Develop with the SDK for Kinetis MCUs

APF-DES-T1016

Li Sen | SW Architecture
Chen Xinyu | Applications Engineer

M A R . 2 0 1 5

External Use

Session Objectives

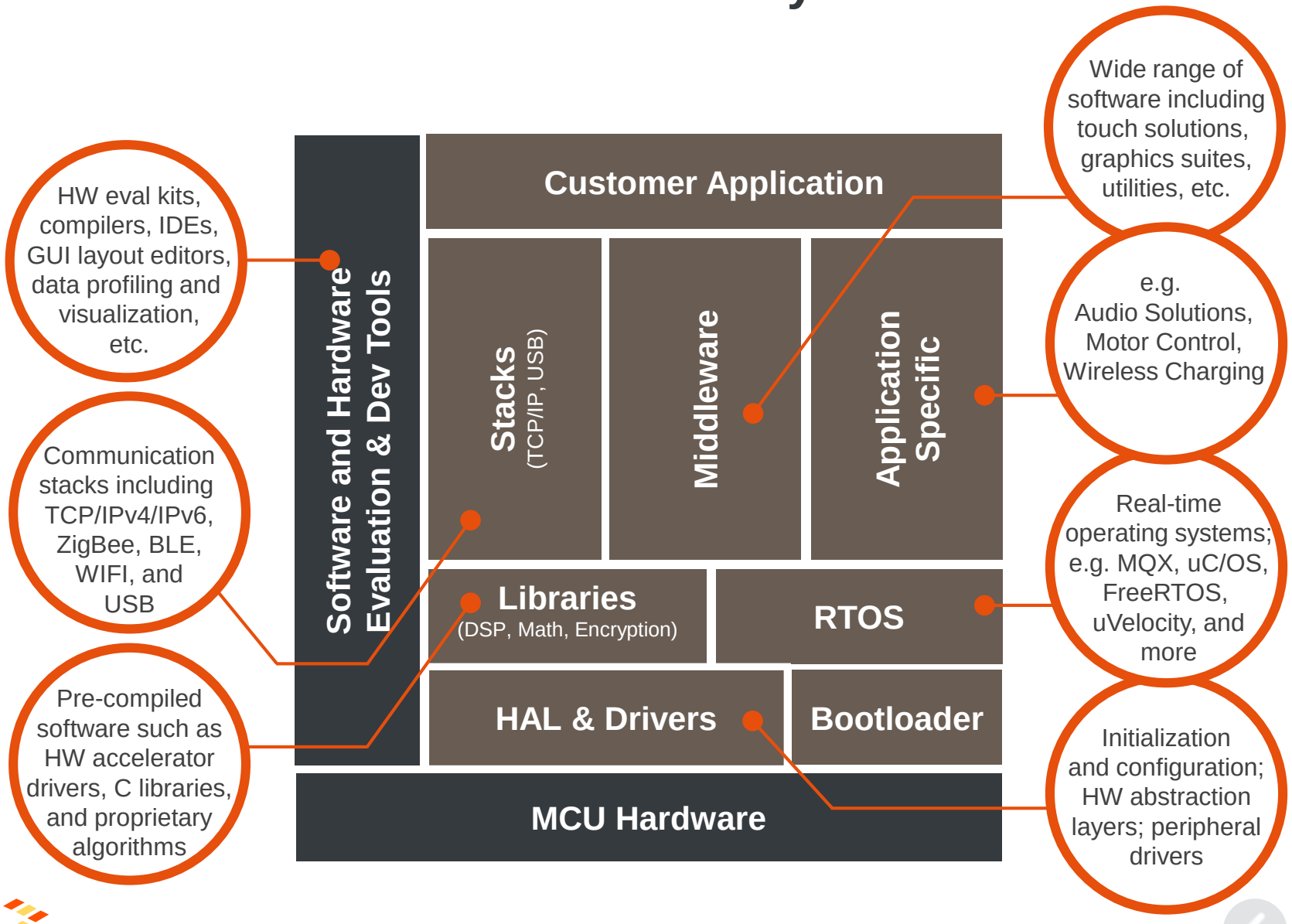
- Have a good grasp on what the Kinetis SDK is and how it will benefit you and your designs with Kinetis MCUs.
- Understand what the future holds for the Kinetis SDK
- Be able to develop bare-metal and RTOS-based applications using the 1.0-Beta release of the Kinetis SDK

Agenda

Introduction to the Kinetis Software Development Kit (SDK)

- Architectural Review
- Source Review
- Stack and Middleware Integration
- Configuration Using Processor Expert
- Roadmap
- Hands-On with the Kinetis SDK

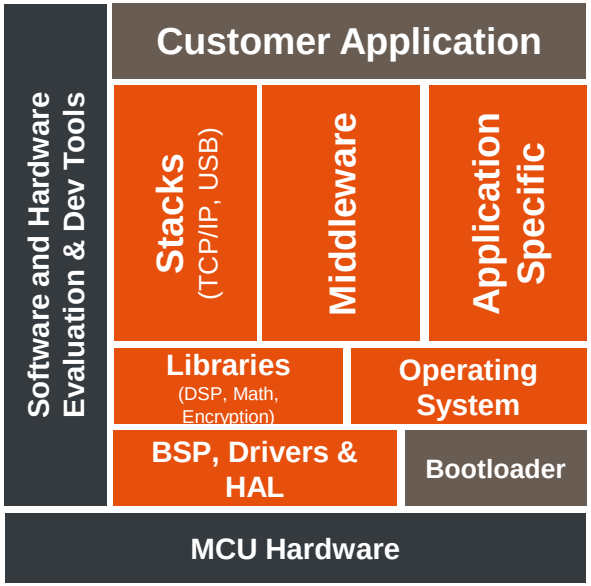
Microcontroller Software Taxonomy



Kinetis Software Development Kit (SDK)

A complete software framework for developing applications across all Kinetis MCUs

HAL, peripheral drivers, libraries, middleware, utilities, and usage examples; delivered in C source



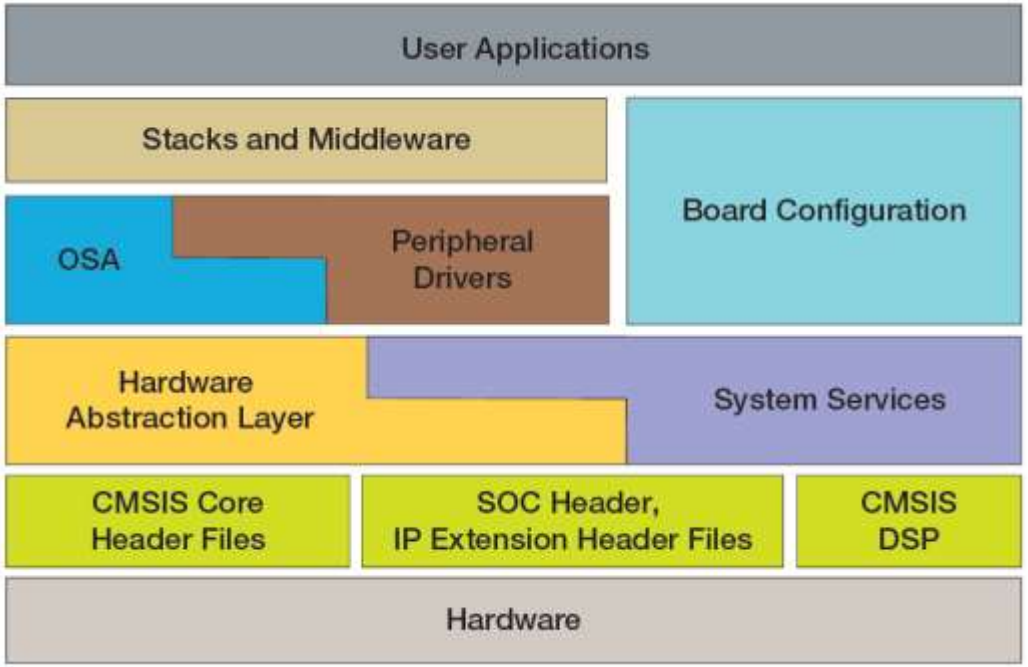
Open Source Initiative

CMSIS COMPLIANT

Product Features

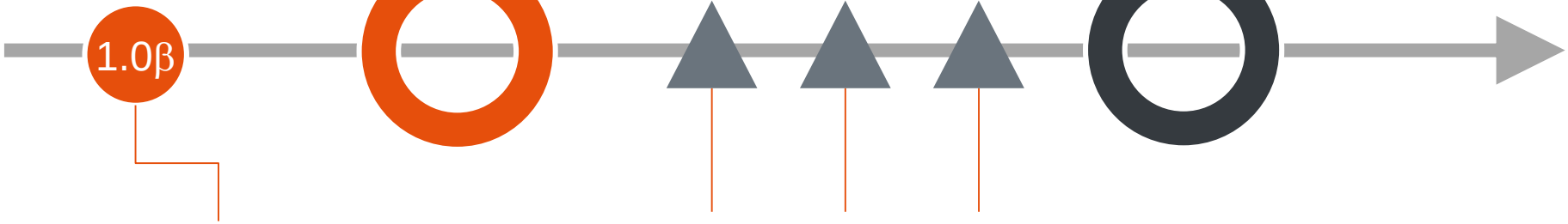
- Open source Hardware Abstraction Layer (HAL) provides APIs for all Kinetis hardware resources
- BSD-licensed set of peripheral drivers with easy-to-use C-language APIs
- Comprehensive HAL and driver usage examples and sample applications for RTOS and bare-metal.
- CMSIS-CORE compatible startup and drivers plus CMSIS-DSP library and examples
- RTOS Abstraction Layer (OSA) with support for Freescale MQX, FreeRTOS, Micrium uC/OS, bare-metal and more
- Integrates USB and TCP/IP stacks, touch sensing software, encryption and math/DSP libraries, and more
- Support for multiple toolchains including GNU GCC, IAR, Keil, and Kinetis Design Studio
- Integrated with Processor Expert

Kinetis Platform SDK Overview



- **Hardware Abstraction Layer**
  - Abstracted IP level Basic operations
  - Useable low level drivers
- **System Services**
  - Clock Manager, Interrupt manager, Low power manager, HW timer…
  - Can be used with HAL, PD and Application
- **FSL Peripheral Drivers**
  - Use case driven high level drivers
- **OS Abstraction Layer (OSA)**
  - Adapt to different OS (MQX, FreeRTOS and uC/OS) through corresponding OSA
- **BSP & Configuration**
  - Board Configuration, Pin Muxing, GPIO Configuration
  - Can be configured using Processor Expert
- **Stacks & Middle Wares**
  - USB stack, TCP/IP stack, Connectivity
  - Audio, Graphics, more...

Kinetis SDK 2014 Release Schedule



1.0 GA - Mid July 2014

- Finalized APIs for HAL and PD
- Add Keil, Atollic and KDS toolchains
- Add support for K22, KL03, KL43Z, more

1.1 GA – TBD Q4 2014

- Additional stack and middleware integration
- MQX RTCS, MFS integration

1.0-Beta - April 8, 2014

- Support for K64F, K63F, and K24F via TWR-K64F120M and FRDM-K64F (new)
- Supports IAR and GCC w/ make
- Training at FTF Americas
- Released to web

Stand-alone releases

- Based on 1.0 GA release
- Adds support for new Kinetis NPIs according to their launch schedule.

Cortex Microcontroller Software Interface Standard (CMSIS)

- Software layers for all ARM® Cortex®-M processor based devices
  - CMSIS-CORE: API for Cortex-M processor and core peripherals
  - CMSIS-DSP : DSP library with over 60 functions for Cortex-M
  - CMSIS-SVD : XML system view description for MCU peripherals
  - CMSIS-RTOS : API for RTOS integration
  - CMSIS-DAP : Standardize firmware for connecting to CoreSight DAP
  - CMSIS-Pack : XML based package description for file collections (packs)
  - CMSIS-Driver : defines generic peripheral driver interface for middleware

CMSIS-CORE Compliant Header Files with IP Extensions

- The Kinetis SDK does
  - use **CMSIS-CORE** API for peripheral access C macros, interrupt handler naming, etc. It also extends the peripheral headers to include
    - Easier access to registers
    - Use bit-banding where possible
  - provide the **CMSIS-DSP** lib and source (for GCC, built-into other tools) and usage example
- The Kinetis SDK does **not**
  - provide a **CMSIS-Driver** compatible layer—this may be considered in a future release

Peripheral IP Feature Header Files

- The Kinetis SDK uses “feature header files” to define IP specific features
- Helps abstract the drivers and provide common code base for variety of different IP or IP configurations

- Example:

fsl_uart_features.h

```
#if defined(CPU_MK10DN512VLK10) || defined(CPU_MK10DN512VLL10)...
    #define FSL_FEATURE_UART_HAS_LOW_POWER_UART_SUPPORT (0)
```

fsl_uart_hal_transfer_functions.c

```
void uart_hal_getchar(uint32_t uartInstance, uint8_t *readData)
{
#if FSL_FEATURE_UART_HAS_LOW_POWER_UART_SUPPORT
    …
#endif
    …
}
```

Hardware Abstraction Layer

- Licensed under BSD 3-clause open-source license
- Provides simple, stateless drivers with an API encapsulating the functions of Kinetis peripherals
- The layer closest to the hardware in our layered driver approach
- Designed to be run-time configurable by taking user defined configuration data through “init” function call

- Taking UART as an example:
  - HAL offers low-level init and byte transfer operations
    - Init, set baud rate, set parity, set stop bit, read byte, write byte…
  - HAL provides both blocking and non-blocking data transfers
    - non-blocking write byte used by interrupt driven Peripheral Driver
    - block write (polling) will make sure the byte can actually write to the data FIFO

Hardware Abstraction Layer

Example API for UART

- Byte write:
  - void uart_hal_putchar(uint32_t uartInstance, uint8_t data)
- Init:
  - uart_status_t uart_hal_init(uint32_t uartInstance, const uart_config_t *config)
- Baud configure:
  - uart_status_t uart_hal_set_baud_rate(uint32_t uartInstance, uint32_t sourceClockInHz, uint32_t desiredBaudRate)
- Status check:
  - bool uart_hal_is_receive_data_register_full(uint32_t uartInstance)

System Services

- Commonly used services
  - Hardware / unified timer can be running on any of the timers in SoC
  - Centralized Clock Manager
  - Centralized Interrupt Manager
  - Low Power Manager
- Can be built using SoC header files and HAL components
- Can be used by Peripheral Drivers or User Application
  - User can just use HAL and System Services to build applications
  - Alternatively, the Peripheral Drivers utilize the System Services and users do not need to use system services
- Can be used by OSA

Peripheral Drivers

- Open-source, high-level peripheral drivers
- Built on top of the HAL layer
- May utilize one or more HAL drivers and can take advantage of System Services
  - Shielded from the underlying hardware details by the HAL and System Services
- Drivers may be used as-is or as a reference for creating custom drivers
- Possibly combine one or multiple HALs and system services for a use case driven high level driver
- Peripheral Drivers are run-time configurable using configuration data structures passed into init()
- Examples:
  - UART: PD can be built on top of the HAL to deal with interrupt driven buffer level of operation
  - Composite drivers: ADC driven by using PDB

NOTE: we do not encourage mixing the usage of HAL and Peripheral Drivers. The application should either using HAL, or PD, but not both at the same time.

Example PD API for UART

- Init
  - uart_status_t uart_init(uint32_t uartInstance, uart_state_t * uartState,
    const uart_user_config_t * uartUserConfig);
- Send
  - uart_status_t uart_send_data(uart_state_t * uartState,
    const uint8_t * sendBuffer,
    uint32_t txByteCount, uint32_t timeout);
- Status
  - uart_status_t uart_get_transmit_status(uart_state_t * uartState,
    uint32_t * bytesTransmitted)

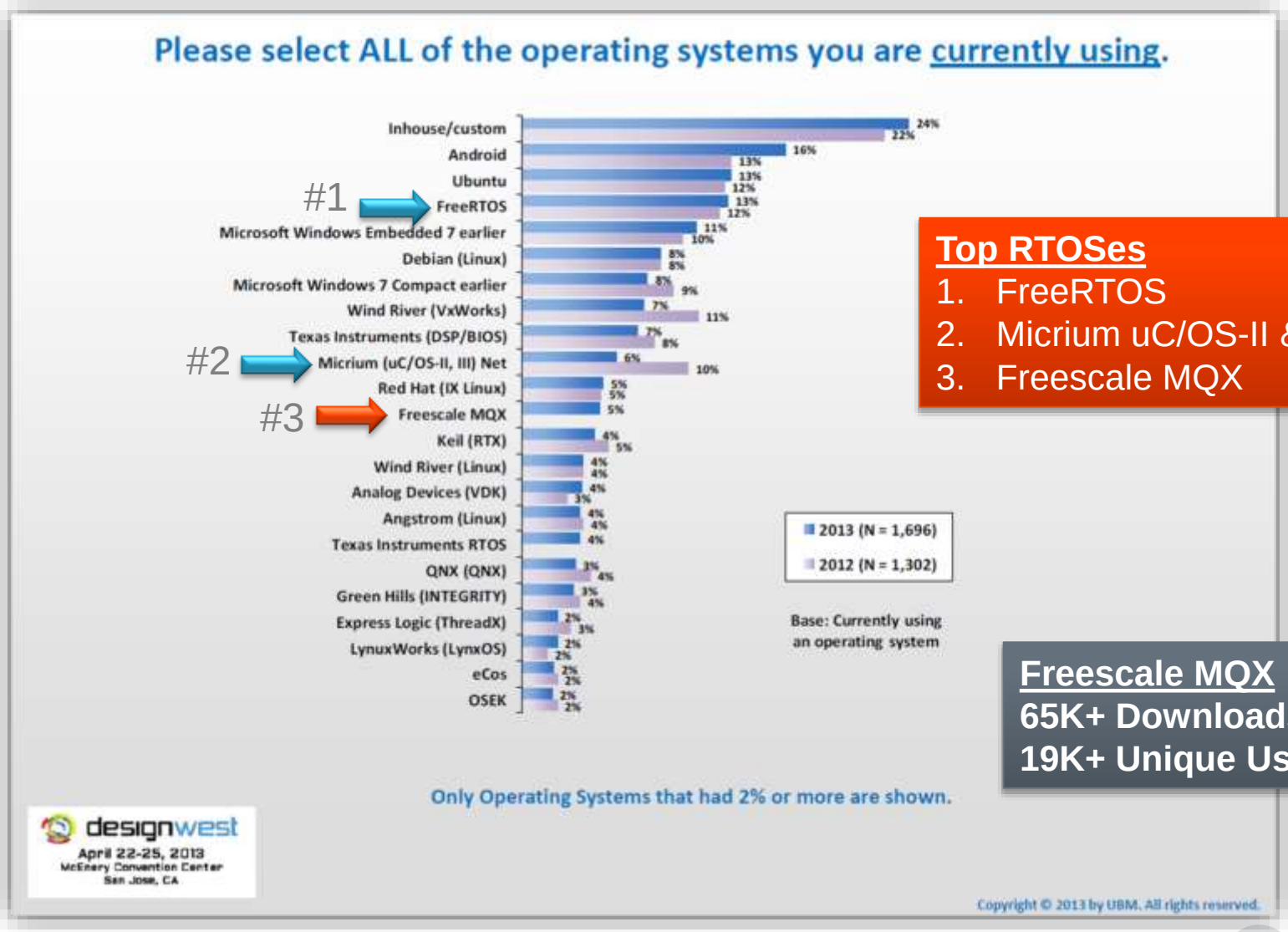
OSA: Real-Time Operating System Abstraction

- Kinetis SDK provides an operating system abstraction (OSA) layer for adapting applications for use with a real time operating system (RTOS) or bare metal (no RTOS) applications.
- OSAs are provided for:
  - Freescale MQX™ RTOS
  - FreeRTOS
  - Micrium uC/OS-II
  - Micrium uC/OS-III
  - bare-metal (no RTOS) RTOS abstraction layer bridges KSDK to work with or without RTOS
- Supports key RTOS services
  - Semaphores, Mutex, Memory Management, Event…

OSA: Real-Time Operating System Abstraction

- **Declarations**
  - Synchronization objects, event handler, queue handler, task handler
- **Task** (FreeRTOS Tasks)
  - **task_create()**, task_destroy(), **time_delay()**
- **Synchronization** (FreeRTOS counting semaphore)
  - sync_create(), sync_wait(), sync_poll(), sync_signal(), sync_signal_from_isr(), sync_destroy()
- **Locking** (FreeRTOS recursive Mutex)
  - lock_create(), lock_wait(), lock_poll(), lock_release(), lock_destroy()
- **Events** (FreeRTOS binary semaphore)
  - event_create(), event_wait(), event_set(), event_set_from_isr(), event_clear(), event_check_flags(), event_destroy()
- **Message Queue** (FreeRTOS Queues)
  - msg_queue_create(), msg_queue_put(), msg_queue_get(), msg_queue_flush(), msg_queue_destroy()
- **Memory**
  - mem_allocate(), mem_allocate_zero(), mem_free()

Top 3 MCU Real-Time Operating Systems



Top RTOSes

1. FreeRTOS
2. Micrium uC/OS-II & III
3. Freescale MQX

Freescale MQX
65K+ Downloads
19K+ Unique Users

Middleware and Stack Integration

- Middleware Integration
  - runs on top of the Kinetis SDK drivers
  - RTOS abstraction addresses the usage with or without RTOS
- Stacks and middleware in source or object formats including:
  - USB Stack: comprehensive device and host stack with extensive USB class support
  - CMSIS DSP: a suite of common signal processing functions
  - Crypto software utilizing the mmCAU hardware acceleration
- 1.0-GA release planning to support:
  - Freescale MQX™ Real-Time TCP/IP Communication Suite (RTCS)
  - Freescale MQX™ File System (MFS)
  - FatFs, a FAT file system for small embedded systems unit
- More to come…

Board Configuration and Support

- Pin Muxing
  - Kinetis SDK driver layer does not handle pin muxing
  - It is handled at the board configuration level
  - Pin muxing functions can be generated using Pin Muxing tool in PEx
- Board Specific configuration
  - GPIO configuration
  - Hardware Initialization code
  - Function to initialize serial console for debug purposes.
- Drivers for common devices included in our evaluation boards provided for building demo applications
  - ENET PHY
  - SPI flash
  - Accelerometer
  - Codec

Processor Expert Software

- A development system to create, configure, optimize, migrate, and deliver software and configuration details for Freescale silicon.

Initialization

CMSIS Headers
CMSIS startup code
Reset register values
Vector Table setup
Peripheral Initialization
Pin Muxing initialization

Device Driver Components

RTOS adaptive drivers
Low power capabilities
Configuration integrated
NOW – Kinetis Platform SDK Drivers supported

Processor Knowledge base

>1000 Processors Supported

Configuration

Reset configuration
DDR configure/validate
Pin Muxing
Device Tree Editor
Uboot configuration

API Factory

Script-based build server
CMSIS Header files
3rd Party Tools NPI support
Detailed Register files
Si Validation scripts
[Used by Common Register Repository initiative]

Kinetis SDK and Processor Expert

- Processor Expert is a complimentary PC-hosted software configuration tool (Eclipse plugin)
- Processor Expert (PEx) provides a time-saving option for software configuration through a graphical user interface (GUI)
- Board configuration and driver tuning tasks include:
  - Optional generation of low-level device initialization code for post-reset configuration
  - Pin Muxing tools to generate pin muxing functions
  - Components based on Kinetis SDK drivers
    - Users configure the SoC and Peripherals in a GUI
    - PEx creates the configuration data structures for driver config and init

Demo Applications and Tool Chain Support

- Kinetis SDK includes software examples demonstrating the usage of the HAL, Peripheral Drivers, supported RTOS, and integrated middle wares
  - The usage of HAL, System Services and Peripheral Drivers
  - Demos work with or without RTOSs
  - Demos to assemble a typical application or solution for specified vertical markets
- Tool chains supported:
  - IAR Embedded Workbench
  - GCC from ARM Embedded project with makefiles
  - *Atollic TrueSTUDIO
  - *ARM Keil MDK
  - *Kinetis Design Studio

Kinetis IDE Options

SDK 1.0-Beta SDK 1.0-GA

Featured IDEs:

Atollic TrueSTUDIO

- Professional ECLIPSE/GNU based IDE with a MISRA-C checker, code complexity analysis and source code review features.
- Advanced RTOS-aware debugger with ETM/ETB/SWV/ITM tracing, live variable watch view and fault analyzer. Dual-core and multi-processor debugging.
- Strong support for software engineering, workflow management, team collaboration and improved software quality.

Green Hills MULTI

- Complete & integrated software and hardware environment with advanced multicore debugger
- Industry first TimeMachine trace debugging & profiler
- EEMBC certified top performing C/C++ compilers

Keil Microcontroller Development Kit

- Specifically designed for microcontroller applications, easy to learn and use, yet powerful enough for the most demanding embedded applications
- ARM C/C++ build toolchain and Execution Profiler and Performance Analyzer enable highly optimized programs
- Complete Code Coverage information about your program's execution

IAR Embedded Workbench

- A powerful and reliable IDE designed for ease of use with outstanding compiler optimizations for size and speed
- The broadest Freescale ARM/Cortex MCU offering with dedicated versions available with functional safety certification
- Support for multi-core, low power debugging, trace, ...

Complimentary Solutions:

Kinetis Design Studio

- Complimentary basic capability integrated development environment (IDE) for Kinetis MCUs
- Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging

mbed Development Platforms

- The fastest way to get started with Kinetis MCUs
- Online project management and build tools – no installation required; option to export to traditional IDEs
- Includes comprehensive set of drivers, stacks and middleware with a large community of developers.

Additional Ecosystem Partners:

Kinetis SDK Directory Structure

- Demo applications
- Board specific code
- Integration guides, QSGs, RM, User's Guides
- Prebuilt libraries for each toolchain and board
- Common make files used for compiling with GCC
- Kinetis platform source
- CMSIS headers and DSP source/libs
- Peripheral Drivers
- Hardware Abstraction
- Linker files
- CMSIS startup
- Debug utilities, bare metal OSA
- OS abstraction layer code
- Freescale's new unified USB stack

Designing with Freescale

Tailored live, hands-on training in a city near you

2014 seminar topics include

- QorIQ product family update
- Kinetis K, L, E, V series MCU product training

freescale.com/DwF

www.Freescale.com