



FTF 2016
TECHNOLOGY FORUM

KINETIS SDK 2.0

FTF-DES-N1960

ERIC OCASIO, GREG HEMSTREET, CLARK JARVIS

FTF-DES-N1960

MAY 18, 2016

PUBLIC USE



AGENDA

- Kinetis SDK v2 Overview
- Kinetis Expert Tools Overview
- Kinetis SDK v2 Folder Structure
- How to Create a New Kinetis SDK Project: Lab
- Kinetis SDK Configuration
- Porting to Custom Hardware
- Kinetis SDK Drivers: Lab
- Kinetis SDK v1 to v2 Transition
- Question & Answer

KINETIS SDK V2 OVERVIEW



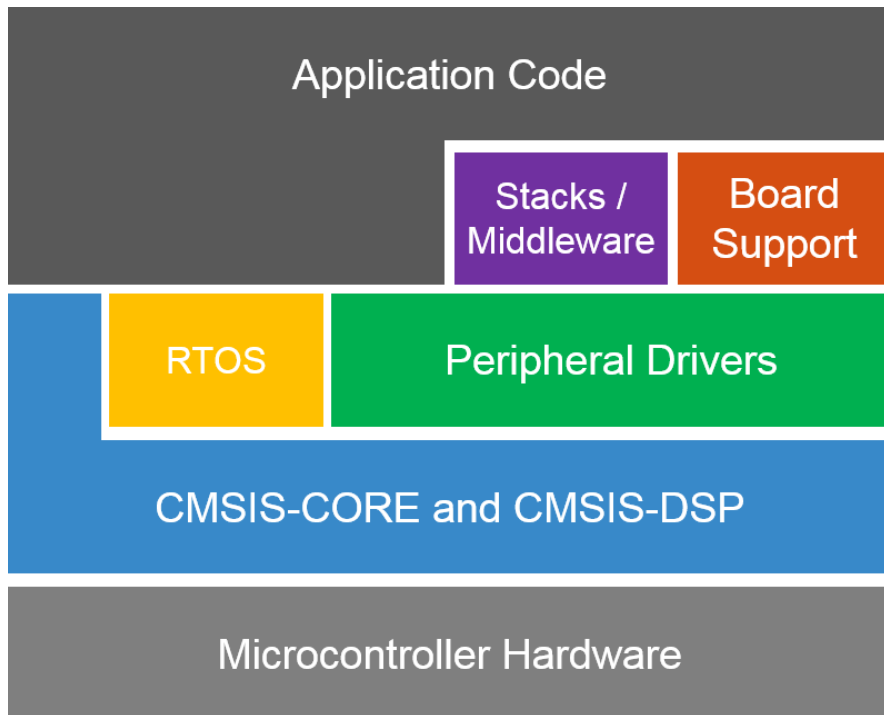
What Is an SDK for and Why It's Needed?

- In general, an SDK is a package of pre-written code that developers can re-use in order to minimize the amount of unique code that they need to develop themselves
- It can help to prevent unnecessary duplication of effort in a development team or community
- It has a common application programming interface (API) for different platforms or peripherals, what shortens the application developing time
- Thanks to use of abstraction layers it's more intuitive and concise for programmers

Kinetis Software Development Kit v2.x (KSDK v2)



The software framework and reference for Kinetis MCU application development



Architecture

- CMSIS-CORE compatible
- Single driver for each peripheral
- Transactional APIs w/ optional DMA support for communication peripherals

Integrated RTOS

- FreeRTOS, Micrium uC/OS-II & -III
- RTOS-native driver wrappers

Integrated Stacks & Middleware

- USB Host, Device and OTG
- lwIP, FatFS
- Crypto acceleration plus wolfSSL & mbedTLS
- SD and eMMC card support

Reference Software

- Peripheral Driver usage examples
- Application Demos
- FreeRTOS usage demos

License

- BSD 3-clause for startup, drivers, USB stack

Toolchains:

- KDS, IAR, Keil, Atollic, GCC w/ CMake

Quality

- Production grade software
- MISRA 2004 compliance
- Checked with Coverity Static Analysis tools

Kinetis SDK v2 – 2016 Release Schedule



Release 1 – Jan 27th

Kinetis K & L “Hot Products”

- FRDM-K22F, TWR-K22F120M
- TWR-K21F120MA
- FRDM-K64F and TWR-K64F120M
- TWR-K65F180M
- TWR-K81F150M and FRDM-K82F
- FRDM-KL43Z and TWR-KL43Z48M
- FRDM-KL27Z

Release 2 – June

Remaining K & L devices, Kinetis M

- FRDM-KL02Z
- FRDM-KL03Z
- FRDM-KL25Z
- FRDM-KL26Z
- FRDM-KL46Z
- TWR-K60D100M
- TWR-K21D50M
- TWR-K24F120M
- TWR-KL82Z72M and FRDM-KL82Z
- TWR-KM34Z75M

Release 3 – June

Kinetis W devices

Release 4 – August

Kinetis V devices

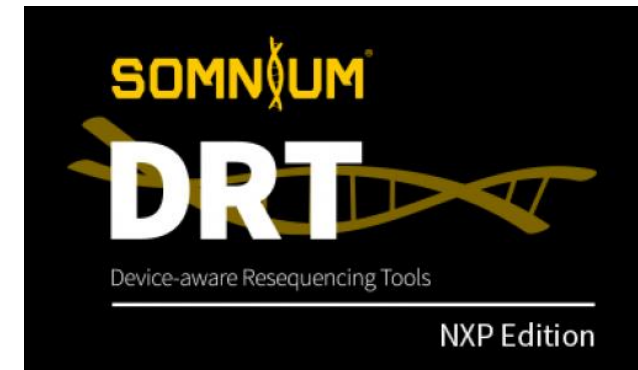
Learn more at: www.nxp.com/KSDK



Available via *Kinetis Expert – SDK Builder*, kex.nxp.com

Kinetis SDK v2 – Toolchain Support

Learn more at: www.nxp.com/KSDK



(Kinetis Design Studio project importer)

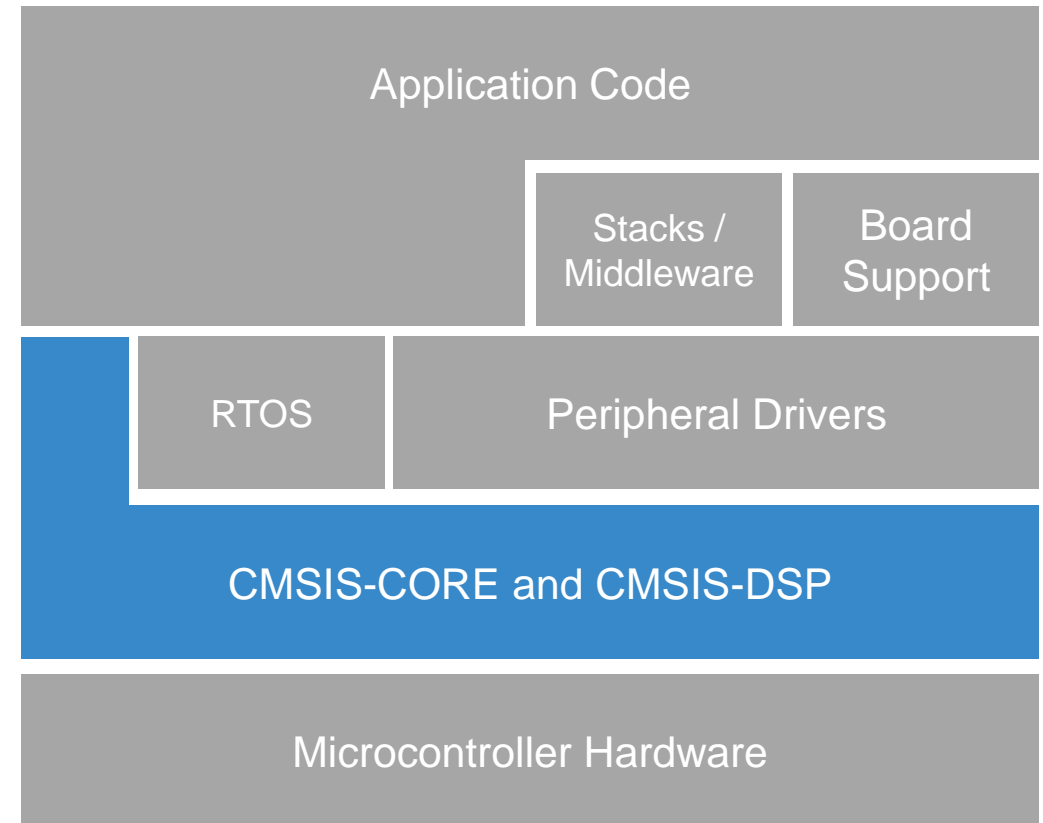
Kinetis SDK v2 – CMSIS Device Support

CMSIS-CORE provides a standard for a basic run-time system and user access to the core and the peripherals:

- Hardware Abstraction Layer (HAL) - definitions for the SysTick, NVIC, FPU registers, and core access functions
- Standardized MCU header file format – common register/bit access methods, system exception and interrupt naming
- Standard methods for system initialization – for example, the [SystemInit\(\)](#) function for essential system configuring
- Intrinsic functions used to generate CPU instructions that are not supported by standard C functions.

CMSIS-DSP is a suite of common signal processing functions including math, filters, matrix, transforms, motor control, statistical, and interpolation functions.

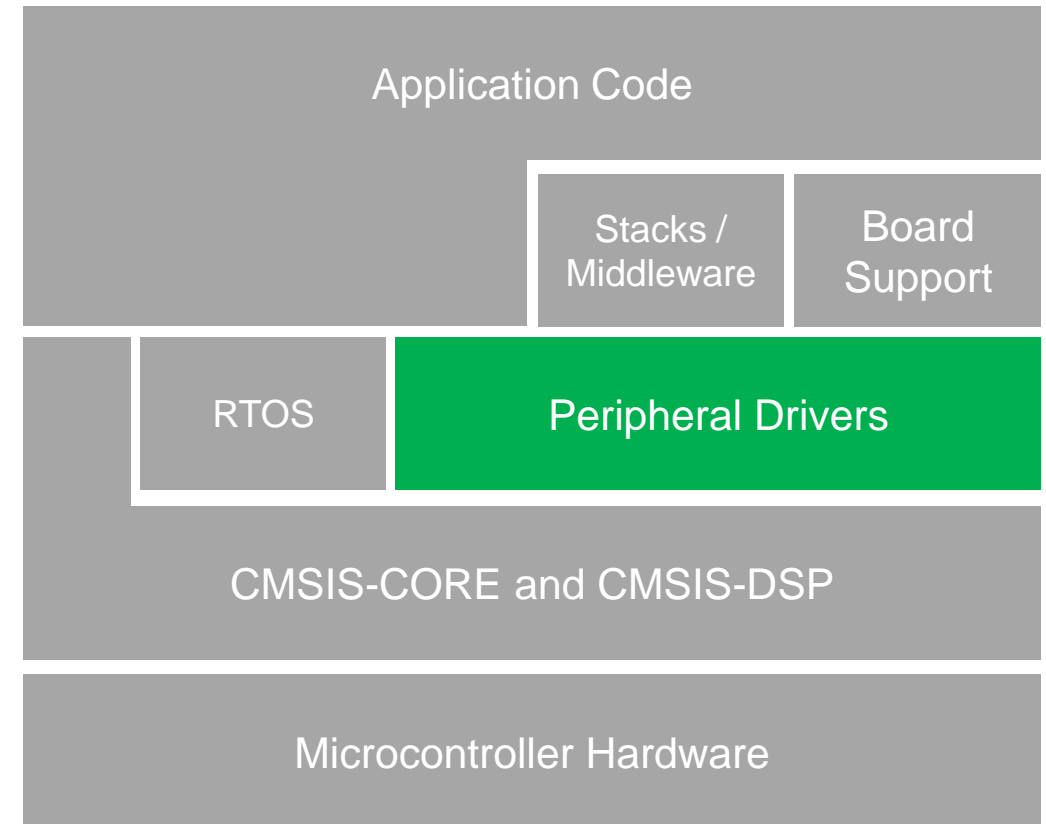
Note, KSDK does not provide a **CMSIS-Driver** compatible layer—this is under consideration for a future release.



Kinetis SDK v2 – Peripheral Drivers

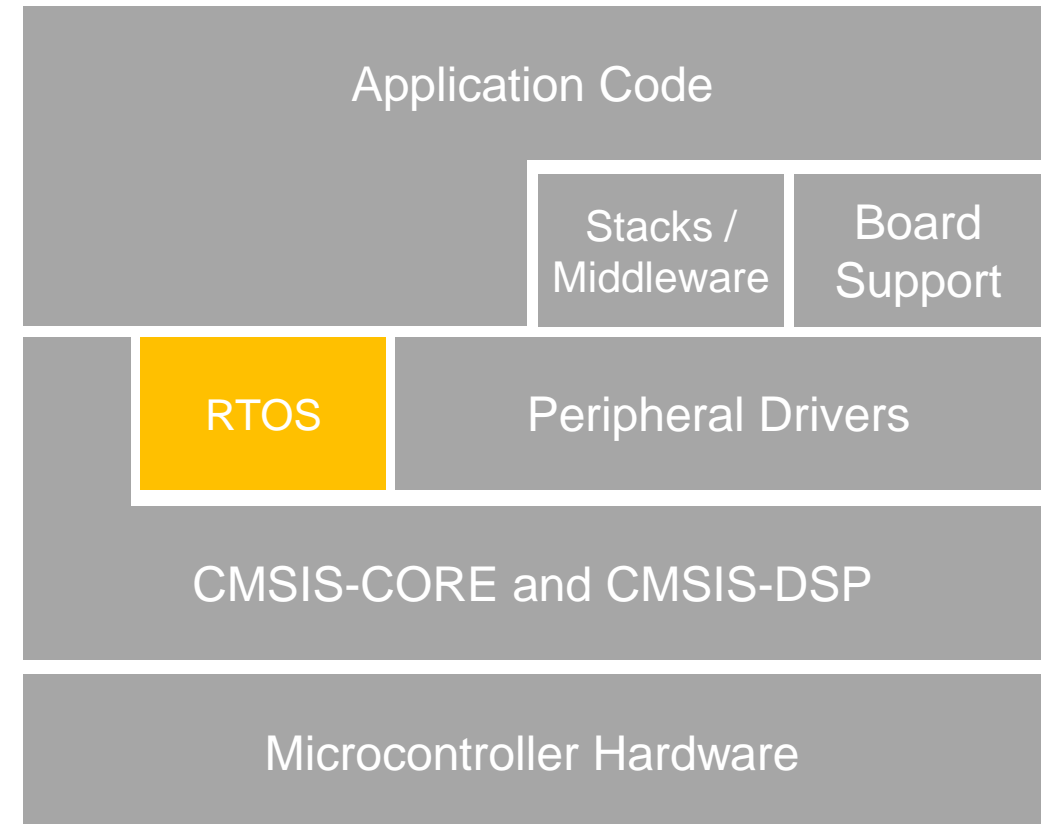
Kinetis SDK Drivers:

- Single driver for each peripheral
- Full peripheral coverage for each MCU
- All drivers include low-level functional APIs
- Communication peripheral drivers feature transactional APIs
 - Non-blocking, interrupt based
- Communication peripheral drivers also have optimized RTOS wrapper drivers
 - Uses native RTOS APIs – no operating system abstraction



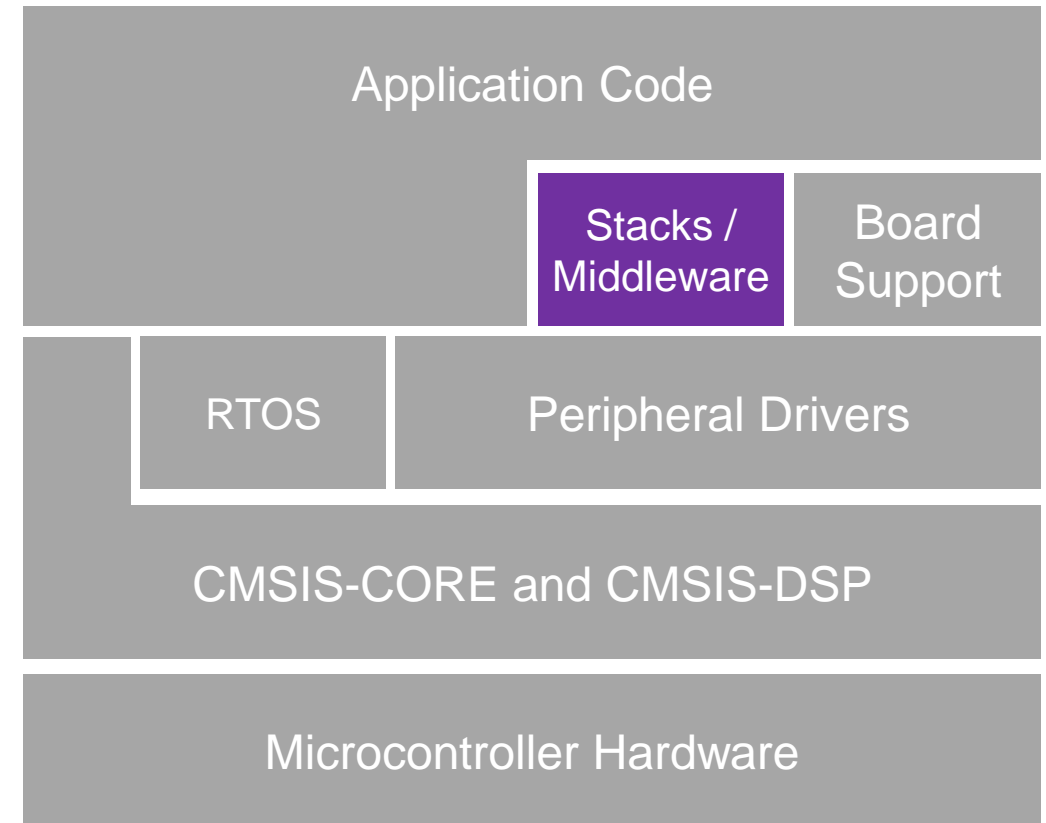
Kinetis SDK v2 – RTOS Options

- FreeRTOS, uC/OS-II, uC/OS-III kernels pre-integrated
- Focus on FreeRTOS with
 - Demonstration applications
 - RTOS Usage examples
- RTOS examples include:
 - freertos_dsipi
 - freertos_event
 - freertos_generic
 - freertos_hello
 - freertos_i2c
 - freertos_mutex
 - freertos_queue
 - freertos_sem
 - freertos_swtimer
 - freertos_tickless
 - freertos_uart
 - ucossiii_hello
 - ucossii_hello

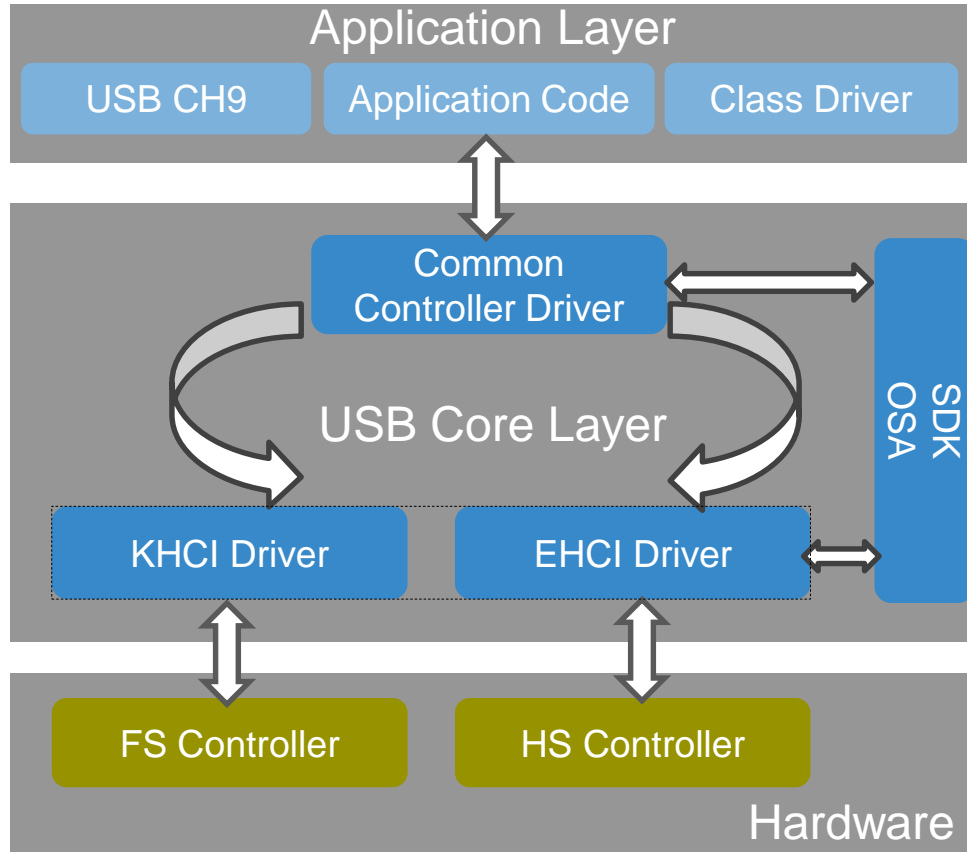


Kinetis SDK v2 – USB Stack

- **KSDK USB Stack** is a comprehensive, open-source device and host stack. It supports baremetal and RTOS application, multiple class implementations, several demo applications.
- **70+ demo applications** that support extensive features including:
 - 7 device classes with 3 **composite** examples
 - 5 host classes with **USB hub** support
 - Full-speed and high-speed under USB 2.0 specs
- **High quality stack, ready for production use**
 - **USB-IF certification** on both FS and HS
 - Optimized for code size - down to **6K flash** and 2K RAM – and performance among competition
 - Device demos have “**Lite**” versions that are even smaller in code size



USB Demo Examples

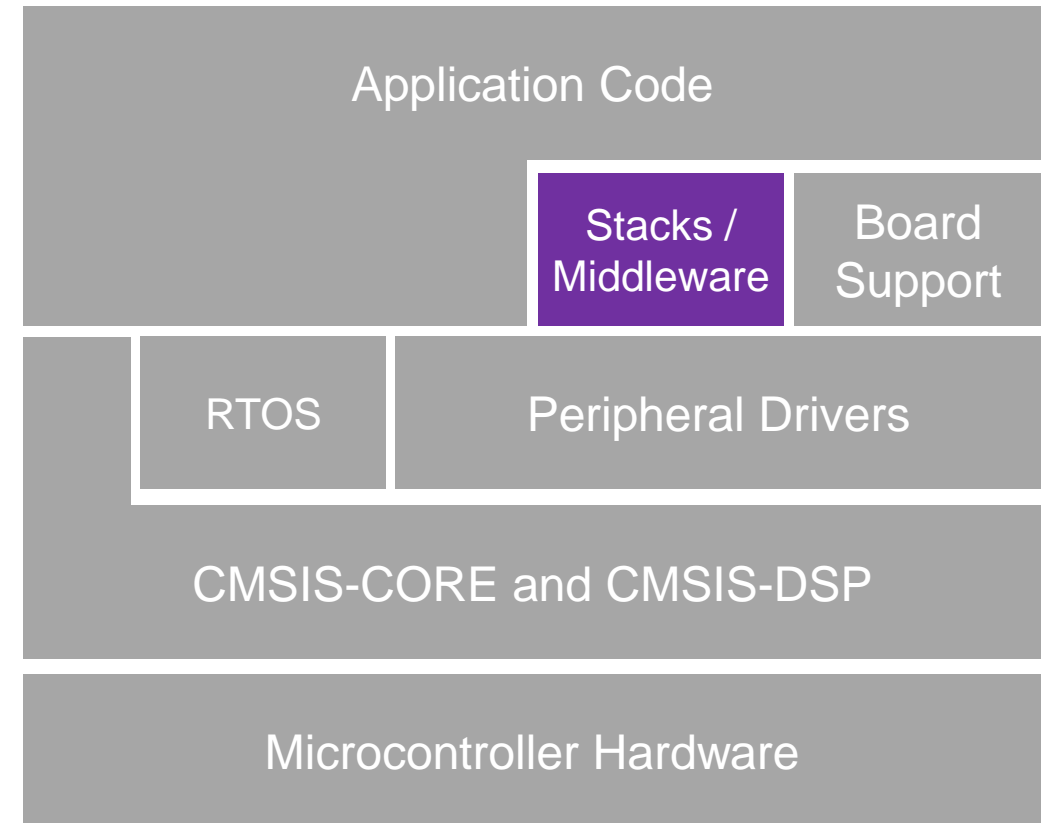


Note: OTG demos will be enabled in by KSDK 2.0 Release 4

	Class	Demos	"Lite" Apps	RTOS
Host Class	HID	Mouse, Mouse+Keyboard	No	FreeRTOS, uC/OS-II & III
	CDC	COM port	No	FreeRTOS
	MSC	UFI/SCSI U-disk, FATFS	No	FreeRTOS
	Audio	Speaker	No	FreeRTOS
	PHDC	Weight scale manager	No	FreeRTOS
Device Class	HID	Generic, Mouse	Yes	FreeRTOS, uC/OS-II & III
	CDC	Virtual com	Yes	FreeRTOS
	MSC	RAMdisk	Yes	FreeRTOS
	Audio	Generator, Speaker	Yes	FreeRTOS
	Video	Virtual camera, FlexIO	Yes	FreeRTOS
	PHDC	Weight scale	Yes	FreeRTOS
	Composite	HID Mouse+Keyboard, HID+Audio, CDC+MSC	Yes	FreeRTOS

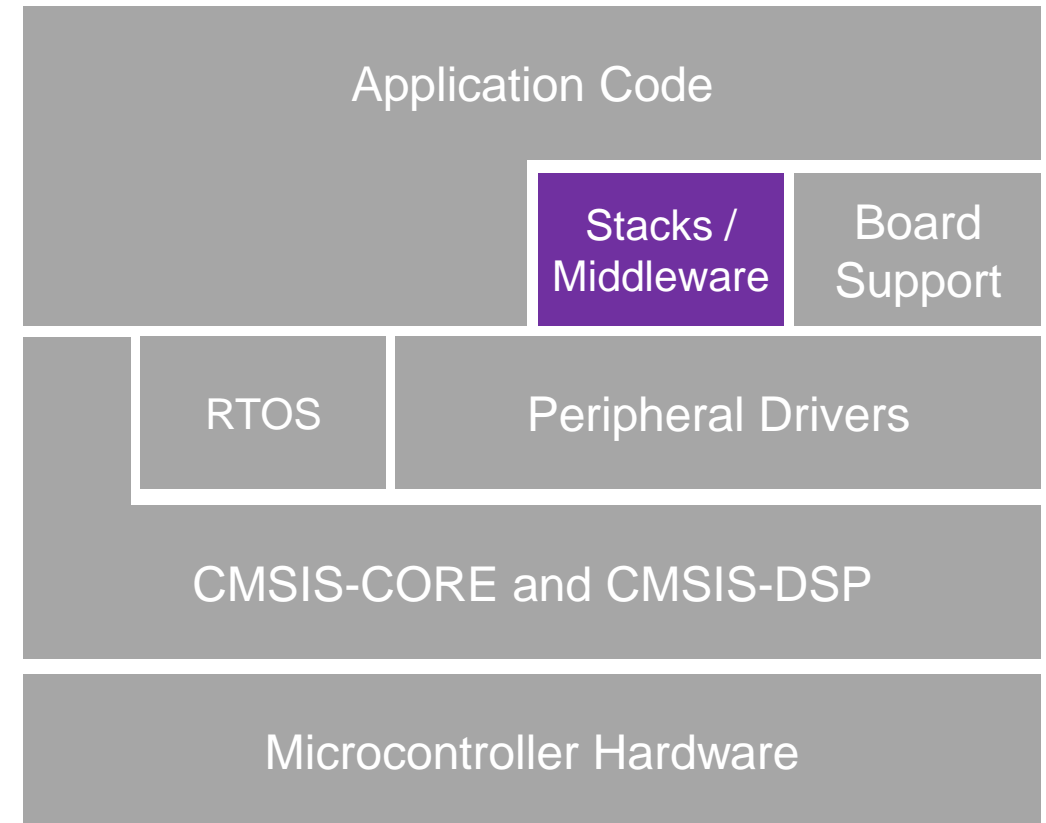
Kinetis SDK v2 – FatFS

- FatFS is a generic FAT file system module for small embedded systems. Features include:
- ANSI C compliant and completely separated from disk I/O layer
- Windows compatible FAT file system
- Very small footprint
- Various configuration options
 - Multiple volumes (physical drives and partitions)
 - Long file name support in ANSI/OEM or Unicode
 - RTOS support
 - FAT sub-types: FAT12, FAT16 and FAT32.
- Available demos:
- `sdcard_fatfs`
- `usb_host_msd_fatfs` (baremetal & FreeRTOS)



Kinetis SDK v2 – lwIP

- lwIP – the lightweight Internet Protocol
- A full scale TCP/IP stack for embedded systems
- lwIP supports the following protocols:
 - ARP
 - DHCP
 - IPv4 and v6
 - ICMP
 - TCP
 - IGMP
 - UDP
 - PPP
 - DNS
 - PPPoE
 - SNMP
- Example applications include:
 - lwip_httpsrv
 - lwip_ping
 - lwip_tcpecho
 - lwip_udpecho



Kinetis SDK v2 – Documentation Updates

- New Kinetis SDK Getting Started Guide
 - Updated for new project structure and file layout
- New Kinetis SDK 2.0 API Reference Manual
 - Also available online at <http://kex.freescale.com/apidoc/2.0/>
- Kinetis SDK 2.0 Transition Document
 - Details on changes between KSDK 1.3 and KSDK 2.0, as well as migration to FreeRTOS
- Getting Started websites and videos also being updated for Kinetis SDK 2.0 instructions

KINETIS EXPERT SDK BUILDER



Kinetis Expert Tool

- KSDK 2.0 is distributed via Kinetis Expert Tool (KEX)
 - Offers ability to customize download options by toolchain, RTOS, and specific device
 - Configurations stored in web interface for downloads
 - Downloads automatically generated
- A monolithic download with all supported boards/devices is no longer offered
- Direct links to a pre-configured download for specific board will be available
- KSDK 1.x still available online at nxp.com/ksdk for legacy support

Kinetis Expert Tool

Start with “Build an SDK”

NXP Kinetis Expert

Kinetis Expert System Configuration Tool

Kinetis Expert provides a set of system configuration tools that help users of all levels with a Kinetis-based MCU solution. It is an expert on all things Kinetis – let it be your guide from first evaluation to production development.

Get started now with the Kinetis Expert online preview.

Build an SDK

Choose a configuration for a specific evaluation board or a specific MCU, then let Kinetis Expert help you build a custom version of the [Kinetis SDK](#) and locate other software and tools.

Power Estimation

Estimate and optimize your system's power consumption quickly with this simple graphical interface.

What's new

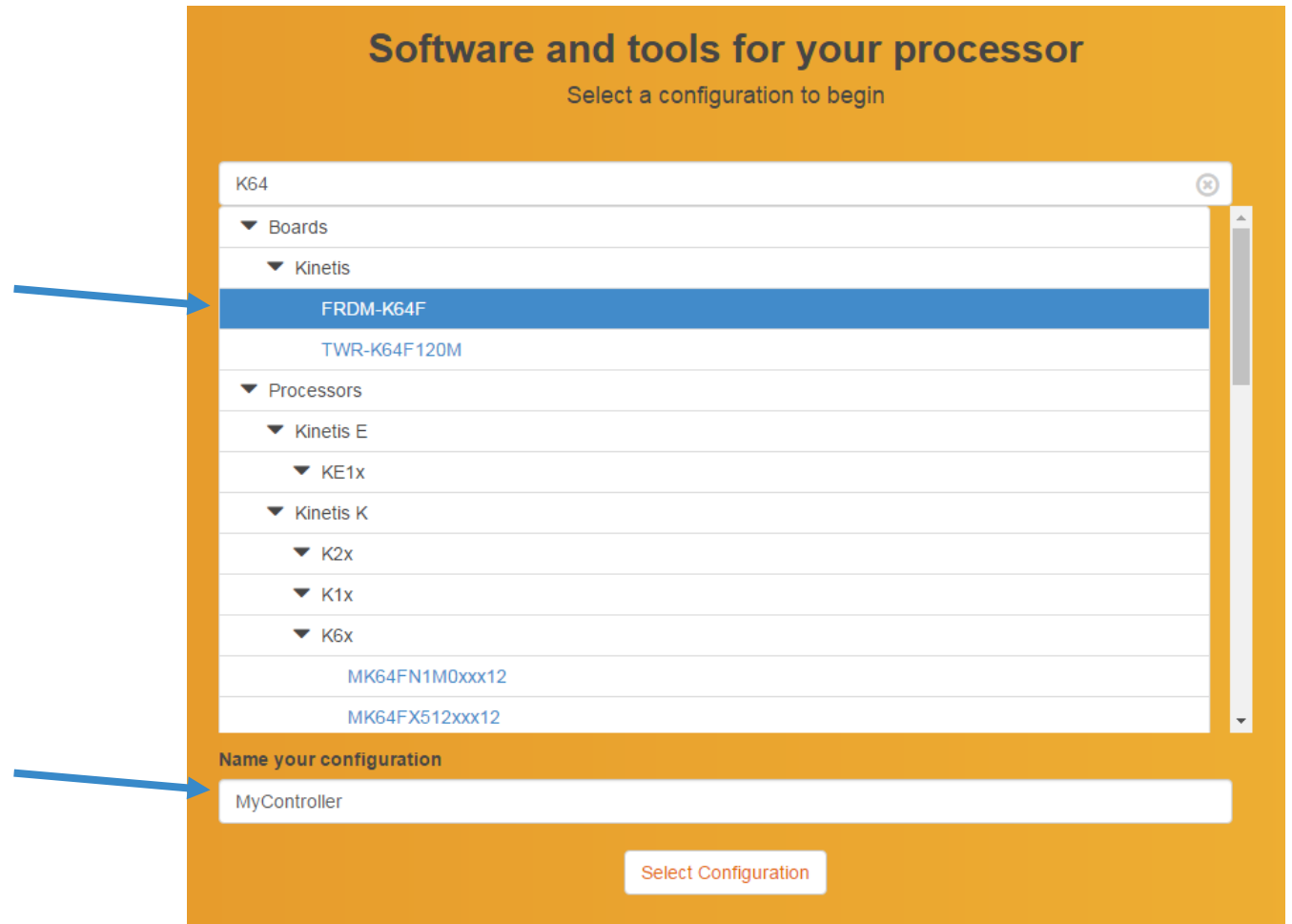
◀ **SDK Builder - This is the debut release of the online SDK Builder for Kinetis, and makes available the new Kinetis SDK v2.0 (for a select set of key Kinetis devices).** ▶



ack

Kinetis Expert Tool – Download KSDK

- After logging in, select the board or device
- Optionally rename the configuration



Kinetis Expert Tool – Download Kinetis SDK

1. SDK Builder page
2. Shows configuration
3. Select optional items
4. Select the configuration options for RTOS, Host OS, toolchain, SDK version, and a unique package name
5. Then click on “Build SDK Package” button

The screenshot displays the NXP Kinetis Expert interface. At the top, the 'Kinetis Expert' header shows the selected board 'MyController (FRDM-K64F)'. The main content area is titled 'Kinetis SDK' and provides a summary of the configuration. On the left, a list of board specifications is shown for the FRDM-K64F: Device (MK64FN1M0VLL12), Core Type (Cortex-M4F), Memory Size (1024 KB Flash, 256 KB RAM), and Maximum CPU Frequency (120 MHz). The central section lists items included in the SDK (e.g., MCU platform support, demo applications, FatFS, USB stack, IWP TCP/IP, documentation) and optional items (FreeRTOS, uC/OS-II, uC/OS-III). Below this, a summary states: 'Your custom version of the Kinetis SDK is now ready to be packaged! Click the button below to complete the process.' This is followed by a form with fields for 'Package name' (SDK_2.0_FRDM-K64F), 'SDK version' (SDK 2.0), 'Supported toolchain(s)' (Kinetis Design Studio), and 'Host OS' (Windows). A prominent orange 'Build SDK Package' button is located below the form. At the bottom, there is a section for 'Other Software Libraries' with a link to 'wolfSSL'. Five blue callout bubbles with numbers 1 through 5 point to specific elements: 1 points to the SDK Builder page title, 2 points to the board selection dropdown, 3 points to the optional items section, 4 points to the 'Build SDK Package' button, and 5 points to the 'Build SDK Package' button.

Kinetis Expert Tool – Package Generation

System may take some time to generate a package. Some configurations are pre-cached, but some may need to be generated

- Generally takes about 5 minutes

Build SDK Package [SDK API Documentation v2.0](#)

Building! In general, SDK builds should complete within a few minutes. However, depending on the complexity of the configuration and bandwidth of the build system, specific build may take up to 30 minutes to complete.




Kinetis Expert Tool – Software Vault

- Once package is available for download, it will be found under the “Software Vault” tab
- Download package by clicking on the Download icon

NXP Kinetis Expert FRDM-K64F (FRDM-K64F) English Anthony

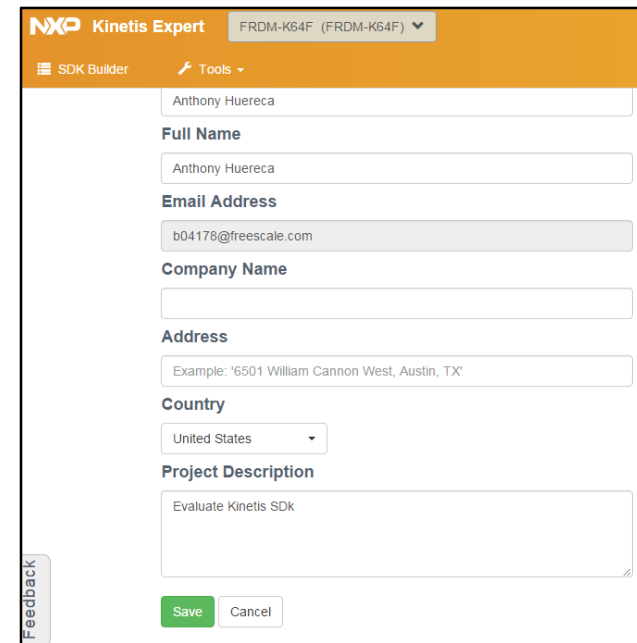
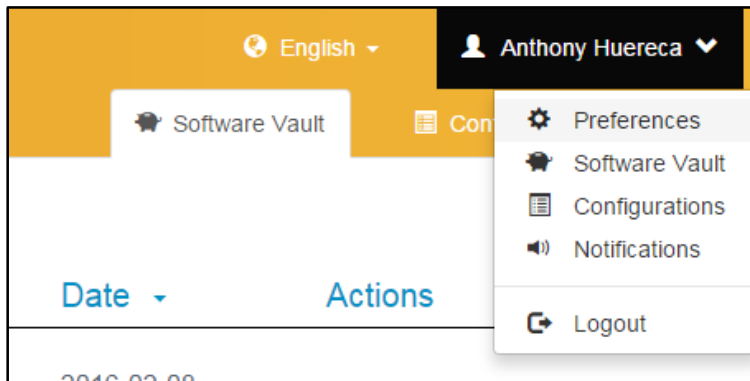
SDK Builder Tools **Software Vault** Configuration

File Vault

Name	Configuration	Date	Actions
 SDK_2.0_FRDM-K64F Board: FRDM-K64F, SDK version: KSDK 2.0.0, OS: Windows, Toolchain: ALL, Selected optional items: FreeRTOS (109MB)	FRDM-K64F	2016-02-08 06:44 AM GMT	 

Kinetis Expert Tool – Download Link

- If the download icon is grayed out, you may need to set the Project Description filed in your preferences.
 - There will be a link, or you can access it by clicking on your name in the upper right hand corner
 - Fill out Project Description and hit Save. Then go back to the Software Vault to download







A screenshot of the Kinetis Expert preferences form. The form is titled 'Kinetis Expert' and 'FRDM-K64F (FRDM-K64F)'. It contains several input fields: 'Full Name' (Anthony Huereca), 'Email Address' (b04178@freescale.com), 'Company Name', 'Address' (with an example: '6501 William Cannon West, Austin, TX'), 'Country' (United States), and 'Project Description' (Evaluate Kinetis SDK). There are 'Save' and 'Cancel' buttons at the bottom. A 'Feedback' button is visible on the left side of the form.

Kinetis Expert Tool – Multiple Packages

You can go back to the SDK Builder tab to create other packages. These new packages will show up under the Software Vault

- In this example, I've created another package that only includes KDS projects

The screenshot shows the NXP Kinetis Expert Software Vault interface. The top navigation bar includes the NXP logo, 'Kinetis Expert', a dropdown menu for 'FRDM-K64F (FRDM-K64F)', 'English', and a user profile icon. Below the navigation bar, there are tabs for 'SDK Builder', 'Tools', 'Software Vault', and 'Conf'. The main content area is titled 'File Vault' and displays a table of software packages.

Name	Configuration	Date	Actions
 SDK_2.0_FRDM-K64F-KDS_only Board: FRDM-K64F, SDK version: KSDK 2.0.0, OS: Windows, Toolchain: KDS, Selected optional items: FreeRTOS (102MB)	FRDM-K64F	2016-02-08 06:54 AM GMT	 
 SDK_2.0_FRDM-K64F Board: FRDM-K64F, SDK version: KSDK 2.0.0, OS: Windows, Toolchain: ALL, Selected optional items: FreeRTOS (109MB)	FRDM-K64F	2016-02-08 06:44 AM GMT	 



Kinetis Expert Tool – Multiple Configurations

- If a new device or board is desired, then need to create a new configuration
- Go to the Configurations tab, and click on “New Configuration...”
 - Or use drop-down box at top of the screen



The screenshot shows the NXP Kinetis Expert tool interface. The top navigation bar is orange and contains the NXP logo, 'Kinetis Expert', a dropdown menu for 'FRDM-K64F (FRDM-K64F)', 'English', and 'Anthony Huereca'. Below the navigation bar, there are icons for 'SDK Builder', 'Tools', 'Software Vault', and 'Configurations'. The 'Configurations' tab is active and highlighted with a red box. Below the navigation bar, the 'Configurations' section is displayed. It features a table with columns for 'Name', 'Board', 'Device', and 'Actions'. The table contains one row with the following data: 'FRDM-K64F' (with an edit icon), 'FRDM-K64F', 'MK64FN1M0xxx12', and a red 'x' icon. Below the table, there is a button labeled 'New Configuration ...' with a plus icon, which is also highlighted with a red box.

Name	Board	Device	Actions
FRDM-K64F 	FRDM-K64F	MK64FN1M0xxx12	

[+ New Configuration ...](#)

Kinetis Expert Tool – Multiple Configurations

Switch between boards/devices on the Configurations tab or via drop down box

NXP Kinetis Expert FRDM-K82F (FRDM-K82F) English Anthony Huereca

SDK Builder Tools Software Vault **Configurations**

Configurations

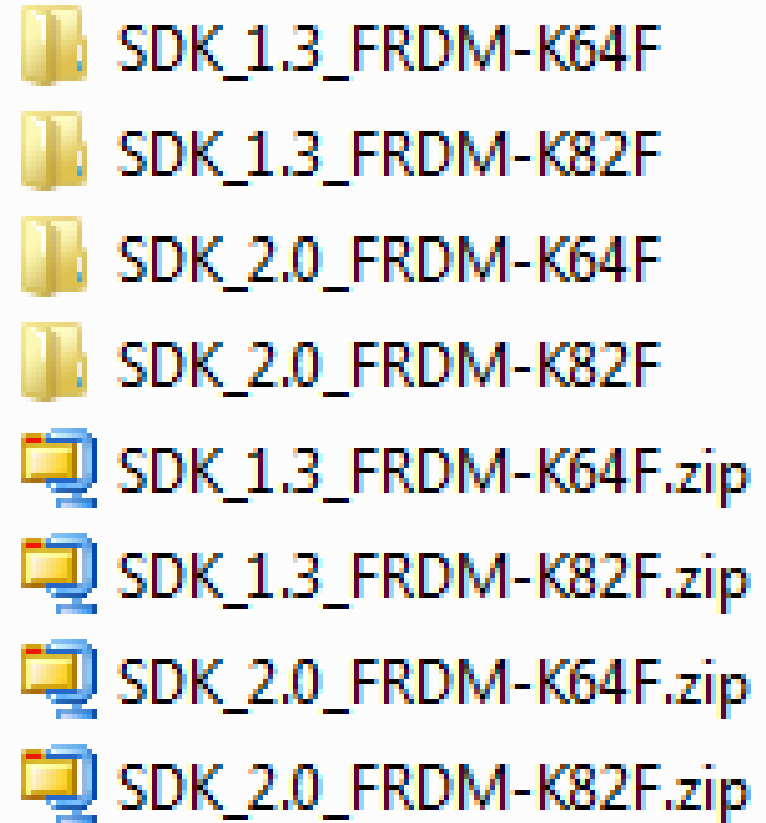
Name	Board	Device	Actions
FRDM-K64F	FRDM-K64F	MK64FN1M0xxx12	✖
FRDM-K82F	FRDM-K82F	MK82FN256xxx15	✖

[+ New Configuration ...](#)

Kinetis Expert Tool – Extract Package

The download will be a zip file to download and unzip anywhere on your computer

- Path should not have spaces
- Consider using “C:\nxp” as the root location
 - KDS and KSDK Project Generator look there first for KSDKs

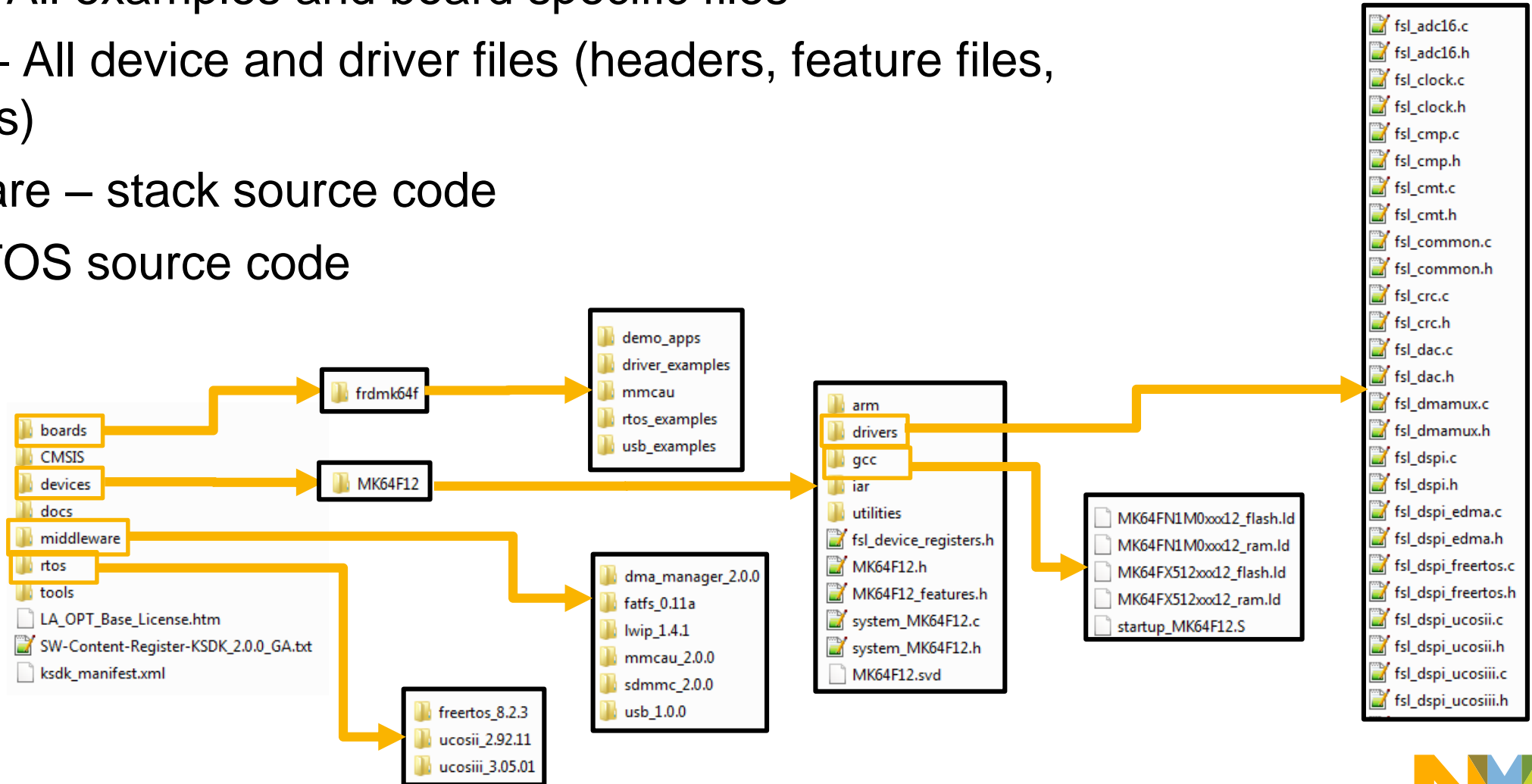


KSDK 2.0 STRUCTURE



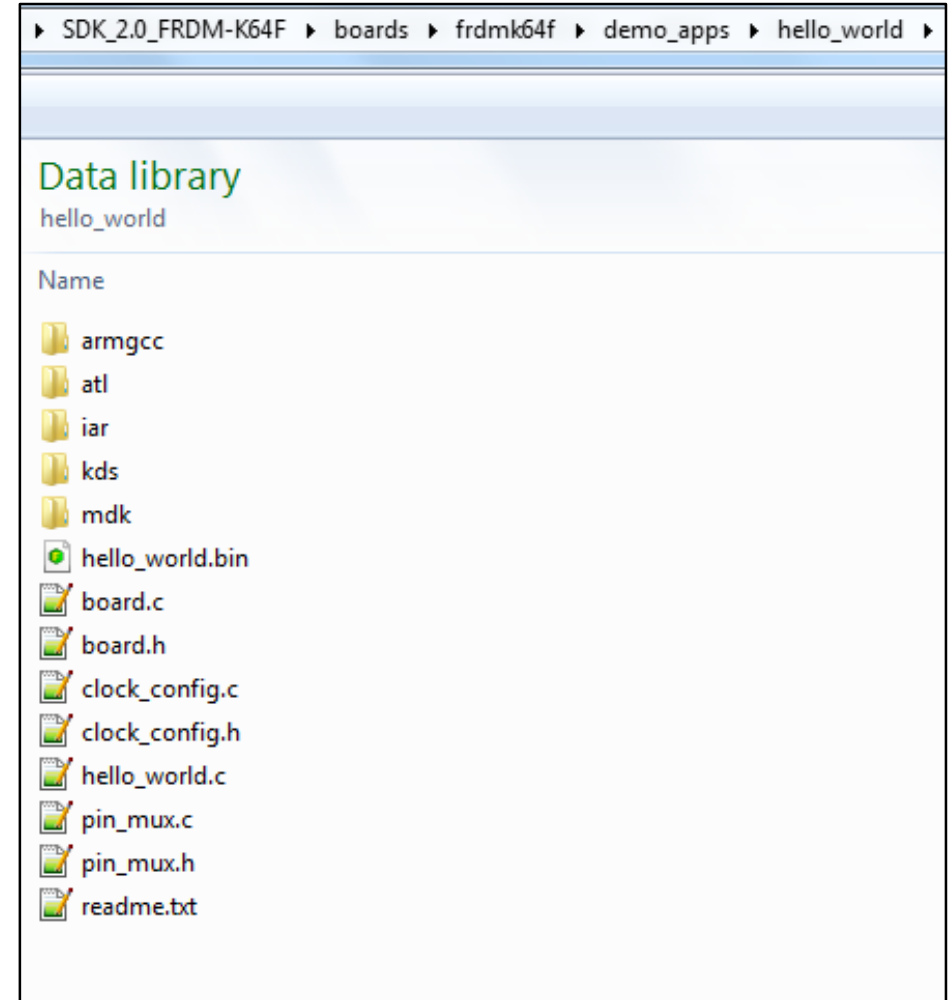
KSDK 2.0 File Structure

- boards – All examples and board specific files
- devices – All device and driver files (headers, feature files, linker files)
- middleware – stack source code
- rtos – RTOS source code



KSDK 2.0 File Structure – Examples

- Each example application has its own unique copy of the board, pin_mux, and clock_config files.
- These files are no longer shared among all examples like they were in KSDK 1.3
- Also each example also contains a pre-compiled .bin file for easy drag-and-drop programming



KSDK 2.0 File Structure – Examples

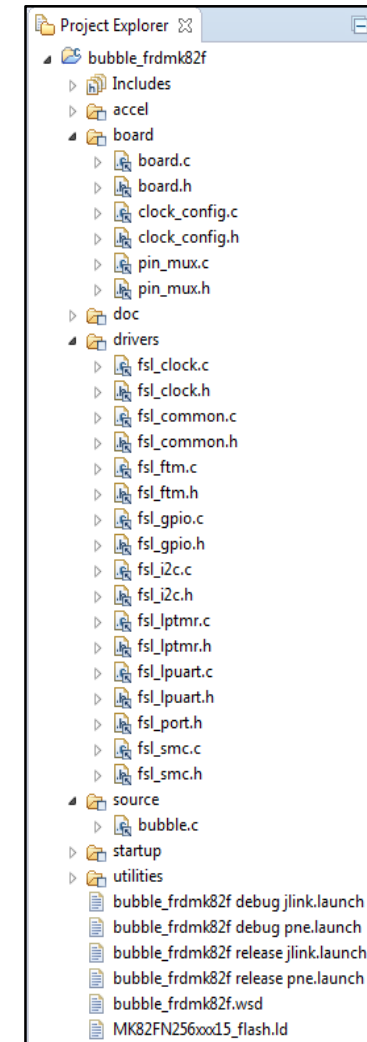
- Most configuration settings are in board.h file
 - UART module
 - UART baud
 - GPIO pins defined

- Default UART pins defined in pin_mux.c in BOARD_InitPins().

```
44  /* The UART to use for debug messages. */
45  #define BOARD_DEBUG_UART_TYPE DEBUG_CONSOLE_DEVICE_TYPE_UART
46  #define BOARD_DEBUG_UART_BASEADDR (uint32_t) UART0
47  #define BOARD_DEBUG_UART_CLKSRC SYS_CLK
48  #define BOARD_DEBUG_UART_CLK_FREQ CLOCK_GetCoreSysClkFreq()
49  #define BOARD_UART_IRQ UART0_RX_TX_IRQn
50  #define BOARD_UART_IRQ_HANDLER UART0_RX_TX_IRQHandler
51
52  #ifndef BOARD_DEBUG_UART_BAUDRATE
53  #define BOARD_DEBUG_UART_BAUDRATE 115200
54  #endif /* BOARD_DEBUG_UART_BAUDRATE */
55
56  /* Define the port interrupt number for the board switches */
57  #define BOARD_SW2_GPIO GPIOC
58  #define BOARD_SW2_PORT PORTC
59  #define BOARD_SW2_GPIO_PIN 6U
60  #define BOARD_SW2_IRQ PORTC_IRQn
61  #define BOARD_SW2_IRQ_HANDLER PORTC_IRQHandler
62  #define BOARD_SW2_NAME "SW2"
```

Kinetis SDK 2.0 Projects

- All source files are included in the example application projects
 - There is no longer platform libraries to compile first
- Drivers are found under the drivers folder
- Board specific files under the board folder
- Application specific files under source folder



CREATING A NEW KSDKK PROJECT



SDK Project Generator Overview

- Standalone utility designed to help users create custom SDK-based projects, within or outside of the SDK source tree
- Cross-platform, supporting Windows, Mac OS and Linux
- Highly customizable output:
 - Simple “Quick Generate” creates a barebones while(1) application in SDK tree
 - Advanced Configuration
 - Device selection by board or specific device
 - New project or clone an existing one
 - RTOS selection
 - IDE selection
 - Project can reside within or completely standalone from SDK source tree

KSDK Project Generator – All IDEs

- Download from <http://nxp.com/ksdk>
- Unzip
- Execute and point to root KSDK installation path

KINETIS-SDK: Software Development Kit for Kinetis MCUs ☆

Overview

Documentation

Downloads

Hardware & Tools

Training & Support

Filter By | [Show All](#)

Recommended Software & Tools (2)

Software Development Tools (3)

Initialization/Boot/Device Driver Code Generation (1)

Software Development Kits (2)

Run-time Software (2)

Middleware - Libraries (2)

Filter Software & Tools by Keyword

Recommended Software & Tools (2)



[Kinetis SDK Builder \(REV 2 & 1.3\)](#) New

Kinetis Expert: Software Development Kit for Kinetis MCUs, Online SDK Builder
Software Development Kits

HTML (154 B) Kinetis SDK Builder

1/28/2016

Download



[Kinetis SDK \(REV 1.x\)](#) Updated

Software Development Kit for Kinetis MCUs (Pre-configured releases)
Software Development Kits

HTML (224 B) Kinetis SDK

1/28/2016

Download

Initialization/Boot/Device Driver Code Generation (1)



[Kinetis SDK Project Generator Tool \(REV 2\)](#) Updated

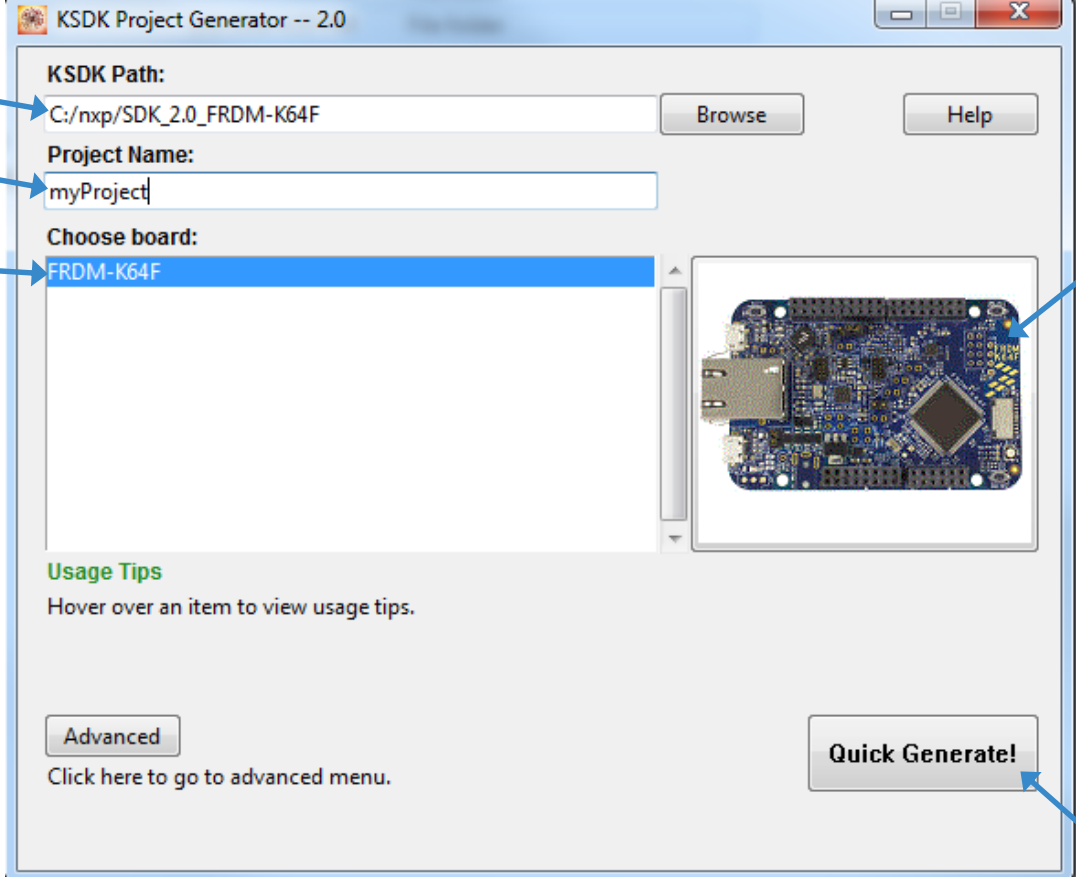
Quickly and easily create new Kinetis SDK (v1 & v2) projects, or clone existing projects.

ZIP (24.9 MB) Kinetis SDK Project Generator Tool

2/3/2016

Download

SDK Project Generator – Quick Generate



The screenshot shows the KSDK Project Generator -- 2.0 window. It features several input fields and buttons. Annotations with blue arrows point to specific elements:

- Point to SDK installation:** Points to the **KSDK Path:** text box containing `C:/nxp/SDK_2.0_FRDM-K64F`.
- Name your project:** Points to the **Project Name:** text box containing `myProject`.
- Select your board:** Points to the **Choose board:** list box where `FRDM-K64F` is selected.
- Click on board image to visit the board's product page:** Points to a photograph of the FRDM-K64F development board.
- Create your project!:** Points to the **Quick Generate!** button.

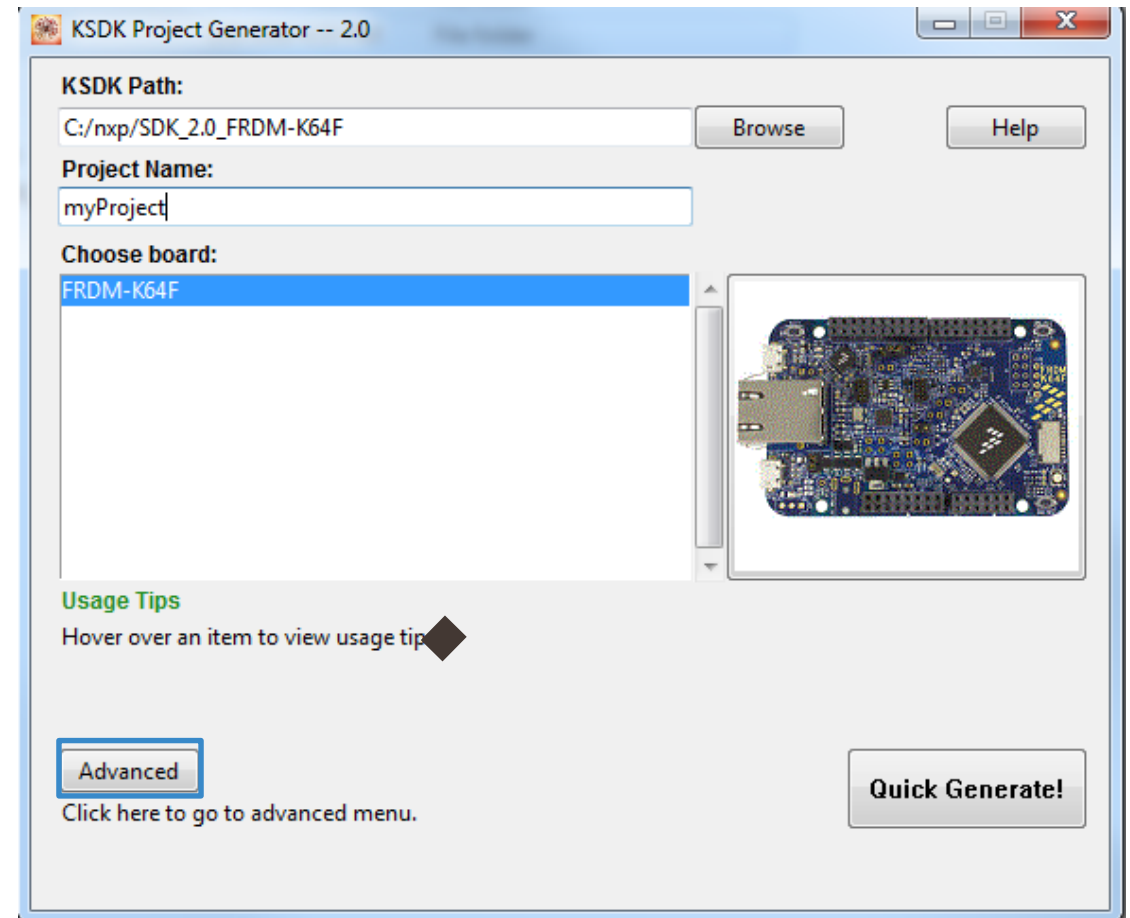
Other visible elements include a **Browse** button next to the KSDK Path, a **Help** button, an **Advanced** button, and a **Usage Tips** section with the text: "Hover over an item to view usage tips."

Output is placed in the SDK tree in:

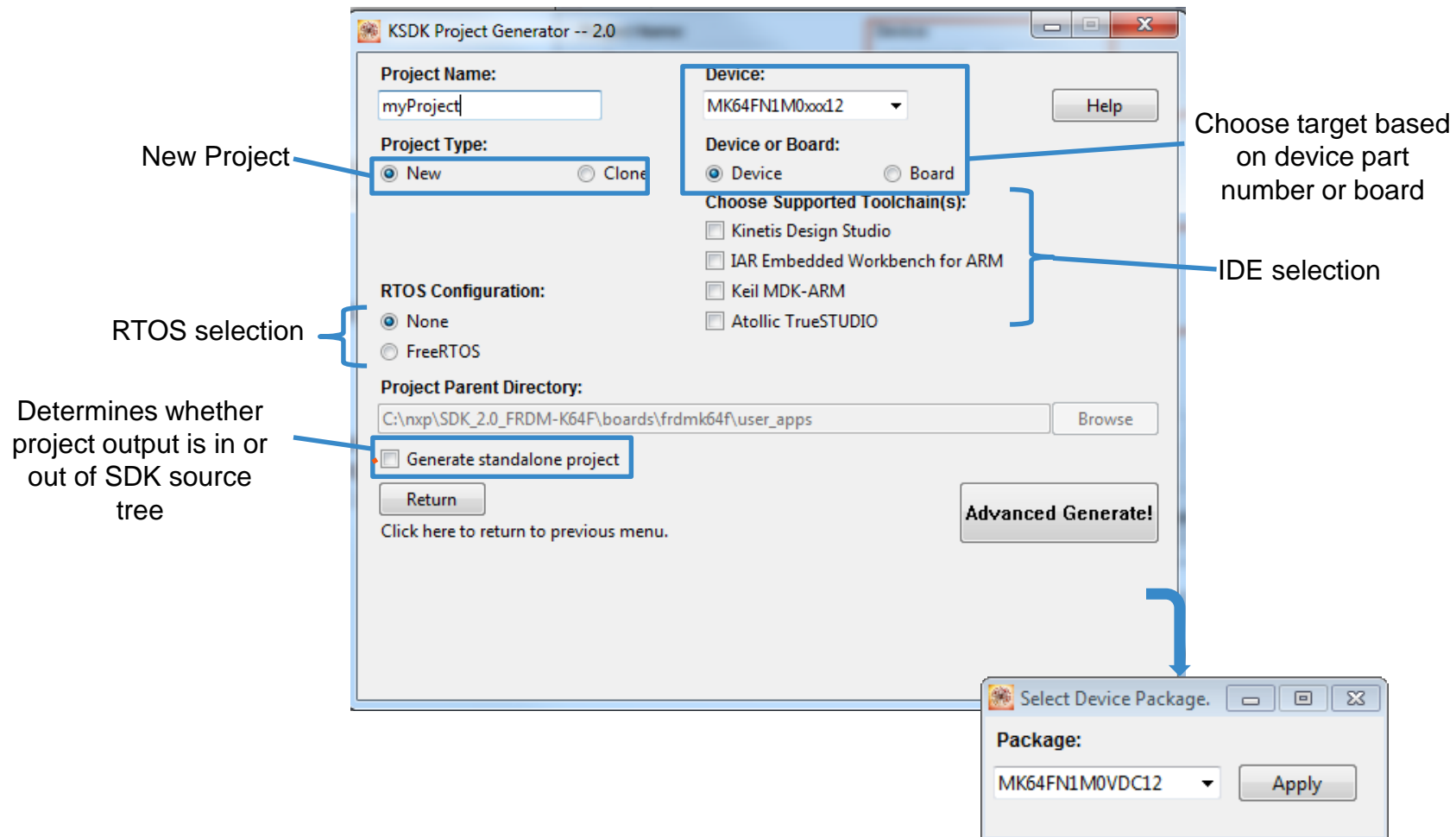
`<SDK_Install_Directory>\boards\<board_name>\user_apps\<project_name>`

SDK Project Generator – Advanced Options

- Click on the **Advanced** button to view advanced options
- Project name and board selection are carried over

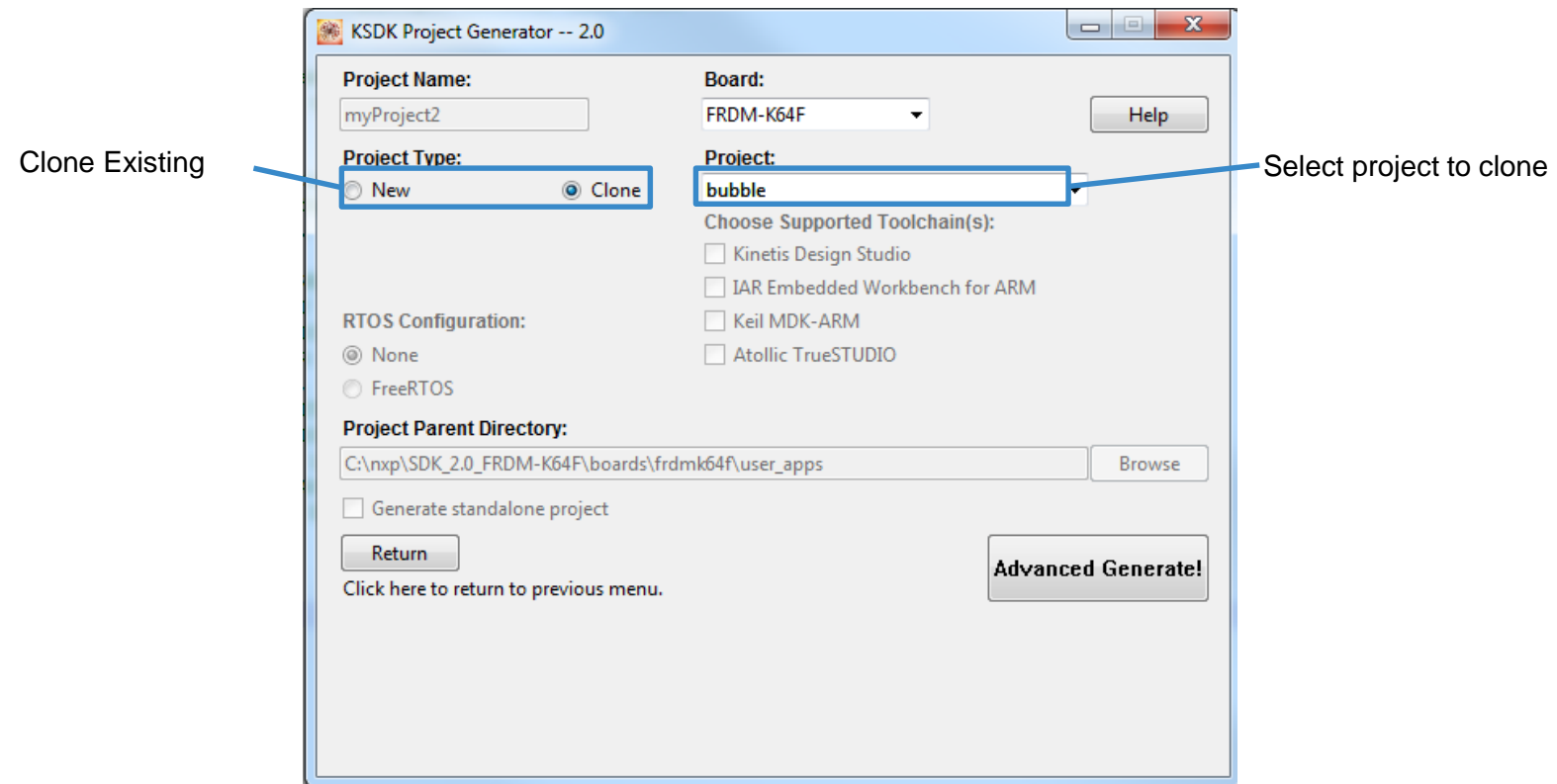


SDK Project Generator – Advanced, New Project



SDK Project Generator – Advanced, Clone Existing Project

- If cloning project, can only clone project once.
 - Work-around is to rename project on hard drive.



LAB #1



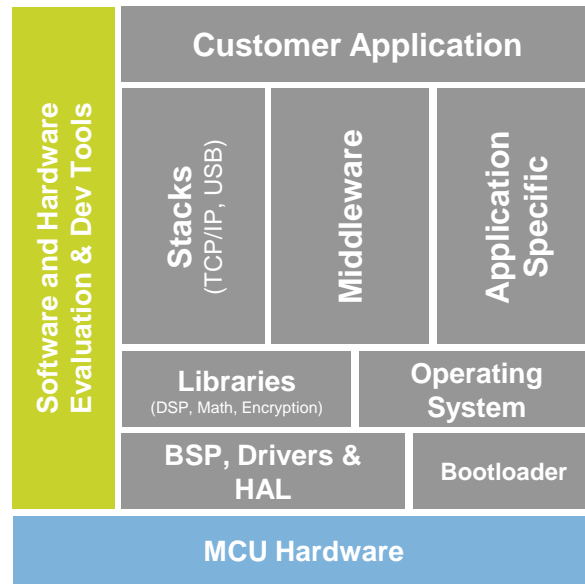
Freedom Development Platforms



Low-cost/low-power development hardware



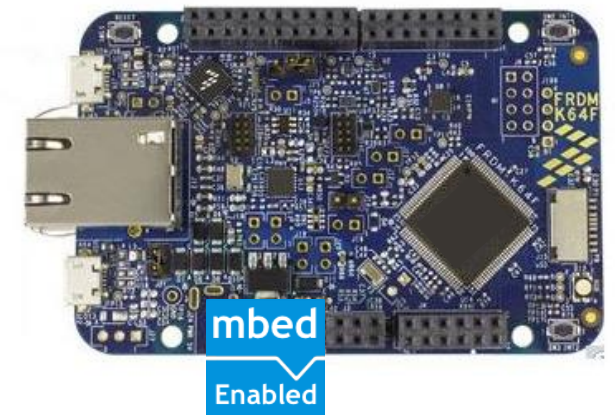
Enables quick application prototyping and demonstration of **Kinetis MCU families**



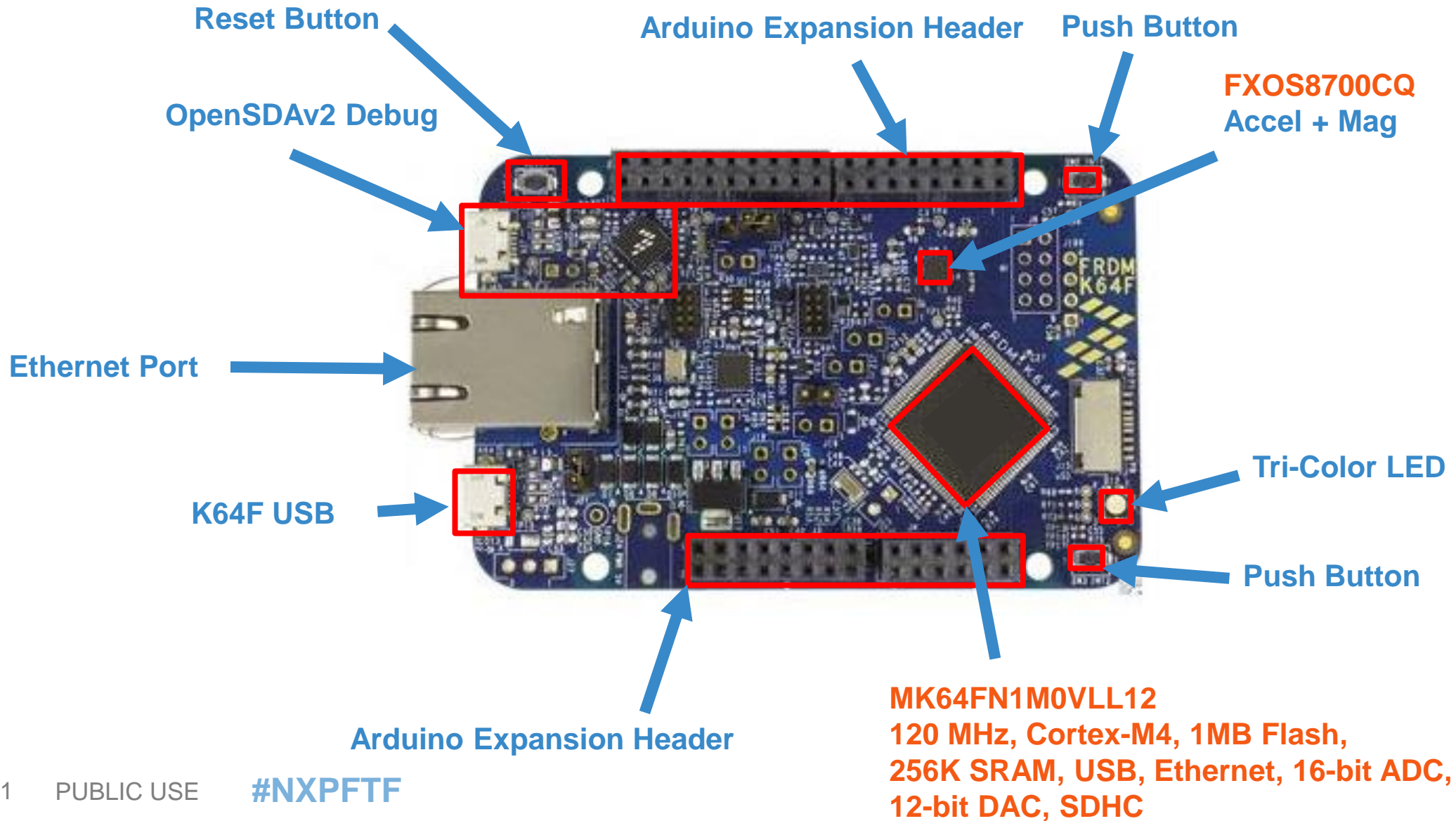
Product Features

- Low-cost (starting at \$12.95 USD)
- Designed in an industry-standard compact form factor (Arduino R3)
- Easy access to the MCU I/O pins
- Integrated open-standard serial and debug interface (OpenSDA)
- Compatible with a rich-set of third-party expansion boards

FRDM-K64F:



FRDM-K64F Hardware Overview



Lab 1 Overview

Objective:

This lab explains how to import and build the demos that are bundled with Kinetis SDK

Lab Flow:

- Importing demo project
- Build Demo
- Download and Debug
- Use Project Cloner Tool to create new project to be used in next lab
- Explore Project Options

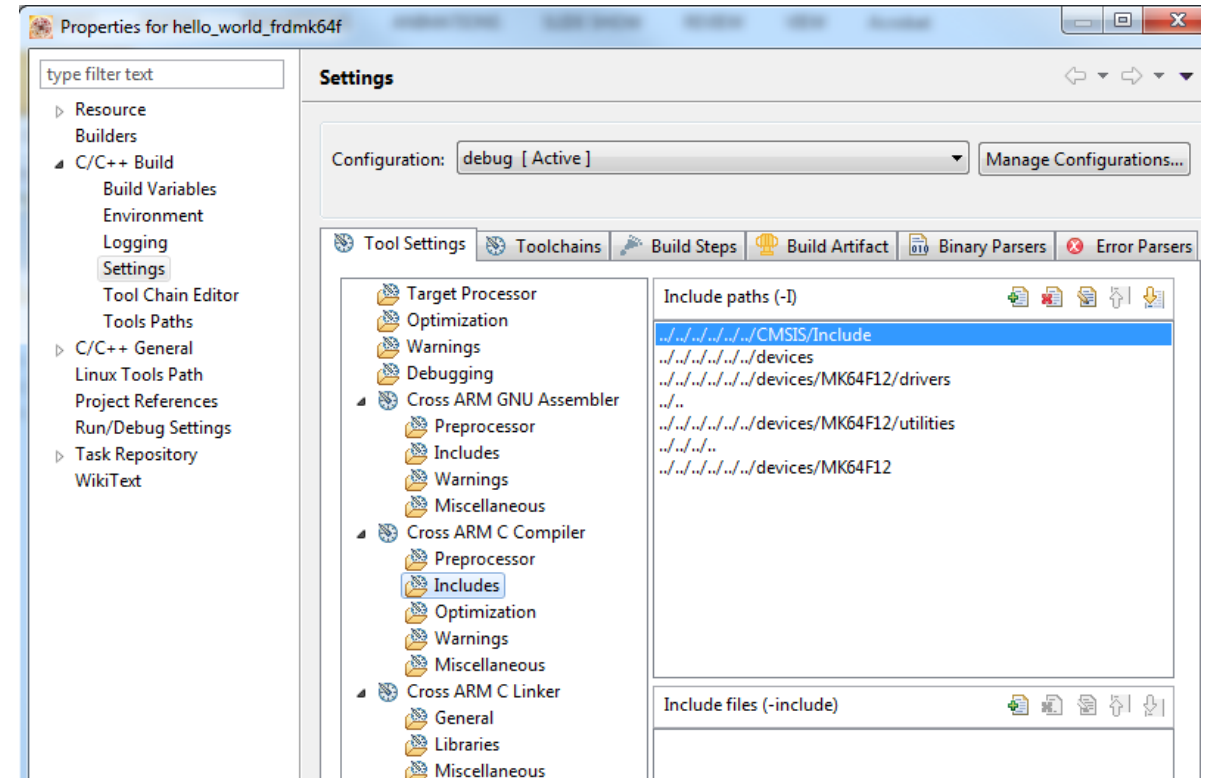
Required Hardware and Software:

- FRDM-K64F Board configured with CMSIS-DAP Debugger
- Kinetis Design Studio (v3.0 or newer)
- Kinetis Software Development Kit (v2.0)
- Mbed serial driver installed
- Terminal Program (ie TeraTerm or PuTTY)
- Micro USB Cable



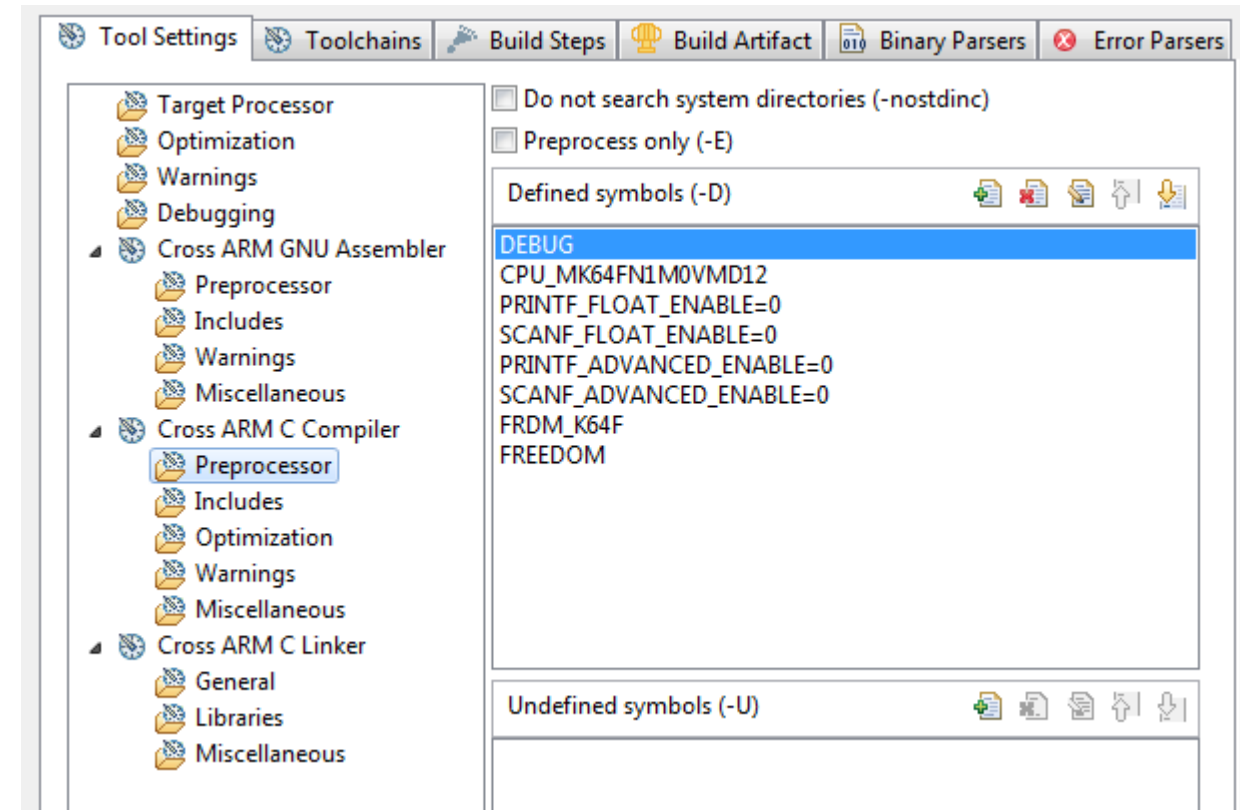
KSDK Project Information

- Right click on project and select Properties
- Navigate to the C/C++ Build->Settings page
- Look at the Cross ARM C Compiler->Includes screen to see how the KSDK directories are included



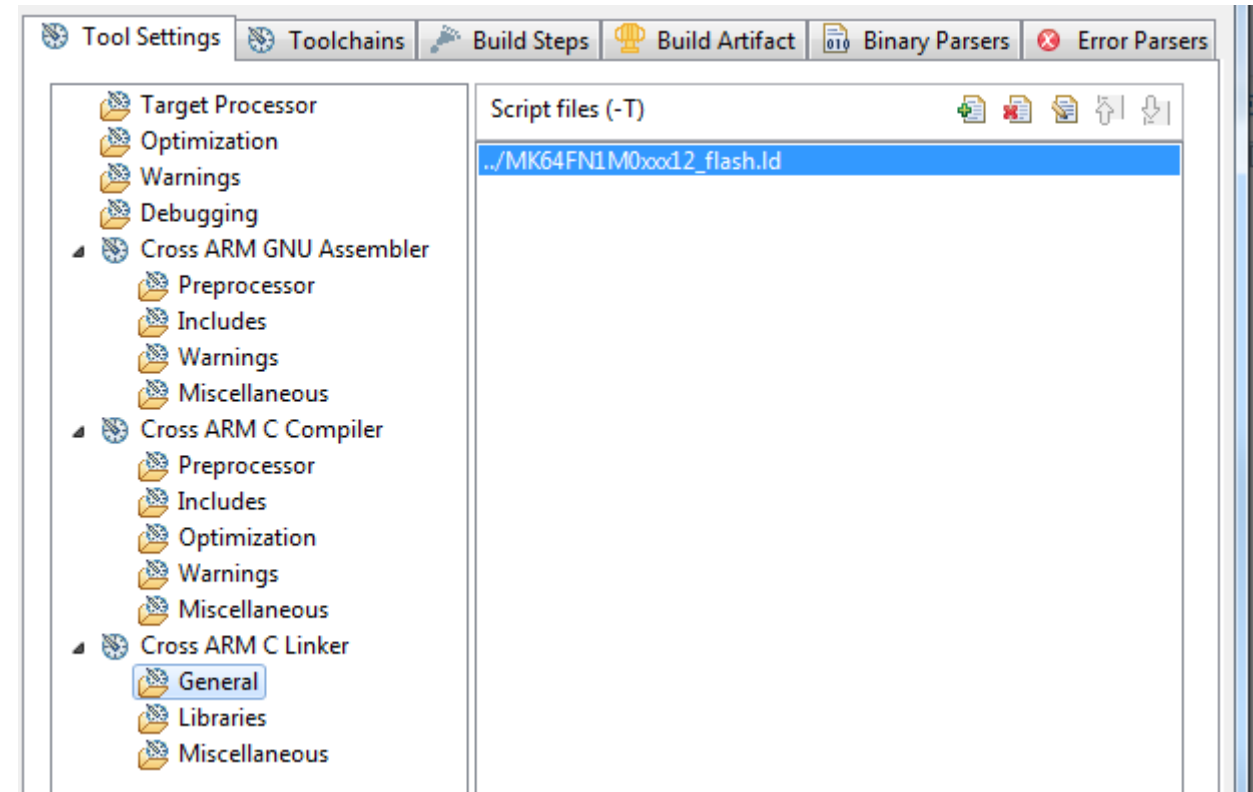
KSDK Project Information Continued

- Look at the Preprocessor screen to see the various KSDK defines



KSDK Project Information Continued

- Linker File



KINETIS EXPERT SYSTEM CONFIGURATION TOOLS



Kinetis Expert (KEx) System Configuration Tools

Kinetis Expert is a suite of evaluation and configuration tools that helps guide users from first evaluation to production software development. The tools are available in online and desktop editions.



SDK Builder packages custom SDKs based on user selections of MCU, evaluation board, and optional software components.



Project Generator creates new or clones existing SDK projects.



Power Estimation tool provides energy and battery-life estimates based on a user's application model



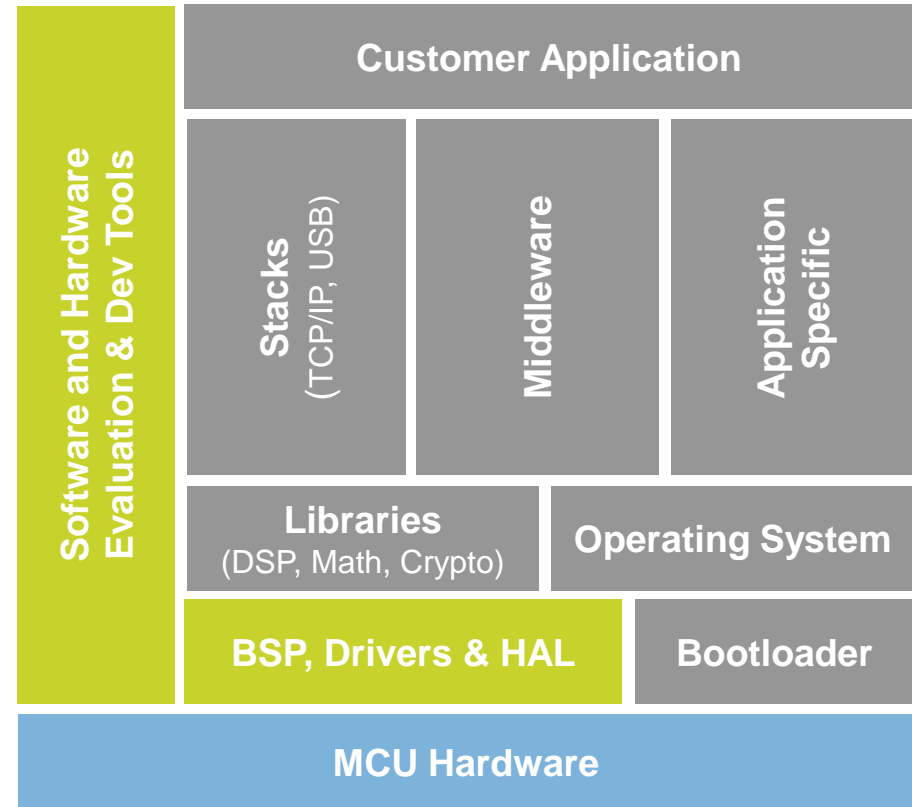
Power Analyzer measures and displays energy consumption data



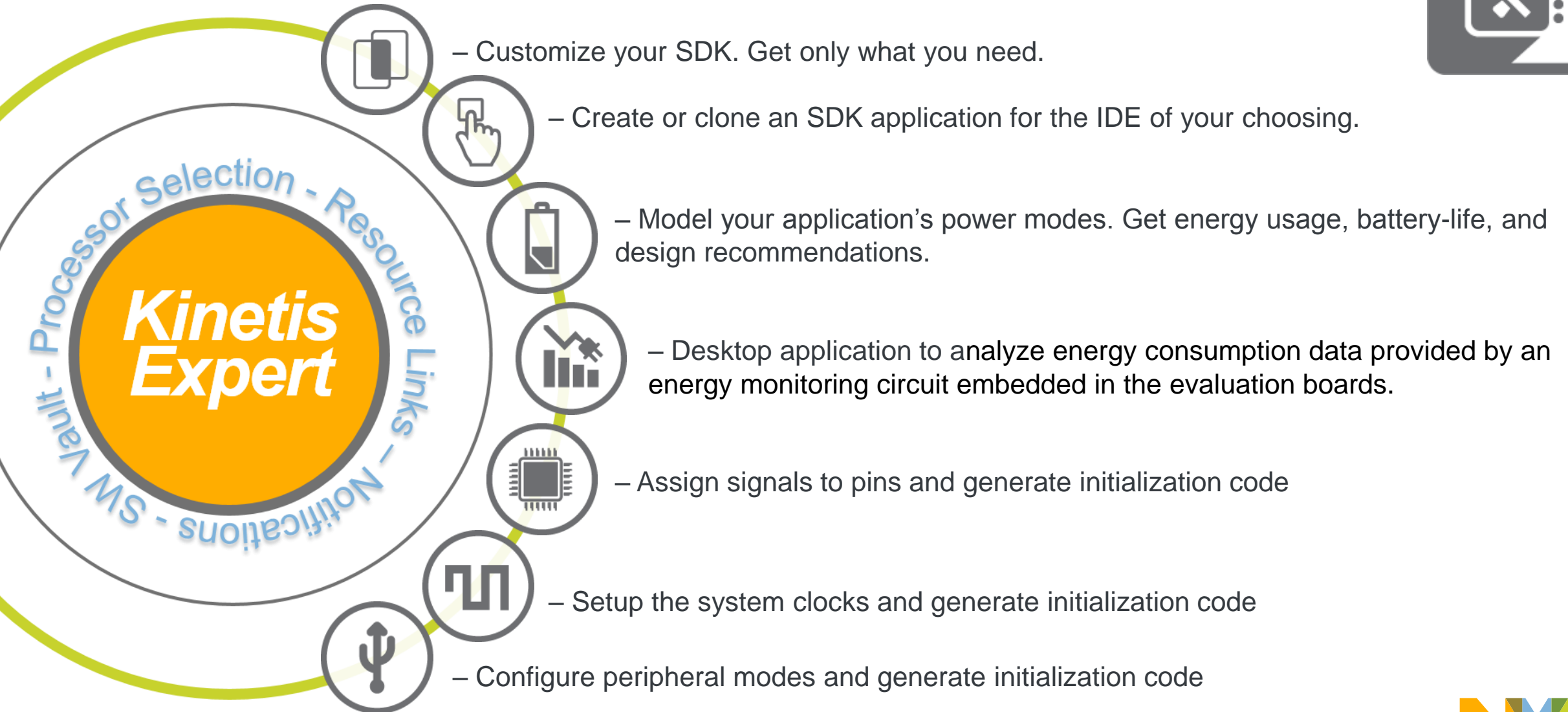
Pins, Clocks, and Peripherals tools generate initialization C code for custom board support.



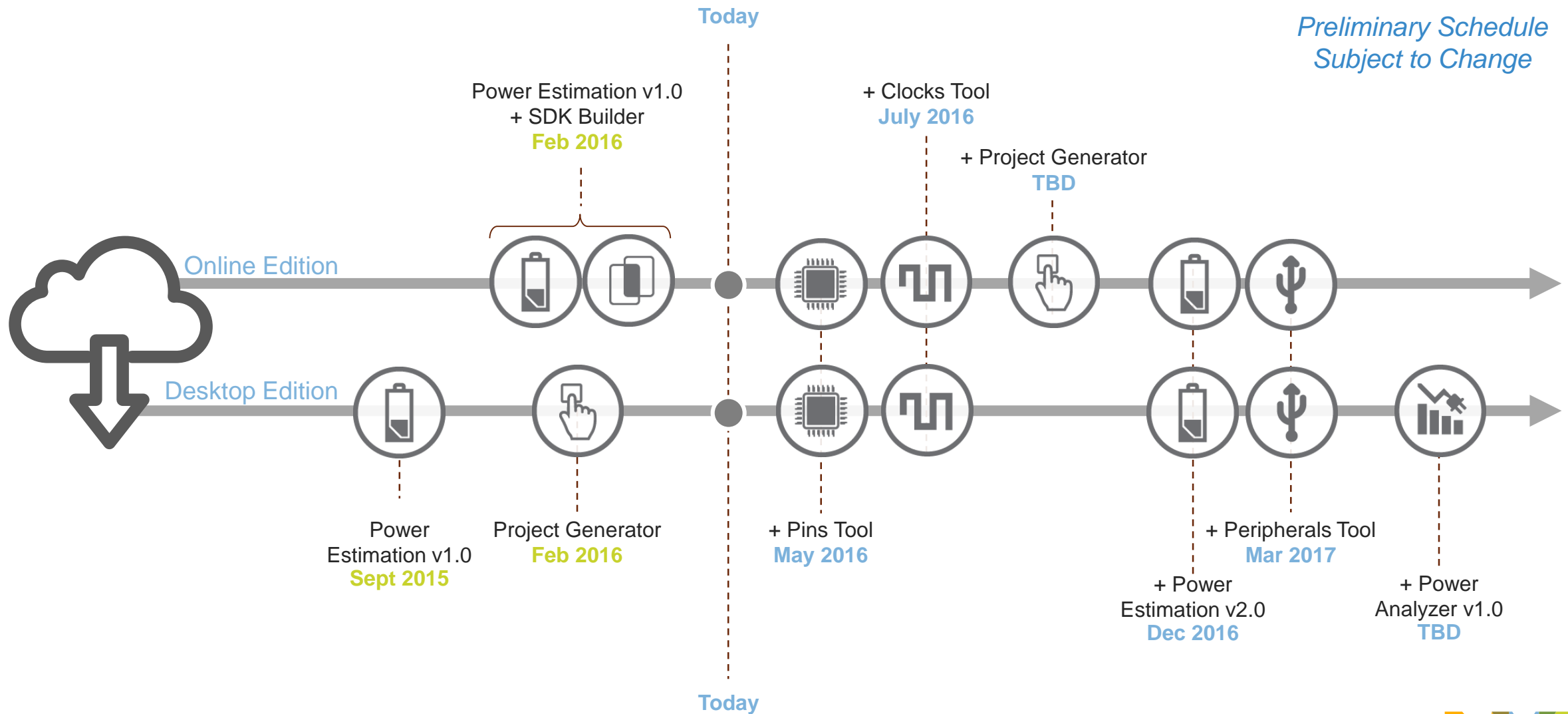
Integrated configuration and development tools for Kinetis MCUs.



Kinetis Expert (KEx) System Configuration Tools



Kinetis Expert (KEx) – 2016 Milestones



Kinetis Expert Power Estimation Tool

Product Features:

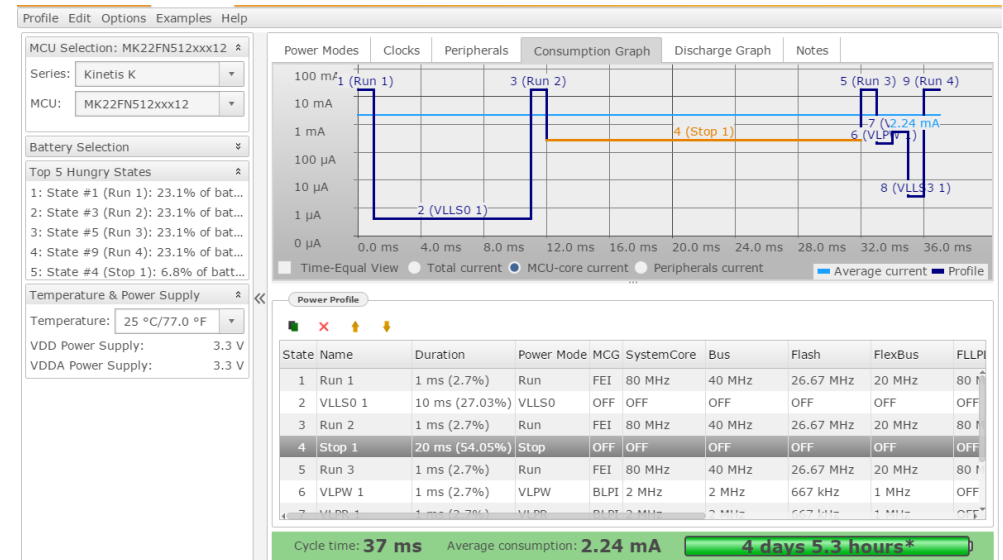
- Part of the Kinetis Expert suite of system configuration tools
- Online and Desktop versions available now
- Models application states and estimates the power profile
- Provides immediate energy consumption & battery life estimations
- Generates consumption and battery discharge graphs
- Provides ability to save & load profiles and generate reports
- Local and online versions to be available
- English & limited Chinese language support
- Backed by real power measurement data
- Quickly evaluate which Kinetis MCU fits your use-case and power budget
- Accelerates learning curve for advanced power management features
- Ideal tool for developing wearable and other battery-operated applications.



Estimate and optimize your system's power consumption



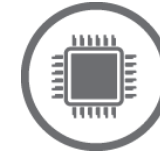
Helps you design for efficient use of energy



Kinetis Expert Pins Tool

Product Features:

- Part of the Kinetis Expert suite of system configuration tools
- Online and Desktop editions released in May 2016
- Muxing and pin configuration with consistency checking
- ANSI-C configuration code
- Kinetis SDK support
- Graphical processor package view
- Multiple configuration blocks/functions
 - Selection of Pins and Peripherals
 - Package with IP blocks
 - Routed pins with electrical characteristics
 - Registers with configured and reset values
 - Source code for C/C++ applications
- Documented and easy to understand source code
- Report generation
- Integrates with any compiler and IDE



Easy-to-use muxing and pin assignments for Kinetis MCU's

The screenshot displays the Kinetis Expert Pins Tool interface. On the left, a 'Peripherals' list includes ADC0 through I2S0, with ADC0 and I2C0 selected. The main area shows a pin package view for 'MK64FN1M0VLQ12 - LQFP 144 package' with various pins and their assigned peripherals. A 'Routed Pins' table is visible at the bottom, showing two entries for ADC0. On the right, the 'Sources' tab shows the generated C code for 'pin_mux.c', including function definitions and configuration options.

#	Peripheral	Signal	Route to	Direction	Slew rate	Open dra.	Drive str
1	ADC0	SE, 1	[23] ADC...Input	n/a	n/a	n/a	n/a
2	ADC0	SE, 2	[3] ADC0...Input	Fast	Disabled	Low	



Kinetis Expert Clocks Tool

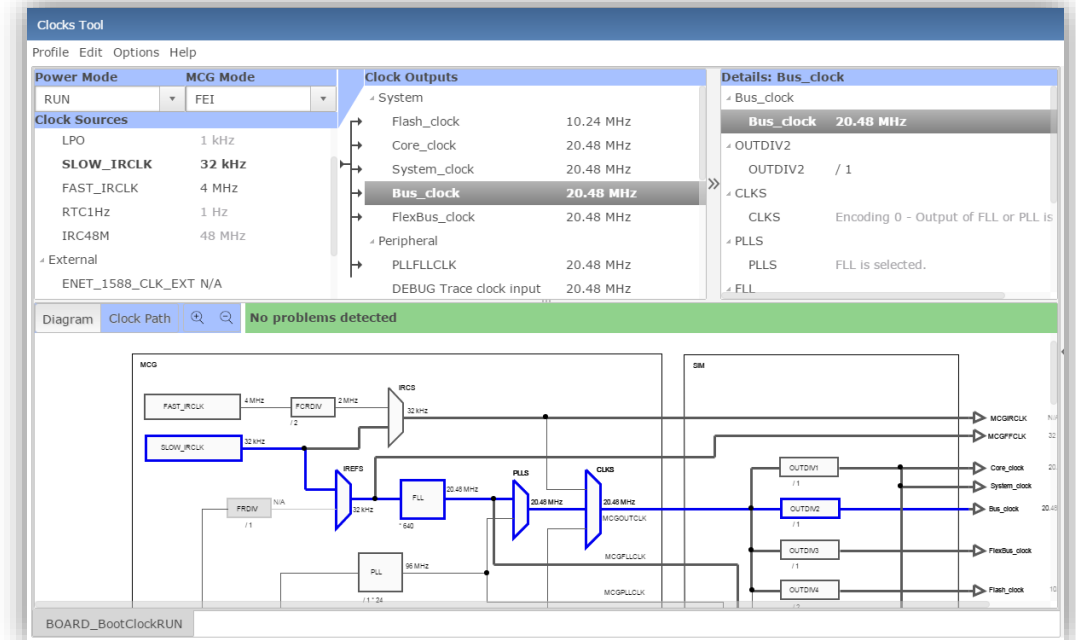
Product Features:

- Part of the Kinetis Expert system configuration tools
- Online and Desktop editions planned for release in **July 2016**
- System clock configuration with consistency checking
- ANSI-C initialization code
- Kinetis SDK v2 support
- Graphical clock diagrams
- Multiple configuration blocks/functions
- Easy-to-use guided graphical user interface
 - Selection of Clock Sources
 - Configuration of prescalers and clock outputs
 - Details and Full Diagram views with clock path
 - Registers with configured and reset values
 - Source code for C/C++ applications
- Documented and easy to understand source code
- Report generation
- Integrates with any compiler and IDE

Coming Soon



Easy-to-use clock configuration for Kinetis MCU's

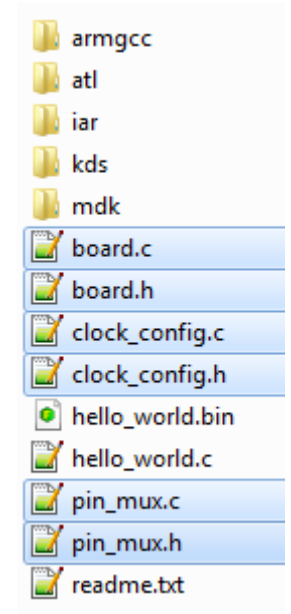


PORTING TO CUSTOM HARDWARE



Custom Board Configuration

- Each example in Kinetis SDK has board configuration files
- Found in `<KSDK_PATH>\boards\<board_name>\demo_apps\<demo_name>`
 - Contains board-specific details for Kinetis SDK
 - Applications easily portable across different boards and devices
- These files should be reviewed and modified for custom hardware:
 - board.c and board.h
 - pin_mux.c and pin_mux.h
 - clock_config.c and clock_config.h



Board Initialization

- Each project calls three configuration functions that are set in these files
 - BOARD_InitPins(); //defined in **pin_mux.c**
 - BOARD_BootClockRUN(); //defined in **clock_config.c**
 - BOARD_InitDebugConsole(); //defined in **board.c**

```
77  /* Board pin, clock, debug console init */
78  BOARD_InitPins();
79  BOARD_BootClockRUN();
80  BOARD_InitDebugConsole();
81
```


board.h File

- Defines debug UART peripheral and pins
 - For stdin/stdout functions, like printf()
- Mainly used for Kinetis SDK examples, specifying:
 - Features available on board, like sensor for demos
 - Peripheral instances for examples, like I2C0
 - Pins for LEDs and buttons

board.c File

- BOARD_InitDebugConsole()
 - Initializes debug console

pin_mux.c and pin_mux.h

- Kinetis devices provide great flexibility in muxing signals
 - Each digital port pin has up to 8 signals muxed on pin
 - Some peripherals route same signals to multiple pins
- BOARD_InitPins() inside pin_mux.c is used to initialize pins
 - Used to set pin mux options for all pins used on board

10.3.1 K64 Signal Multiplexing and Pin Assignments

144 LQFP	144 MAP BGA	121 XFBG A	100 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	EzPort
1	D3	E4	1	PTE0	ADC1_SE4a	ADC1_SE4a	PTE0	SPI1_PCS1	UART1_TX	SDHC0_D1	TRACE_CLKOUT	I2C1_SDA	RTC_CLKOUT	
2	D2	E3	2	PTE1/ LLWU_P0	ADC1_SE5a	ADC1_SE5a	PTE1/ LLWU_P0	SPI1_SOUT	UART1_RX	SDHC0_D0	TRACE_D3	I2C1_SCL	SPI1_SIN	
3	D1	E2	3	PTE2/ LLWU_P1	ADC0_DP2/ ADC1_SE6a	ADC0_DP2/ ADC1_SE6a	PTE2/ LLWU_P1	SPI1_SCK	UART1_CTS_b	SDHC0_DCLK	TRACE_D2			
4	E4	F4	4	PTE3	ADC0_DM2/ ADC1_SE7a	ADC0_DM2/ ADC1_SE7a	PTE3	SPI1_SIN	UART1_RTS_b	SDHC0_CMD	TRACE_D1		SPI1_SOUT	

K64 Sub-Family Reference Manual, Rev. 2, January 2014

GPIO Driver Uses These Defines

- `GPIO_PinInit(BOARD_LED_GPIO, BOARD_LED_GPIO_PIN, &led_config) //Init`
- `GPIO_WritePinOutput(BOARD_LED_GPIO, 1u << BOARD_LED_GPIO_PIN, 1); //Turn On`
- `GPIO_DRV_WritePinOutput(BOARD_LED_GPIO, 1u << BOARD_LED_GPIO_PIN, 0); //Turn Off`

- `PORT_SetPinConfig()` used to set more pin options
 - Pull-ups/pull-downs
 - Skew
 - etc

Kinetis SDK Porting — Change Default UART

- Modify board.h to select the UART and baud rate to use

```
44  /* The UART to use for debug messages. */
45  #define BOARD_DEBUG_UART_TYPE DEBUG_CONSOLE_DEVICE_TYPE_UART
46  #define BOARD_DEBUG_UART_BASEADDR (uint32_t) UART0
47  #define BOARD_DEBUG_UART_CLKSRC SYS_CLK
48  #define BOARD_DEBUG_UART_CLK_FREQ CLOCK_GetCoreSysClkFreq()
49  #define BOARD_UART_IRQ UART0_RX_TX_IRQn
50  #define BOARD_UART_IRQ_HANDLER UART0_RX_TX_IRQHandler
51
52  #ifndef BOARD_DEBUG_UART_BAUDRATE
53  #define BOARD_DEBUG_UART_BAUDRATE 115200
54  #endif /* BOARD_DEBUG_UART_BAUDRATE */
```

- Modify pin_mux.c to select the pins to use

```
50  /* Initialize UART0 pins below */
51  /* Ungate the port clock */
52  CLOCK_EnableClock(kCLOCK_PortB);
53  /* Affects PORTB_PCR16 register */
54  PORT_SetPinMux(PORTB, 16U, kPORT_MuxAlt3);
55  /* Affects PORTB_PCR17 register */
56  PORT_SetPinMux(PORTB, 17U, kPORT_MuxAlt3);
```

clock_config.c and clock_config.h

- Defines clock configuration and default speeds
- Defines external clock frequency
- Definition of BOARD_BootClockRUN() which is called from main() to initialize clocks.
 - Also have BOARD_BootClockVLPR() option.

```
180 void BOARD_BootClockRUN(void)
181 {
182     CLOCK_SetSimSafeDivs();
183
184     CLOCK_InitOsc0(&g_defaultClockConfigRun.oscConfig);
185     CLOCK_SetXtal0Freq(BOARD_XTAL0_CLK_HZ);
186
187     CLOCK_BootToPeeMode(g_defaultClockConfigRun.mcgConfig.oscsel, kMCG_P11ClkSelP110,
188                       &g_defaultClockConfigRun.mcgConfig.pll10Config);
189
190     CLOCK_SetInternalRefClkConfig(g_defaultClockConfigRun.mcgConfig.irclkEnableMode,
191                                  g_defaultClockConfigRun.mcgConfig.ircs, g_defaultClockConfigRun.mcgConfig.fcrdiv);
192
193     CLOCK_SetSimConfig(&g_defaultClockConfigRun.simConfig);
194
195     SystemCoreClock = g_defaultClockConfigRun.coreClock;
196 }
```

USB Hardware Porting

- USB clock set inside each USB project via `CLOCK_EnableUsbfs0Clock()`
- Modify these lines if USB clock source needs to change

```
329 #if ((defined FSL_FEATURE_USB_KHCI_IRC48M_MODULE_CLOCK_ENABLED) && (FSL_FEATURE_USB_KHCI_IRC48M_MODULE_CLOCK_ENABLED))
330     CLOCK_EnableUsbfs0Clock(kCLOCK_UsbSrcIrc48M, 48000000U);
331 #else
332     CLOCK_EnableUsbfs0Clock(kCLOCK_UsbSrcPll10, CLOCK_GetFreq(kCLOCK_Pll1FllSelClk));
333 #endif /* FSL_FEATURE_USB_KHCI_IRC48M_MODULE_CLOCK_ENABLED */
```

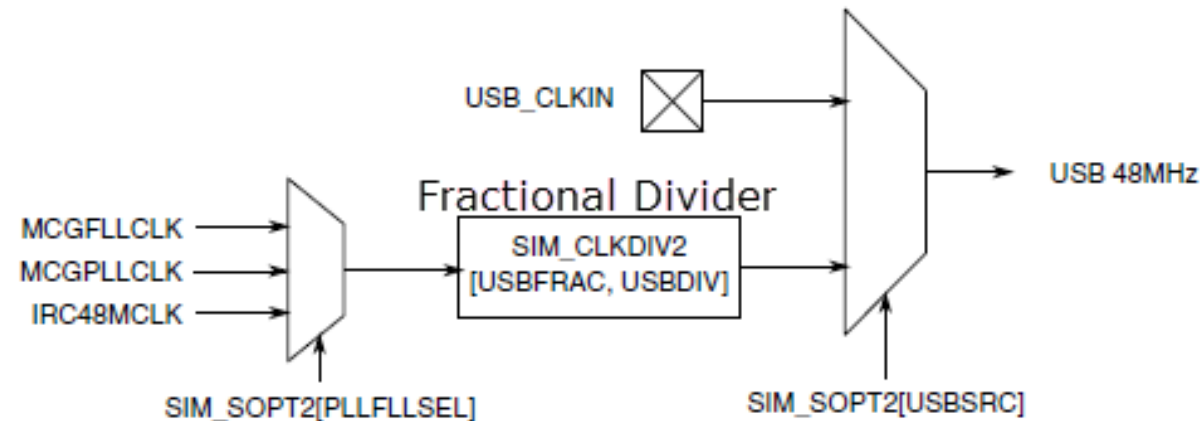


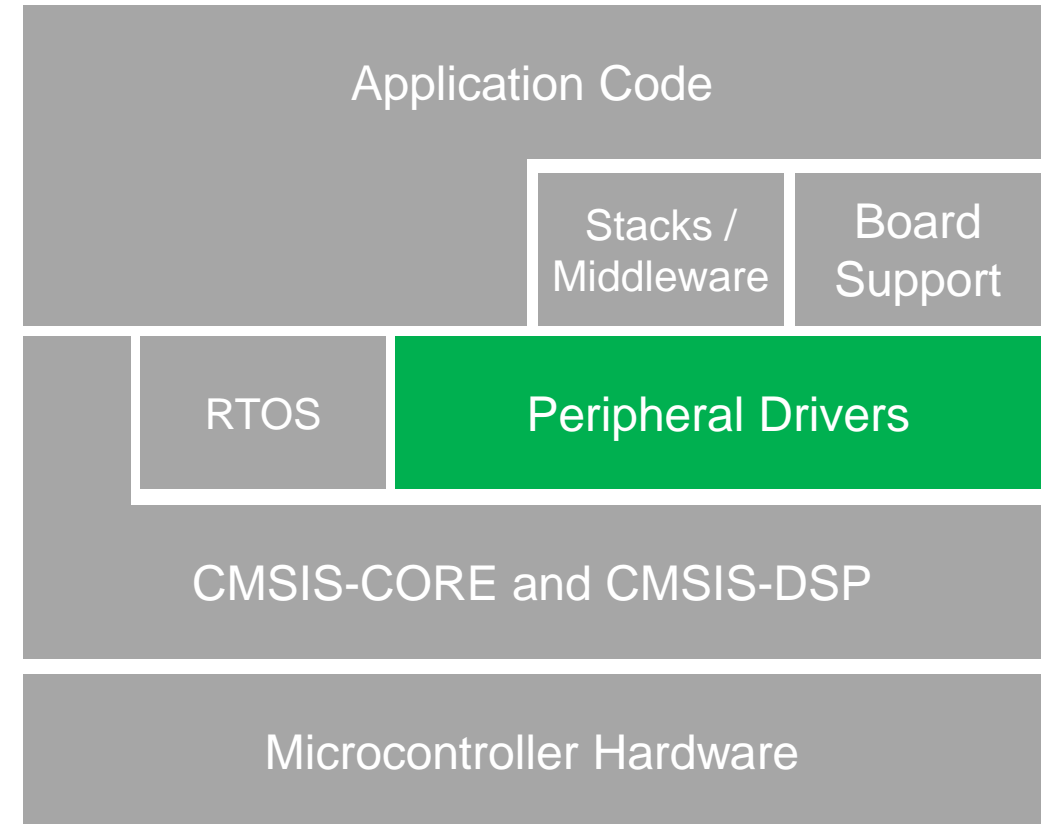
Figure 5-7. USB 48 MHz clock source

KINETIS SDK DRIVERS



Kinetis SDK v2 – Peripheral Drivers

- Kinetis SDK Drivers:
 - Single driver for each peripheral
 - Full peripheral coverage for each MCU
 - All drivers include low-level functional APIs
 - Communication peripheral drivers feature transactional APIs
 - Non-blocking, interrupt based
 - Communication peripheral drivers also have optimized RTOS wrapper drivers
 - Uses native RTOS APIs – no operating system abstraction



KSDK v2 Drivers

- Unified driver layer
 - No longer have separate HAL and Driver layer
- Designed to simplify the most common driver use-cases
 - Does not cover every peripheral feature.
- The **Kinetis SDK v.2.0 API Reference Manual** contains the details of the new API.
- Examples found in <KSDK_Installation>\boards\<board_name>\driver_examples

Low Power with Kinetis SDK

- The SMC drivers allow Kinetis SDK projects to go into low power modes
- Uses the SMC_SetPowerMode<mode> API
- Selectable wakeup sources
- See the power mode examples for details:
 - \boards\<<board_name>\demo_apps\power_manager
 - \boards\<<board_name>\demo_apps\power_mode_switch

LAB #2



Lab 2 Overview

Objective:

This lab will show how to use the GPIO and I2C drivers. You will turn on and off an LED, read accelerometer data, and explore the Kinetis SDK start-up flow.

Lab Flow:

- Modify the newly created project from the previous lab
- Build Demo
- Download and Debug

Required Hardware and Software:

- FRDM-K64F Board configured with CMSIS-DAP Debugger
- Kinetis Design Studio (v3.0 or newer)
- Kinetis Software Development Kit (v2.0)
- Mbed serial driver installed
- Terminal Program (ie TeraTerm or PuTTY)
- Micro USB Cable

KINETIS SDK 2.0 TRANSITION



Highlighted Changes from KSDK v1.x to v2

- New unified driver layer with new API
- FreeRTOS is now the main RTOS. MQX is no longer available.
 - The MQX RTCS Ethernet and MQX MFS File System stacks have been removed. lwIP and FatFs are still available for applications needing Ethernet and file system stacks.
- File structure changed
- Startup and board configuration changed
- The USB stack has been re-written and is now a BSD licensed solution
- There is no longer a separate KSDK platform library. All examples and demos consist of a single project, which include all the necessary Kinetis SDK and application files.
- The Kinetis Expert tool is replacing Processor Expert, so there is no longer Processor Expert support with KSDK 2.0.

Highlighted Changes from KSDK v1.x to v2 (Continued)

- Not all boards from 1.x are currently supported in 2.0, but will be rolled out over 2016
- Updates for Kinetis Design Studio to integrate it with Kinetis SDK are now provided via the online update tool, and no longer as part of a zip file included with Kinetis SDK. This update provides a wizard to create new SDK v2 projects inside of KDS.
- The Operating System Abstraction (OSA), Power Manager, and clock manager are no longer required by the drivers
- mbed TLS now included as part of the accelerated cryptography drivers

Kinetis SDK API

- The driver and HAL layers are now combined into a single layer
- Because of this significant change, there is not a straight forward mapping of the API modifications required
- The **Kinetis SDK v.2.0 API Reference Manual** contains the details of the new API.

Kinetis SDK Project Transition

- Most straight forward method to move project from 1.x is to create a new KSDK 2.0 project using the Kinetis SDK Project Generator tool
 - There is also the option to use the New Project wizard in Kinetis Design Studio to create this new project too.
- Then copy in the existing application code into this new project.
- Then go through the `hardware_init()` function and setup the pins and clocks using the new KSDK 2.0 API.
- Finally, the driver code utilized by the application can be re-written to be compatible with the 2.0 API.

QUESTIONS?





SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE Flex, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. © 2015–2016 NXP B.V.

