitation.

## 8.8 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.

## 8.9 Maestro recording to a file has missing audio samples

When recording audio from the on-board mic and saving the output to a file on SD card, there are missing samples in the output pcm file. The maestro_demo design causes the issue. Writing data to SD card should be buffered and handled in a separate thread. The fix for the issue is planned for the next release.

## 8.10 usb_device_mtp example cannot boot on Keil MDK µVision

After reset, the usb_device_mtp and usb_device_mtp_lite examples cannot boot successfully when using Keil MDK µVision. Adding the *--predefine="-DXIP_BOOT_HEADER_ENABLE=1"* into **Options for target > Linker > Misc controls** can fix this issue.

# 9 Change Log

Change log of software components included in the package, see the MCUXpresso SDK ChangeLog_MIMXRT1166.pdf.