Freescale Semiconductor
Release Notes

Document Number: KSDK120KM34ZRN
Rev.0, 06/2015

Kinetis SDK v1.2.0 Release Notes for the TWR-KM34Z75M Tower System Module

1 Overview

These are the release notes for standalone release supporting the TWR-KM34Z75M Freescale Tower System module, based on Kinetis SDK (KSDK) 1.2.0. 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.

Contents

2 Development Tools

This release was compiled and tested with these development tools:

- IAR Embedded Workbench for ARM® version 7.40.2
- MDK-ARM Microcontroller Development Kit (Keil) 5.15
- Makefiles support with GCC revision 4.8.2014-q3-update from ARM Embedded
- Kinetis Design Studio IDE (KDS) 3.0.0
- Atollic® TrueSTUDIO® 5.3.0

This table provides a list of default debugger configurations for the TWR-KM34Z75M Freescale Tower System module.

Table 1. List of Default Debugger Configurations

IDE	Debugger
IAR Embedded Workbench for ARM	P&E Micro
MDK-ARM Microcontroller Development Kit (Keil)	P&E Micro
Kinetis Design Studio IDE	P&E Micro
Atollic TrueSTUDIO for ARM	P&E Micro
ARM GCC	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
TWR-KM34Z75M	**MKM34Z256VLQ7**, MKM34Z256VLL7

4 Release Contents

This table describes the release contents.

Table 3. Release Contents

Deliverable	Location
Specific content for the evaluation boards	<install_dir>/examples/twrkm34z75m/...
Demo applications	<install_dir>/examples/twrkm34z75m/demo_apps/...
Example applications	<install_dir>/examples/twrkm34z75m/driver_examples/...
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	<install_dir>/platform/CMSIS/…
IP extension header files	<install_dir>/platform/devices/MKM34Z7/include <install_dir>/platform/devices/MKM34Z7/include/…
Linker control files for each supported tool chain	<install_dir>/platform/devices/MKM34Z7/linker/… <install_dir>/platform/devices/MKM34Z7/linker/…
CMSIS-compliant Startup Code	<install_dir>/platform/devices/MKM34Z7/startup/… <install_dir>/platform/devices/MKM34Z7/startup/…
Peripheral Drivers	<install_dir>/platform/drivers/…
Hardware Abstraction Layer	<install_dir>/platform/hal/…
OS Abstraction for Bare Metal	<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/…
A Processor Expert service pack and cmake patch for tool chains.	<install_dir>/tools

5 Debugging with P&E Micro OpenSDA debugger

The TWR-KM34Z75M board is delivered with OpenSDA firmware version V116. Install the latest OpenSDA firmware V117 to evaluate software projects for the MKM34Z256VLQ7 on the TWR-KM34Z75M Tower System module. Download the latest OpenSDA firmware from P&E Micro website: www.pemicro.com/OpenSDA/.

To update the OpenSDA with the latest application follow these instructions:

1. Start the board in the Bootloader mode by holding the SW2 RESET button while plugging the board into the USB port.
2. Drag and drop the MSD-DEBUG-TWR-KL43Z_Pemicro_v117.SDA application file into the Bootloader mass storage device.
3. Unplug and re-plug the TWR-KM34Z75M Tower System module from the USB port after the .SDA file is copied.
4. To confirm the update was successful, open the SDA_INFO.HTM file in enumerated TWR-KM34Z75M mass storage device and check the application version number.

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/Example applications
- Documentation

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 space or include a prebuilt startup code library for a cleaner project space.

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 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 instances, 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 instance, 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 KSDK12KM34ZAPIRM).

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.

NOTE

Because of the limited size of RAM in this device, the available RAM is also limited. When using the default linker files for applications, especially the RAM linker file, the users should be aware of the text size and data sections.

6.1.7 Utilities

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

6.1.8 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 implementations.

6.1.9 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 v.1.2 Demo Applications User's Guide* (document KSDK12KM34ZDEMOUG).

7 Known Issues

7.1 Maximum file path length in Windows® 7 operating system

Windows 7 OS 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 OS. The recommended location is the C:\Freescale folder.

7.2 Smartcard issue

Currently, the Smartcard driver is targeted to support DMA transfer mode. To use the CPU transfer mode, ensure that the DMA is initialized first in the application code.

8 Revision History

This table summarizes revisions to this document.

Revision History		
Revision Number	**Date**	**Substantive changes**
0	06/2015	Initial release

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
www.freescale.com/support

Document Number: KSDK120KM34ZRN
Rev.0
06/2015