Freescale Semiconductor
Release Notes

Document Number: KSDKKL03RN
1.0.0, 09/2014

Kinetis SDK Release Notes for the Freescale Freedom FRDM-KL03Z Platform

Contents

1 Overview

These are the release notes for the 1.0.0 version of the Freescale Freedom FRDM-KL03Z platform, based on Kinetis SDK (KSDK) 1.0.0. The core of the Kinetis SDK is a set of peripheral drivers built in two layers: the Hardware Abstraction Layer (HAL) and the Peripheral Driver layer.

The HAL abstracts the hardware register access into a set of stateless functional primitives which provide the building blocks for the high level peripheral drivers or applications. The Peripheral Driver layer implements use case driven drivers by utilizing one or more HAL layer components and sometimes other Peripheral Drivers.

The Kinetis SDK includes a set of example applications demonstrating the use of the drivers and other integrated software.

2 Development Tools

The alpha version for the Freescale Freedom FRDM-KL03Z platform was compiled and tested with these development tools:

- IAR Embedded Workbench for ARM® version 7.20.2
- Keil µVision 5.11.0
- Makefiles support with GCC revision 4.8.3
- Kinetis Design studio 1.1

This table provides a list of default debugger configurations for the FRDM-KL03Z platform.

Table 1. List of Default Debugger Configurations

IDE	FRDM
IAR	P&E Micro
Keil	P&E Micro
GCC	J-Link
KDS	J-Link

3 Supported Development Systems

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

Table 2. Supported MCU devices and development boards

Development boards	Kinetis MCU devices
FRDM-KL03Z	MKL03Z8VFG4, MKL03Z16VFG4, MKL03Z32VFG4, MKL03Z32CAF4, MKL03Z8VFK4, MKL03Z16VFK4, **MKL03Z32VFK4**

4 Release Contents

This table describes the release contents.

Table 3. Release Contents

Deliverable	Location
Example applications	<install_dir>/demos/...
Specific content for the evaluation boards	<install_dir>/boards/...
Documentation	<install_dir>/doc/...
Pre-built libraries and projects to build libraries	<install_dir>/lib/...
Driver library, startup code and utilities	<install_dir>/platform/...
Cortex Microcontroller Software Interface Standard (CMSIS) ARM Cortex®-M header files, DSP library source, and IP extension header files	<install_dir>/platform/CMSIS/…
Peripheral Drivers	<install_dir>/platform/drivers/…
Hardware Abstraction Layer	<install_dir>/platform/hal/…
CMSIS-compliant Startup Code	<install_dir>/platform/startup/…
Utilities such as debug console and Bare Metal OS Abstraction	<install_dir>/platform/utilities/…
Linker control files for each supported toolchain	<install_dir>/platform/linker/…

5 Kinetis SDK Release Overview

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

- Kinetis MCU platform support
- Board configuration support
- Demo applications
- The FatFs FAT File System
- Documentation (Kinetis SDK reference manual and various user's guides)

5.1 Kinetis platform support

The platform directory contains the startup code, driver libraries for peripherals, OS abstraction implementation for bare metal cases, and utilities such as the debug console.

5.1.1 Startup code

The Kinetis SDK includes a set of simple CMSIS-compliant startup code which efficiently deliver the code execution to the `main()` function. An application can either include the startup code directly in the workspace, or include a prebuilt startup code library for a cleaner project space.

5.1.2 Operating system abstraction

The drivers are designed to work with or without an operating system through the OSA. The OSA defines a common set of services that abstract most of the OS kernel functionality. 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 OS-less "bare metal" usage. The bare metal OS abstraction implementation is selected as the default option.

5.1.3 System Services

The system services contain a set of software entities that can be used either by the Peripheral Drivers or with 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.

5.1.4 Driver library

The Kinetis SDK provides a set of drivers for the peripherals found on Kinetis product families. The drivers are designed and implemented around the peripheral hardware blocks rather than for a specific

Kinetis SoC, and work with or without an OS through the OS Abstraction layer. The drivers are built in two layers: the Hardware Abstraction Layer (HAL) and the Peripheral Driver layer (PD).

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

The PD is built on top of HAL to provide a set of easy-to-use interfaces that handle high level data and stateful transactions.

The PD is designed for the most common use cases identified for the underlying hardware block. The drivers are written in C language and are efficient in terms of memory and performance. The drivers are also designed to initialize at runtime based on the driver configuration, so the drivers can be easily ported from product to product and can be used in ROM, with minimum effort when necessary. In most cases, the drivers can be used as is. If the PD does not address the target use cases, it can either be modified, enhanced, or completely rewritten to meet the target functionality and other requirements. The existing peripheral drivers can be used as references to build custom drivers based on the HAL.

Detailed implementation of IP functionality, for both HAL and peripheral drivers, 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 will be implemented in future releases.

5.1.5 Header files

The Kinetis SDK CMSIS directory contains CMSIS-compliant 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 for each peripheral instantiated on the Kinetis MCU. Along with the SoC header files and peripheral extension header files, the Kinetis SDK also includes common CMSIS header files for the ARM Cortex-M core and DSP library from the ARM CMSIS version 4.0 release.

5.1.6 Linker files

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

5.1.7 Utilities

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

5.2 Board configuration

The board directory in the Kinetis SDK is mainly used for the board-specific configuration and pin muxing. The board directory also contains software components specific to the boards, such as Accelerometer and SPI Flash implementations.

5.3 Demo applications

The example applications demonstrate the usage of the driver libraries and other integrated software solutions on supported evaluation boards. For details, see the *Kinetis SDK Demo Applications User's Guide* (document KSDKDKL03DEMOUG).

6 Known Issues

6.1 Maximum file path length in Windows® 7

Windows® 7 imposes a 260 maximum character length for file paths. When installing Kinetis SDK, place it in a directory close to the root to prevent file paths exceeding the maximum character length specified by Windows. The recommended location is the C:\Freescale folder.

6.2 Default KDS toolchain (GCC 4.8.0) RAM Overhead

The default KDS toolchain (GCC 4.8.0) RAM has two overhead issues.

1. The built out applications binary image requires 700 bytes and 1KB RAM more than the GCC 4.8.3 official toolchain and IAR.
2. The default C runtime library requires more than 400 bytes HEAP reserved to set up the environment. This is not acceptable, as HEAP is not mandatory.

With these issues there is no RAM space left for the application's STACK and HEAP. All the demo applications cannot run on the KL03, because it only has 2K RAM.

6.3 Installer Issue

Note that the Linux installer was tested only on a host with Ubuntu 12.0.4.

When uninstalling the Kinetis SDK, the system variable KSDK_PATH will remain set in the Windows Registry until the next PC reboot. If you attempt to install the Kinetis SDK before rebooting the PC, the installer will think the previous instance is still be valid, and may not set the KSDK_PATH variable correctly. Reboot your PC after uninstalling the Kinetis SDK to avoid this problem.

7 Revision History

This table summarizes revisions to this document.

Revision History		
Revision number	**Date**	**Substantial changes**
1.0.0	9/2014	Initial release

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
www.freescale.com/support

Document Number: KSDKKL03RN
Rev. 1.0.0
09/2014