MCUXSDKMIMXRT6XXRN

MCUXpresso SDK Release Notes for EVK-MIMXRT685

Rev. 1 — 05 August 2020

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 other middleware packages, such as multicore support and FatFs, and integrated RTOS support for FreeRTOS^{TM} 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 more details about MCUXpresso SDK, see the MCUXpresso SDK homepage MCUXpresso-SDK: Software Development Kit.

2 MCUXpresso SDK

As part of the MCUXpresso software and tools, MCUXpresso SDK is the evolution of Kinetis SDK, 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 the 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. It is suggested to keep the downloaded SDK archive in the root directory of your drive to avoid any unexpected build issues caused by deep path of files.

---

3 Development tools

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

- IAR Embedded Workbench for Arm version 8.50.5 with RT600 patch applied

- Makefiles support with GCC revision 9-2019-q4-major GCC9 from Arm Embedded

- MCUXpresso IDE v11.2.0

- MDK-Arm Microcontroller Development Kit (Keil)® 5.31

4 Supported development systems

This release supports 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
EVK-MIMXRT685	MIMXRT685SFFOB, **MIMXRT685SFVKB**, MIMXRT685SFAWBR, MIMXRT633SFFOB,MCIMX8QXP-CPU MIMXRT633SFVKB, MIMXRT633SFAWBR

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`
AWS IoT SDK examples	`<install_dir>/boards/<board_name>/aws_examples`
AWS Remote Control	`<install_dir>/boards/<board>/aws_examples/remote_control_android/AwsRemoteControl.apk`
CMSIS Arm Cortex®-M header files, DSP library source	`<install_dir>/CMSIS`
Cortex Microcontroller Software Interface Standard (CMSIS) driver examples	`<install_dir>/boards/<board_name>/cmsis_driver_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, SoC header files, extension header files and feature header files, utilities	`<install_dir>/devices/<device_name>`
dsp_audio_framework	`<install_dir>/middleware/dsp/audio_framework`
Driver examples	`<install_dir>/boards/<board_name>/driver_examples`
DSP examples	`<install_dir>/boards/<board_name>/dsp_examples`
FatFS examples	`<install_dir>/boards/<board_name>/fatfs_examples`
FatFS stack	`<install_dir>/middleware/fatfs`
gradle	`<install_dir>/boards/<board>/aws_examples/remote_control_android/gradle`
LittlevGL	`<install_dir>/middleware/littlevgl`
LittlevGL examples	`<install_dir>/boards/<board_name>/littlevgl_examples`
LittlevGL library	`<install_dir>/middleware/littlevgl`
lwIP Documentation	`<install_dir>/docs/lwip`
lwIP stack	`<install_dir>/middleware/lwip`

Table continues on the next page...

Table 2. Release contents (continued)

mbed TLS	`<install_dir>/middleware/mbedtls`
mbed TLS examples	`<install_dir>/boards/<board_name>/mbedtls_examples`
naturedsp	`<install_dir>/middleware/dsp/naturedsp_hifi4`
nghttp2	`<install_dir>/middleware/nghttp2`
NXP WiFi	`<install_dir>/middleware/wifi`
NXP WiFi examples	`<install_dir>/boards/<board_name>/wifi_examples`
Peripheral Drivers	`<install_dir>/devices/<device_name>/drivers`
RPMSG lite	`<install_dir>/middleware/multicore/rpmsg_lite`
RTOS Kernel Code	`<install_dir>/rtos`
RTOS examples	`<install_dir>/boards/<board_name>/rtos_examples`
SDMMC card driver	`<install_dir>/middleware/sdmmc`
Tools	`<install_dir>/tools`
Trustzone examples	`<install_dir>/boards/<board_name>/trustzone_examples`
USB demo applications	`<install_dir>/boards/<board_name>/usb_examples`
USB Documentation	`<install_dir>/docs/usb`
USB stack	`<install_dir>/middleware/usb`
Utilities such as debug console	`<install_dir>/devices/<device_name>/utilities`

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.

6.2.1.1 Peripheral devices tested with USB Host stack

Table 3 provides a list of USB devices tested with the USB Host stack.

Table 3. Peripheral devices

Device type	Device
USB HUB	BELKIN F5U233
	BELKIN F5U304
	BELKIN F5U307
	BELKIN F4U040
	UNITEK Y-2151
	Z-TEK ZK032A
	HYUNDAI HY-HB608
USB flash drive	ADATA C008 32 GB
	ADATA S102 8 G
	ADATA S102 16 G
	Verbatim STORE N GO USB Device 8 G
	Kingston DataTraveler DT101 G2
	SanDisk Cruzer Blade 8 GB
	Unisplendour 1 G
	Imation 2 GB
	V-mux 2 GB
	Sanmina-SCI 128 M
	Corporate Express 1 G
	TOSHIBA THUHYBS-008G 8 G
	Transcend JF700 8 G
	Netac U903 16 G
	SSK SFD205 8 GB
	Rex 4 GB
	SAMSUNG USB3.0 16GB
USB card reader/adapter	SSK TF adapter

Table continues on the next page...

Table 3. Peripheral devices (continued)

Device type	Device
	Kawau Multi Card Reader Kawau TF adapter Kawau SDHC card
USB Mouse	DELL MS111-P DELL M066U0A DELL MUAVDEL8 TARGUS AMU76AP DELL MD56U0 DELL MS111-T RAPOO M110
USB Keyboard	DELL SK8135 DELL SK8115

6.2.2 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.

6.2.3 NXP WiFi drivers

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

The following are the supported external processor and modules:

- NXP 88W8977 based PAN9026 Panasonic SDIO module (ENWF9201AZEF)
- NXP 88W8801 based AW-NM191SM AzureWave module (AW-NM191NF-uSD)

6.2.4 RTOS

The MCUXpresso SDK is integrated with FreeRTOS OS.

6.2.5 CMSIS

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

6.2.6 SDMMC

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.

6.2.7 Other middleware

Optional middleware packages can be included in the release based on the user selection. See `<install_dir>/SW-Content-Register.txt` for a list of components and associated licenses.

7 MISRA compliance

All MCUXpresso SDK drivers and USB stack comply to MISRA 2012 rules with the following exceptions.

Table 4. 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.
Rule 8.9	An 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.

Table continues on the next page...

Table 4. MISRA exceptions (continued)

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.
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.

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 Low speed devices not supported

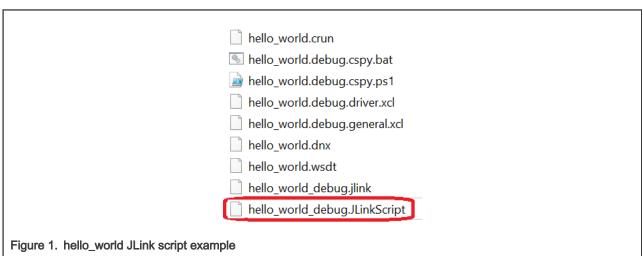The host examples cannot support low-speed devices.

8.3 IAR cannot debug RAM application with J-Link

Currently, IAR will call J-Link reset after the application is downloaded to SRAM, but such operation will cause SRAM data lost. Here's a workaround to avoid real reset, with the cost of no any reset during the debugging, and hardware status uncleared.

1. Build and debug IAR project once and see the **settings** folder created.

2. Create the *<ProjectName>_<DebugConfig>.JLinkScript* file in the **settings** folder with the following contents.

```
void ResetTarget(void) {
  JLINK_TARGET_Halt();
}
```

3. Debug the project again and now it can work.

Figure 1. hello_world JLink script example

8.4 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: `serial_manager_uart`, `serial_manager_uart_xxx`, `serial_manager_usb_cdc`, `serial_manager_usb_cdc_xxx`, `serial_manager_swo`, `serial_manager_swo_xxx`, `xip_device`.

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.

8.5 Semihosting not supported on shell-related demo

The semihosting is not supported on shell-related demos (`shell`, `dsp_xaf_demo_cm33`, and `dsp_audio_demo_bm_cm33`). The shell component is based on debug console and serial manager. When semihosting is used, debug console and serial manager are bypassed. So the shell related demos cannot work with semihosting.

8.6 CMSIS PACK new project compile failure

The generated configration 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 a new project(s).

8.7 Dsp_xaf_demo cannot handle certain audio file types

Audio files encoded using ffmpeg tool and Lavf/Lavc codec cannot be played. M4A format of AAC files is not supported either.

If the demo stops responding while trying to play any of unsupported files, please reset the board.