Freescale Semiconductor
Release Notes

Document Number: KSDK130KL82ZRN
Rev. 0, 12/2015

Kinetis SDK v.1.3.0 Release Notes for KL82Z

1 Overview

These are the release notes for the Freescale Kinetis Software Development Kit (SDK) 1.3.0 for MKL82Z.

The core of the Kinetis SDK is a set of drivers architected in two layers: the Hardware Abstraction Layer (HAL) and the Peripheral Driver Layer.

The HAL abstracts the hardware register accesses into a set of stateless functional primitives which provide the building blocks for high-level Peripheral Drivers or applications. The Peripheral Driver Layer implements use-case driven drivers by utilizing one or more HAL layer components, system services, and possibly other Peripheral Drivers.

The Kinetis SDK includes a set of example applications demonstrating the use of the Peripheral Drivers and other integrated software modules such as a Real-Time Operating System (RTOS) through an RTOS abstraction layer. The Kinetis SDK also integrates middleware such as the Freescale USB stack to provide an easy-to-use Software Development Kit for Kinetis microcontroller (MCU) product families. For the latest version of this and

Contents

other Kinetis SDK documents, see the Kinetis SDK homepage KINETIS-SDK: Software Development Kit for Kinetis MCUs.

2 What is New

These are the new features for Kinetis SDK 1.3.0:

- Added device family support:
  - MKL82

3 Development Tools

The Kinetis SDK 1.3.0 was compiled and tested with these development tools:

- Kinetis Design Studio IDE v3.0.0
- IAR Embedded Workbench for ARM® version 7.50.1
- MDK-ARM Microcontroller Development Kit (Keil)® 5.17
- Makefiles support with GCC revision 4.9-2015-q1-update from ARM Embedded
- Atollic® TrueSTUDIO® 5.4

4 Supported Development Systems

This release supports the boards and devices listed in this table. Boards and devices listed in boldface were tested in this release.

Table 1. Release Contents

Development boards	Kinetis MCU devices
TWR-KL82Z72M	**MKL82Z128VMC7**, MKL82Z128VLK7, MKL82Z128VLL7, MKL82Z128VLH7, MKL82Z128VMP7
FRDM-KL82Z	MKL82Z128VMC7, **MKL82Z128VLK7**, MKL82Z128VLL7, MKL82Z128VLH7, MKL82Z128VMP7

5 Release Contents

The table below describes the contents of this release.

Table 2. Release Contents

Deliverable	Location
Examples	<install_dir>/examples/...

Deliverable	Location
Demo applications	<install_dir>/examples/<board_name>/demo_apps/…
USB Demo applications	<install_dir>/examples/<board_name>/demo_apps/usb/…
Driver examples	<install_dir>/examples/<board_name>/driver_examples/...
Documentation	<install_dir>/doc/...
USB Documentation	<install_dir>/doc/usb/...
Projects to build libraries	<install_dir>/lib/...
Middleware	<install_dir>/middleware/...
File system	<install_dir>/middleware/filesystem/...
Driver library, startup code and utilities	<install_dir>/platform/...
Cortex Microcontroller Software Interface Standard (CMSIS) ARM Cortex®-M header files, DSP library source	<install_dir>/platform/CMSIS/…
Composite drivers for SD-card and Soundcard support	<install_dir>/platform/composite/...
Linker control files for each supported tool chain	<install_dir>/platform/devices/MKL82Z7/linker/…
SoC header files, Extension header files and feature header files	<install_dir>/platform/devices/MKL82Z7/include
CMSIS-compliant startup code	<install_dir>/platform/devices/MKL82Z7/startup/…
Peripheral Drivers	<install_dir>/platform/drivers/…
Hardware Abstraction Layer	<install_dir>/platform/hal/…
OS Abstraction for Bare Metal and RTOS	<install_dir>/platform/osa/…
System Services such as clock manager, interrupt manager, unified hardware timer, and low power manager	<install_dir>/platform/system/…
Utilities such as debug console	<install_dir>/platform/utilities/…
RTOS Kernel Code, RTOS abstraction implementations, and RTOS kernel folders	<install_dir>/rtos/...
cmake toolchain files	<install_dir>/tools
USB stack and USB projects to build libraries	<install_dir>/usb/...
Utilities such as shell	<install_dir>/utilities/...

6 Kinetis SDK Release Overview

The Kinetis SDK is intended for use with Freescale's Kinetis MCU product family based on the ARM Cortex-M0+ series architectures. The release consists of:

- Kinetis MCU platform support
- Demo applications/Driver examples
- The FatFs FAT File System
- USB Host and Device OTG stacks
- RTOS support components
- Documentation (Kinetis SDK API Reference Manual and various user's guides)

6.1 Kinetis MCU platform support

The Kinetis SDK platform directory contains the startup code, operating system abstraction, system services, driver libraries for peripherals, header files, linker files, and utilities such as the debug console implementation.

6.1.1 Startup code

The Kinetis SDK includes simple CMSIS-compliant startup code for the supported Kinetis MCUs which efficiently deliver the code execution to the main() function. An application can either include the startup code directly in the project build environment or include a prebuilt startup code library for a cleaner project build environment.

6.1.2 Operating system abstraction

The drivers are designed to work with or without an operating system through the Operating System Abstraction layer (OSA). The OSA defines a common set of services that abstract most of the OS kernel functionalities. The OSA either maps an OSA service to the target OS function, or implements the service when no OS is used (bare metal) or when the service does not exist in the target OS. The Kinetis SDK implements the OSA for Freescale MQX™ RTOS, FreeRTOS, μC/OS-II, μC/OS-III, and for OS-less "bare metal" usage. The bare metal OSA implementation is selected as the default option.

6.1.3 System services

The system services contain a set of software entities that can be used either by the Peripheral Drivers or with the HAL to build either Peripheral Drivers or an application directly. The system services include the interrupt manager, clock manager, low power manager, and the unified hardware timer interface.

6.1.4 Driver library

The Kinetis SDK provides a set of drivers for the Kinetis MCU product family on-chip peripherals. The drivers are designed and implemented around the peripheral hardware blocks rather than for a specific Kinetis MCU, and work with or without an OS through the OS Abstraction layer. The drivers are architected into two layers: the Hardware Abstraction Layer and the Peripheral Driver Layer.

The HAL is designed to abstract hardware register accesses into functional accesses. It is stateless and is intended to cover the entire hardware functionality.

The Peripheral Drivers are built on top of the HAL to provide a set of easy-to-use interfaces that handle high-level data and stateful transactions. They are designed for the most common use cases identified for the underlying hardware block and are reasonably efficient in terms of memory and performance. They are written in C language and can be easily ported from product to product as they are designed to be initialized at runtime based on the driver configuration passed in by the user. In most cases, the Peripheral Drivers can be used as is. However, if the Peripheral Driver does not address a particular target use case, it can either be modified/enhanced or completely rewritten to meet the target functionality and other requirements. In this case, the existing Peripheral Driver can be used as a reference to build a custom driver based on the HAL. For more details, see the Architectural Overview chapter in the *Kinetis SDK API Reference Manual* (document KSDK13APIRM).

Detailed implementation of hardware peripheral functionality, for both the HAL and Peripheral Driver, is implemented in stages. For example, the current version of the UART driver does not support modem control and smart card features. Likewise, the current version of the I2C driver does not support the SMBUS feature. The features which are missing from the current driver versions may be implemented in future releases.

6.1.5 Header files

The Kinetis SDK devices directory contains device-specific header files which provide direct access to the Kinetis MCU peripheral registers. Each supported Kinetis MCU device in the Kinetis SDK has an overall System-on-Chip (SoC) memory-mapped header file. In addition to the overall SoC memory-mapped header file, the Kinetis SDK includes extension header files and feature header files for each peripheral instantiated on the Kinetis MCU. Along with the SoC header files, peripheral extension header files, and feature header files the Kinetis SDK CMSIS directory includes common CMSIS header files for the ARM Cortex-M core and DSP library from the ARM CMSIS version 4.2 release.

6.1.6 Linker files

The Kinetis SDK devices directory contains linker control files (or simply linker files) for each supported tool chain and Kinetis MCU device.

6.1.7 Utilities

The utilities directory contains useful software utilities such as a debug console.

6.2 Demo applications

The demo applications demonstrate the usage of the driver libraries and other integrated software solutions on supported development systems. For details, see the *Kinetis SDK v. 1.3 Demo Applications User's Guide* (document KSDK13DEMOUG).

6.3 Driver examples

The driver examples demonstrate configuring drivers by passing configuration data to the API functions. For details, see the *Kinetis SDK v.1.3 Demo Applications User's Guide* (document KSDK13DEMOUG).

6.4 Other integrated software solutions

The Kinetis SDK is designed for easy integration with other software solutions such as OS kernels, USB stack and file systems.

6.4.1 USB stack

A Freescale USB stack is integrated with the Kinetis SDK and was tested both with and without an OS through the OS abstraction layer. For details, see the *Integration of the USB Stack and Kinetis SDK* (document USBKSDKUG).

6.4.2 File System

A FAT file system is integrated with Kinetis 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.

6.4.3 RTOS

The Kinetis SDK is pre-integrated with FreeRTOS, μC/OS-II, and μC/OS-III. OS abstraction layers are implemented for these RTOSes.

7 Known Issues

7.1 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 Kinetis 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:\Freescale folder.

7.2 No spaces in the Kinetis SDK installation

The Freescale MQX RTOS build uses batch files, which do not work when there are spaces in the file path.

7.3 USB HUB power supply

The external power supply of the USB HUB must be provided before it can be used. This is the result of the development board which is not designed to power a USB HUB and the devices connected to the HUB. Therefore, the external USB HUB that is connected to the development board should have its own power supply.

7.4 USB host cdc_serial/host_phdc/OTG example on KSDK MQX RTOS

Follow these steps to run the USB host cdc_serial/host_phdc/OTG example on the KSDK MQX RTOS.

1. Add two lines to the rtos\mqx\config\mcu\MKL82Z7\mqx_sdk_config.h

   #define BSPCFG_ENABLE_IO_SUBSYSTEM (0)

   #define printf debug_printf

2. Re-compile all libraries, such as mqx_<board_name>, mqx_stdlib_<board_name>, ksdk_mqx_lib, and usbh_sdk_<board_name>_mqx.

7.5 USB SRP feature in OTG mode

Because MKL82Z MCU does not have an internal USB voltage regulator, it cannot support the Session Request Protocol SRP (SRP) feature in the USB OTG mode. The Host Negotiation Protocol (HNP) feature is supported in the USB OTG mode because it is independent from the MCU's internal USB voltage regulator and can be implemented by software.

7.6 TPM module issue

Enable the TPM0 clock to create an interrupt for other TPM instances (TPM1 and TPM2).

7.7 P&E Micro debugger issue

Going forward, the P&E Micro OpenSDA firmware will not be shipped on any specific Freescale Freedom platform or Tower System module. A single firmware file can be used on different boards if they have the same firmware version.

On FRDM-KL82Z platforms and TWR-KL82Z72M modules, to debug with P&E Micro debugger, DEBUG-FRDM-K64F_Pemicro_v108a_for_OpenSDA_v2.0.bin should be used to update the firmware because the current KL82Z MCU's OpenSDA is v2 (instead of DEBUG_OpenSDA_for_MBED_Bootloader_by_Pemicro_v108.bin, which only works with OpenSDA v1).

7.8 Changing OSCSEL issue

The OSCSEL field in the MCG_C7 register should not be changed when using the external clock. The current MCG API allows this but can cause a potential issue, which will be fixed in the KSDK 2.0 version of the driver.

7.9 Known issue with DELL MS111-P

When the DELL MS111-P mouse is used and a keyboard and a mouse are attached to the board simultaneously through an external USB hub, the KL82 MCU's USB host_keyboard_mouse example does not recognize the keyboard and the mouse. This issue is probably the result of the DELL MS111-P power consumption. For other tested mouse devices, this example works well.

8 Updating Debuggers for MKL82Z

- P&E debug plugins can be updated on the P&E Eclipse update site:
  http://mcuoneclipse.com/2014/09/11/pe-eclipse-update-site-for-gnu-arm-eclipse-plugins/
- The J-Link installer is able to detect installed IDEs, including KDS. Tutorial for this can be found at following link:
  http://mcuoneclipse.com/2015/08/30/updating-segger-tools-in-eclipse-kinetis-design-studio/
- To ensure that the USB OTG HID mouse example works, set the jumpers as follows:

  On TWR-KL82Z72M Tower System module, set the jumper settings as follows:

  - J26: jumper on 3-4, 5-6, 7-8, and 9-10.
  - J25: jumper on 1-2, 3-5, and 6-8.
  - J22: jumper on 1-2.
  - J20: jumper on 2-3.

  On TWR-SER Tower System module, set the jumper settings as follows:

  - J10: jumper on 1-2.
  - J11: jumper on 3-4.
  - J16: jumper on 5-6.

9 Revision History

This table summarizes revisions to this document.

Revision History		
Revision number	**Date**	**Substantive changes**
0	12/2015	Initial release

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
www.freescale.com/support

Document Number: KSDK130KL82ZRN
Rev. 0
12/2015