Freescale Semiconductor, Inc.
Release Notes

Document Number: KSDK110KL16RN
Rev. 0, 03/2015

Kinetis SDK v1.1.0 Release Notes for KL16, KL26, and KW01 Derivatives

1 Overview

These are the release notes for the Freescale Kinetis Software Development Kit (KSDK) supporting KL16, KL26, and KW01 derivatives based on the Kinetis SDK version 1.1.0. 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

Contents

easy-to-use Software Development Kit for Kinetis microcontroller (MCU) product families.

2 What Is New

These are the new features for Kinetis SDK 1.1.0:

- Added device family support:
  - MKL16Z4
  - MKL26Z4
  - MKW01Z4

3 Development Tools

The Kinetis SDK 1.1.0 was compiled and tested with these development tools:

- Kinetis Design Studio IDE v2.0
- IAR Embedded Workbench for ARM® version 7.30
- MDK-ARM Microcontroller Development Kit (Keil) ® 5.12
- Makefiles support with GCC revision 4.8.3 from ARM® Embedded
- Atollic® TrueSTUDIO® 5.2

4 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-KL26Z	MKL16Z32VFM4,MKL16Z64VFM4,MKL16Z128VFM4, MKL16Z32VFT4,MKL16Z64VFT4,MKL16Z128VFT4, MKL16Z32VLH4,MKL16Z64VLH4,MKL16Z128VLH4, MKL26Z32VFM4,MKL26Z64VFM4,MKL26Z128VFM4, MKL26Z128CAL4,MKL26Z32VFT4,MKL26Z64VFT4, MKL26Z128VFT4,MKL26Z32VLH4,MKL26Z64VLH4, **MKL26Z128VLH4**
MRB-KW01 (see Note below)	**MKW01Z128CHN**

Note that all regional versions of the MRB-KW01 board (MRB-KW019032NA, MRB-KW019032JA, and MRB-KW019032EU) share a common board support directory mrbkw01. The difference in the value of the crystal oscillator is supported by available startup clock configurations (see /platform/startup/MKW01Z4/system_MKW01Z4.h file). There are two options to select the appropriate clock configuration in the startup:

1. Define one of these symbols with *#define* directive in the beginning of the system_MKW01Z4.h file: MRB_KW01_9032EU, MRB_KW01_9032NA or MRB_KW01_9030JA according to your board version. Note that it is also possible to define the symbol using compiler options.
2. Modify the CLOCK_SETUP macro in the system_MKW01Z4.h file. The CLOCK_SETUP 4 is pre-configured for the MRB-KW019032NA and MRB-KW019032EU boards. The CLOCK_SETUP 5 is pre-configured for the MRB-KW019032JA board.

5 Release Contents

This table describes the release contents.

Table 3. Release Contents

Deliverable	Location
Specific content for the evaluation boards	<install_dir>/boards/...
Demo applications	<install_dir>/demos/...
Documentation	<install_dir>/doc/...
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/…
Linker control files for each supported tool chain	<install_dir>/platform/linker/…
OS Abstraction for Bare Metal and RTOS	<install_dir>/platform/osa/…
CMSIS-compliant Startup Code	<install_dir>/platform/startup/…
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/...
A Processor Expert service pack and MQX RTOS task-aware plugins for tool chains.	<install_dir>/tools
USB stack and demos	<install_dir>/usb/...
Utilities such as shell and standard libraries	<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-M series architectures. The release consists of:

- Kinetis MCU platform support
- Board configuration support
- Demo applications
- USB Host and Device stacks
- RTOS support components
- Documentation (Kinetis SDK reference manual and various user’s guides)
- The FatFs FAT File System

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*.

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 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.

6.1.6 Linker files

The Kinetis SDK 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 Board configuration

The board directory in the Kinetis SDK is mainly used for the board-specific configuration and pin muxing.

6.3 Demo applications

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

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 system.

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*.

6.4.2 RTOS

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

6.4.3 File System

A FAT file system is integrated with the Kinetis SDK and can be used to access the USB memory stick when using the USB mass storage device class implementation.

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 due to the fact that the development board 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 Installer issue

Note that the Linux® operating system installer was tested only on a host with Ubuntu 14.04.

7.5 C linkage declaration

The KSDK is fully tested with all supported C compilers. However, the C linkage declaration for C++ compilers may not be done correctly. This will be fixed in the next KSDK release.

7.6 J-Link GDB server issue

Some SEGGER J-Link GDB server versions fail to program the MKW01 derivative (MRB-KW01 board). To prevent this issue, ensure that that the SEGGER J-Link GDB server V4.94i and higher is installed from the appropriate link: www.segger.com/jlink-software.html.

When using the Atollic TrueSTUDIO, the path to the newly installed GDB J-Link server must be manually updated in the main menu by going to Window->Preferences->Run/Debug->Embedded C/C++ Application->Debug Hardware->SEGGER J-Link.

7.7 P&E Multilink issue

The P&E GDB server installed with Atollic TrueSTUDIO may fail to program the MKW01 derivative (MRB-KW01 board). To prevent this issue, ensure that the latest version the P&E GDB server is installed from the appropriate link: www.pemicro.com.

Additionally, verify that the path to the newly installed P&E GDB server is correct in the main menu by going to Window->Preferences->Run/Debug->Embedded C/C++ Application->Debug Hardware->P&E Micro.

7.8 KDS IDE and Linux OS issue

When using KDS IDE with Linux OS, the path to the newly installed GDB J-Link server must be manually updated in the main menu by going to Window->Preferences->Run/Debug-> String Substitution. Edit the jlink_path and change the Value to the J-link installation path.

7.9 UART driver/HAL support

Three UART modules are available for the MKL16, MKL26, and MKW01 derivatives. However, these modules are supported by different HAL/drivers:

- UART0 module is supported by LPSCI HAL/driver
- UART and UART2 modules are supported by UART HAL/driver

For more information about HAL/drivers, see *Kinetis SDK v.1.1 API Reference Manual* (document KSDK11APIRM).

8 Revision History

This table summarizes revisions to this document.

Revision History

Revision number	Date	Substantial changes
0	03/2015	Initial release

Document Number: KSDK110KL16RN
Rev. 0
03/2015