Developing with the **Kinetis Software Development Kit**

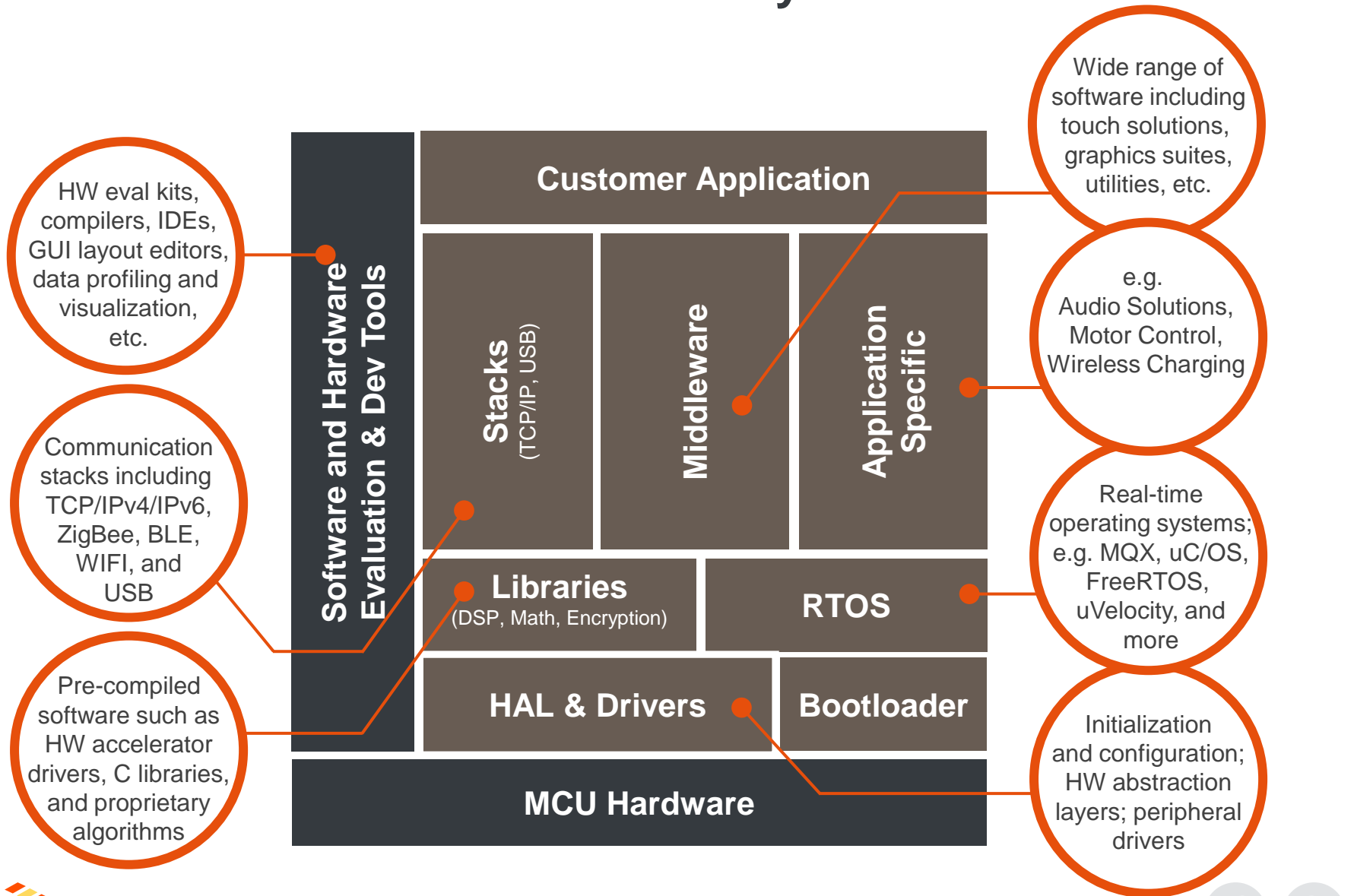Jerry Zeng | Software Manager

J A N . 2 0 1 5

freescale™

Agenda

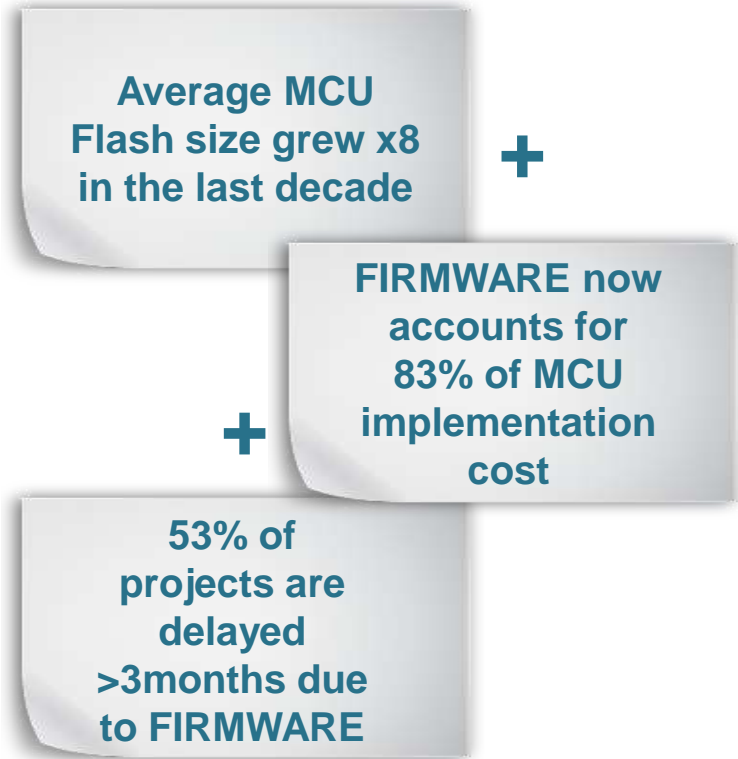Introduction to the Kinetis Software Development Kit (SDK)

- Oveview & Architecture

- Source review

- Stack and Middleware Integration

- Configuration Using Processor Expert

- Roadmap

- Release Content

- Kinetis SDK for MAPS Platform

Microcontroller Software Taxonomy

HW eval kits, compilers, IDEs, GUI layout editors, data profiling and visualization, etc.

Communication stacks including TCP/IPv4/IPv6, ZigBee, BLE, WIFI, and USB

Pre-compiled software such as HW accelerator drivers, C libraries, and proprietary algorithms

Software and Hardware Evaluation & Dev Tools

Customer Application

Stacks (TCP/IP, USB)

Middleware

Application Specific

Libraries (DSP, Math, Encryption)

RTOS

HAL & Drivers

Bootloader

MCU Hardware

Wide range of software including touch solutions, graphics suites, utilities, etc.

e.g. Audio Solutions, Motor Control, Wireless Charging

Real-time operating systems; e.g. MQX, uC/OS, FreeRTOS, uVelocity, and more

Initialization and configuration; HW abstraction layers; peripheral drivers

Growing Importance of Enablement

Average MCU Flash size grew x8 in the last decade

+

FIRMWARE now accounts for 83% of MCU implementation cost

+

53% of projects are delayed >3months due to FIRMWARE

Firmware is MCU developers **BIGGEST** pain point

Software and Hardware Evaluation & Dev Tools	Customer Application		
	Stacks (TCP/IP, USB)	Middleware	Application Specific
	Libraries (DSP, Math, Encryption)		RTOS
	HAL & Drivers		Bootloader

MCU Hardware

*freescale*™
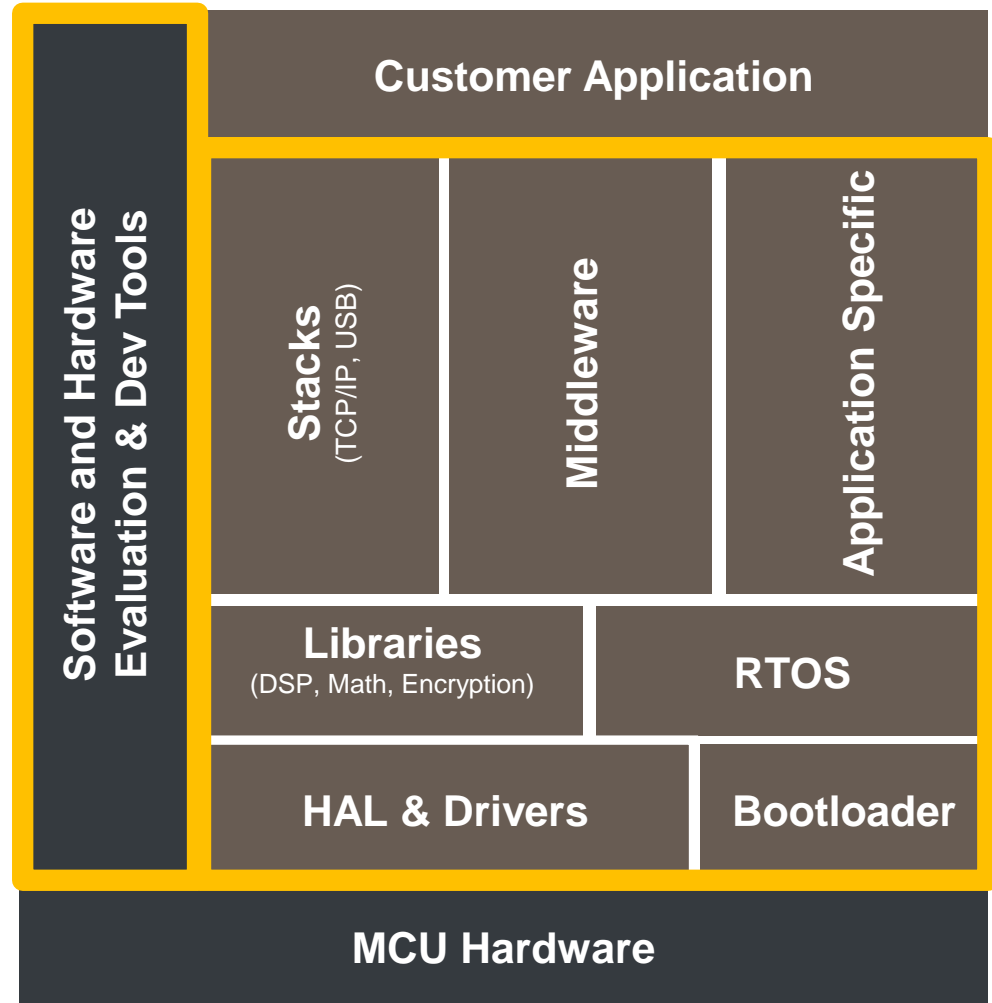
Kinetis Software Development Kit (SDK)

A complete software framework for developing applications across all Kinetis MCUs

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

Customer Application

Software and Hardware Evaluation & Dev Tools

Stacks (TCP/IP, USB)

Middleware

Application Specific

Libraries (DSP, Math, Encryption)

Operating System

BSP, Drivers & HAL

Bootloader

MCU Hardware

Open Source Initiative

CMSIS COMPLIANT ARM® Cortex™ Microcontroller Software Interface Standard

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

The OSI logo trademark is the trademark of Open Source Initiative.
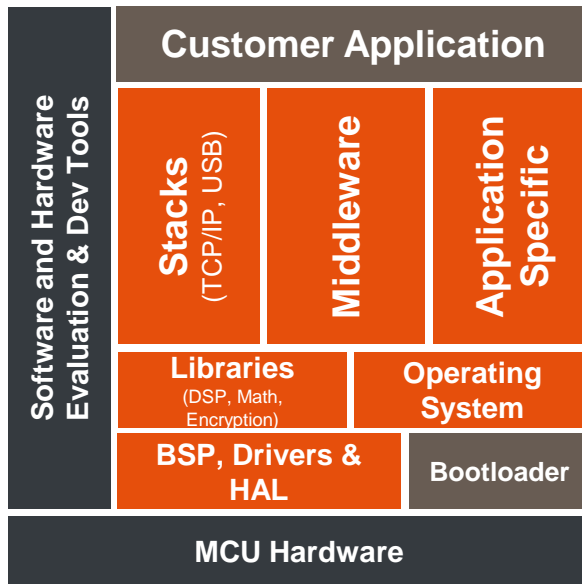
Kinetis Platform SDK Overview
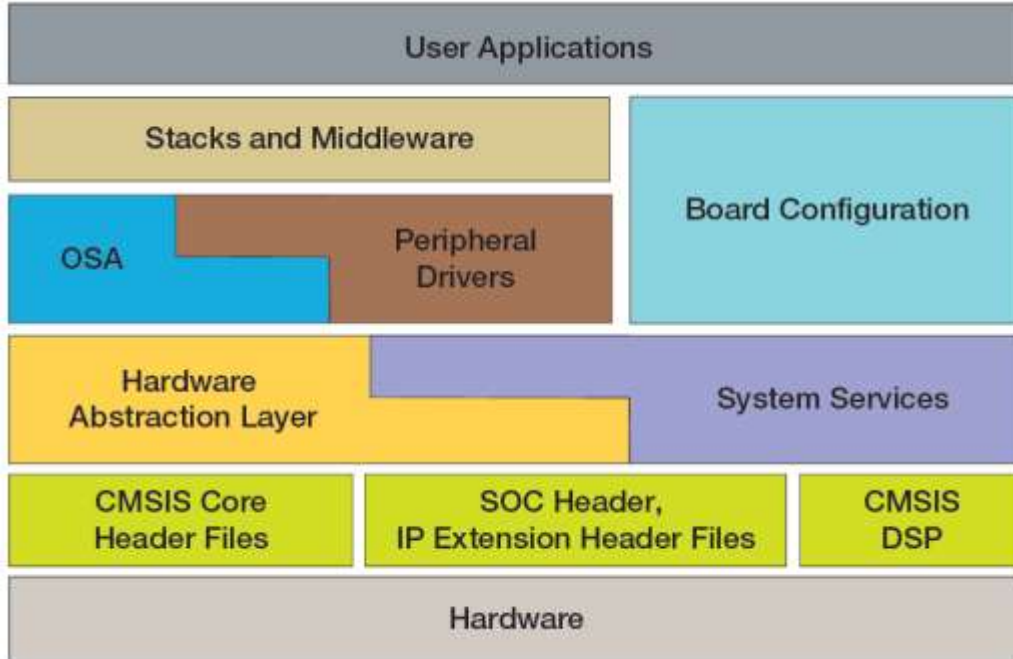
- **Hardware Abstraction Layer**
  - Abstracted IP level Basic operations
  - Useable low level drivers
- **System Services**
  - Clock Manager, Interrupt manager, Low power manager, HW timer…
  - Can be used with HAL, PD and Application
- **FSL Peripheral Drivers**
  - Use case driven high level drivers
- **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...

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-Driver : defines generic peripheral driver interface for middleware

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller
Software Interface Standard

freescale™

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
    …
}
```

freescale ™

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

- Init:
  - void UART_HAL_Init(uint32_t baseAddr)
- Enable Tx:
  - void UART_HAL_EnableTransmitter(uint32_t baseAddr)
- Baud configure:
  - uart_status_t UART_HAL_SetBaudRate(uint32_t baseAddr, uint32_t sourceClockInHz, uint32_t baudRate)
- Byte write:
  - void UART_HAL_Putchar(uint32_t baseAddr, uint8_t data)
- Status check:
  - bool UART_HAL_GetStatusFlag(uint32_t baseAddr, uart_status_flag_t statusFlag)

freescale ™

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

freescale ™

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.

Peripheral Drivers

Example PD API for UART

- Init
  - uart_status_t UART_DRV_Init(uint32_t instance,
    uart_state_t * uartStatePtr,
    const uart_user_config_t * uartUserConfig);

- Send
  - uart_status_t UART_DRV_SendData(uint32_t instance,
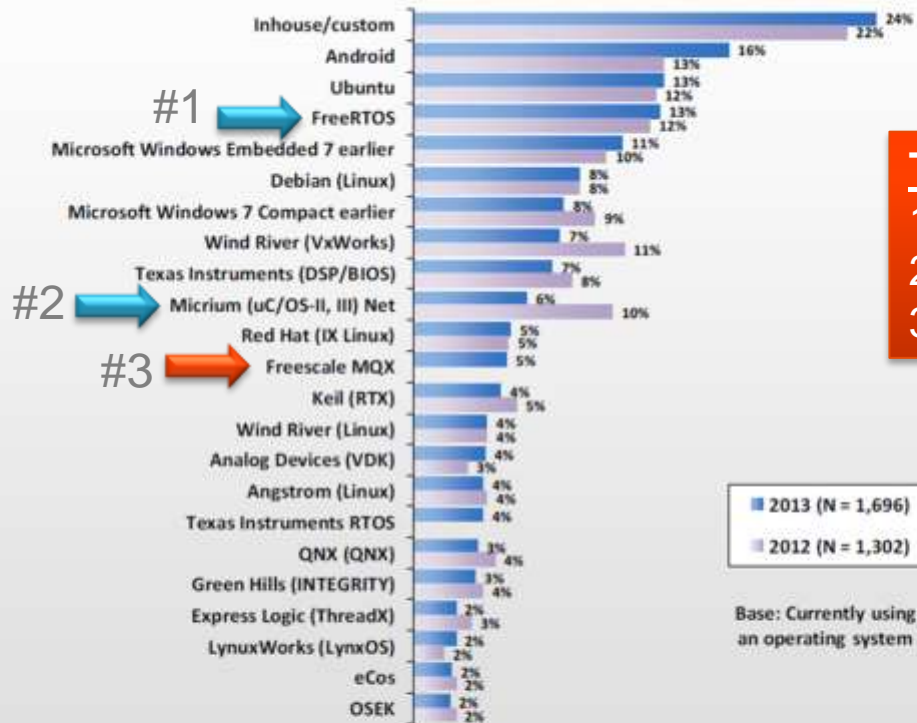   const uint8_t * txBuff, uint32_t txSize);

- Status
  - uart_status_t UART_DRV_GetTransmitStatus(uint32_t instance,
    uint32_t * bytesRemaining);

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, Task, Message ..

Top 3 MCU Real-Time Operating Systems

Please select ALL of the operating systems you are currently using.

Operating System	2013	2012
Inhouse/custom	24%	22%
Android	16%	
Ubuntu	13%	13%
FreeRTOS (#1)	12% / 13%	12%
Microsoft Windows Embedded 7 earlier	11%	10%
Debian (Linux)	8%	8%
Microsoft Windows 7 Compact earlier	8%	9%
Wind River (VxWorks)	7%	11%
Texas Instruments (DSP/BIOS)	7%	8%
Micrium (uC/OS-II, III) Net (#2)	6%	10%
Red Hat (IX Linux)	5%	5%
Freescale MQX (#3)	5%	
Keil (RTX)	4%	5%
Wind River (Linux)	4%	4%
Analog Devices (VDK)	4%	3%
Angstrom (Linux)	4%	4%
Texas Instruments RTOS	4%	
QNX (QNX)	3%	4%
Green Hills (INTEGRITY)	3%	4%
Express Logic (ThreadX)	2%	3%
LynuxWorks (LynxOS)	2%	2%
eCos	2%	2%
OSEK	2%	2%

2013 (N = 1,696)
2012 (N = 1,302)

Base: Currently using an operating system

Only Operating Systems that had 2% or more are shown.

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

Freescale MQX
65K+ Downloads
19K+ Unique Users

designwest
April 22-25, 2013
McEnery Convention Center
San Jose, CA

OSA: Real-Time Operating System Abstraction

- **Task**
  - OSA_TaskCreate(), OSA_TaskDestroy(), OSA_TimeDelay()
- **Semaphore**
  - OSA_SemaCreate(), OSA_SemaWait(), OSA_SemaPost(), OSA_SemaDestroy()
- **Mutex**
  - OSA_MutexCreate(), OSA_MutexLock(), OSA_MutexUnlock(), OSA_MutexDestroy()
- **Events**
  - OSA_EventCreate(), OSA_EventWait(), OSA_EventSet(), OSA_EventClear(), OSA_EventDestroy()
- **Message Queue**
  - OSA_MsgQCreate(), OSA_MsgQPut(), OSA_MsgQGet(), OSA_MsgQDestroy()
- **Memory**
  - OSA_MemAlloc(), OSA_MemFree()

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
  - FatFs, a FAT file system for small embedded systems unit
  - TCP/IP stack - lwIP
  - Freescale MQX™ Real-Time TCP/IP Communication Suite (RTCS)
  - Freescale MQX™ File System (MFS)

- More to come…
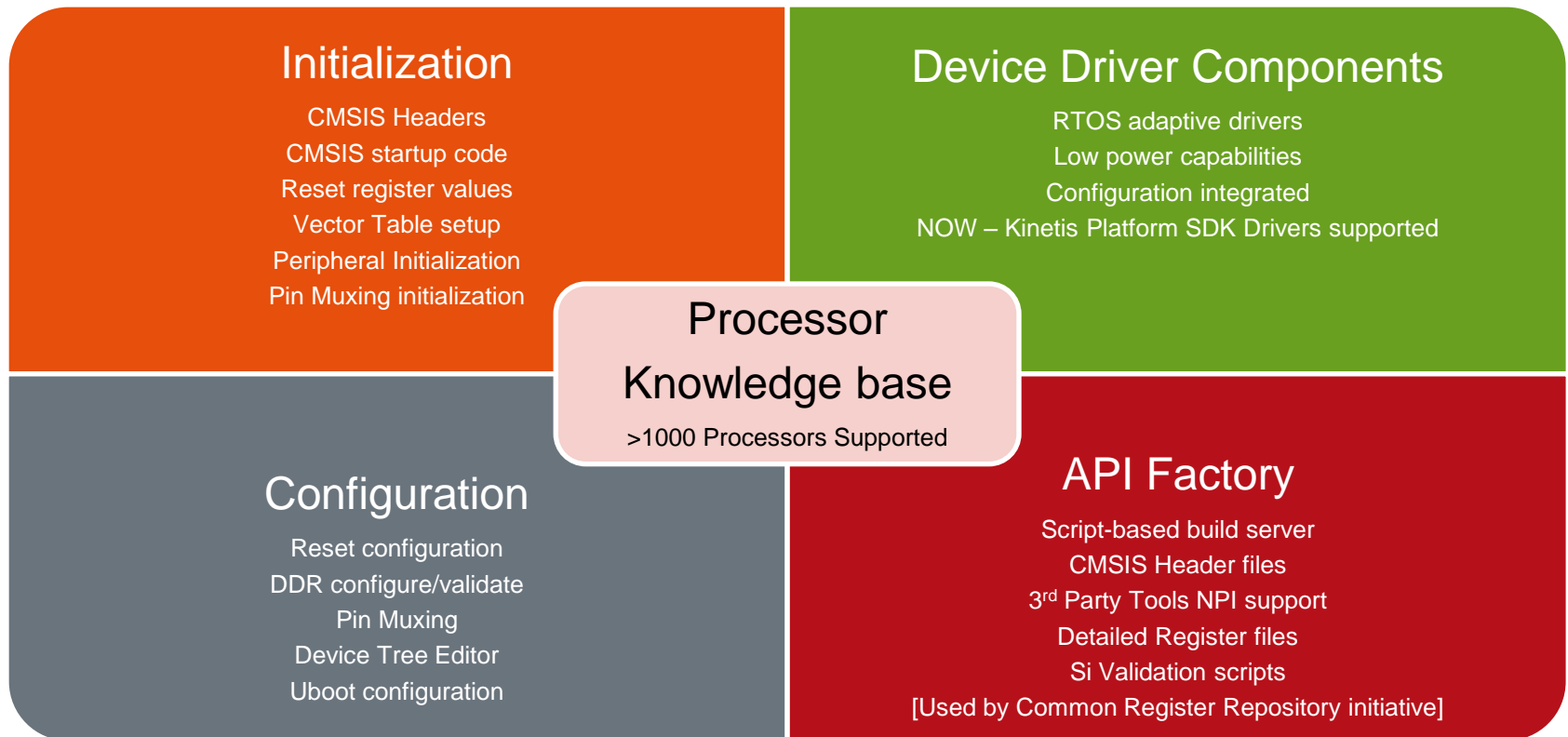
freescale ™

External Use | 17

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
  - Accelerometer
  - Codec

freescale ™

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
  - ARM Keil MDK
  - Kinetis Design Studio
  - Atollic TrueSTUDIO*

Kinetis IDE Options

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
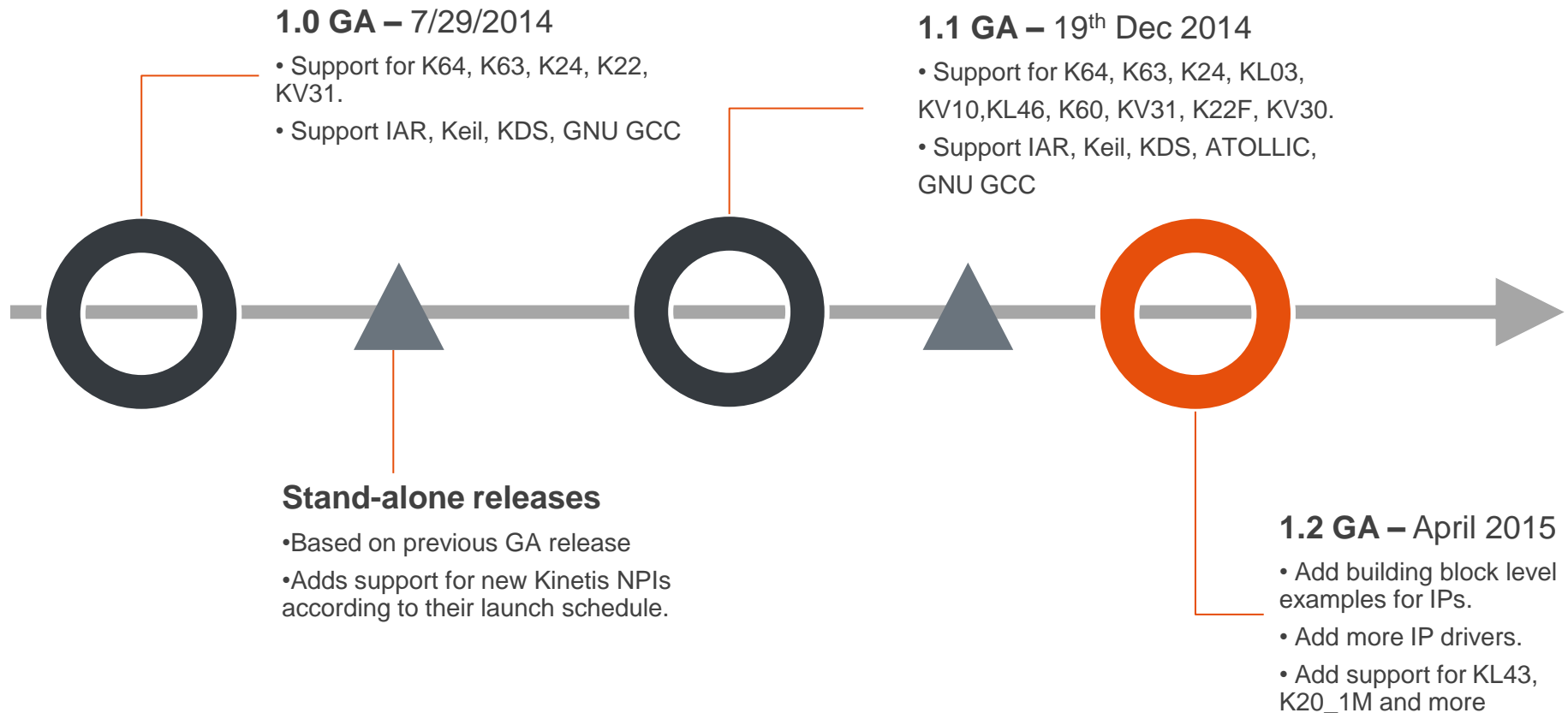
mbed Development Platforms

- 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 2014/2015 Release Schedule

1.0 GA – 7/29/2014

• Support for K64, K63, K24, K22, KV31.

• Support IAR, Keil, KDS, GNU GCC

1.1 GA – 19th Dec 2014

• Support for K64, K63, K24, KL03, KV10,KL46, K60, KV31, K22F, KV30.

• Support IAR, Keil, KDS, ATOLLIC, GNU GCC

Stand-alone releases

• Based on previous GA release

• Adds support for new Kinetis NPIs according to their launch schedule.

1.2 GA – April 2015

• Add building block level examples for IPs.

• Add more IP drivers.

• Add support for KL43, K20_1M and more

Kinetis SDK 1.0 Directory Structure

- KSDK_1.0.0
  - boards
  - demos
  - doc
  - filesystem
  - lib
  - mk
  - platform
    - CMSIS
    - drivers
    - hal
    - linker
    - osa
    - startup
    - system
    - utilities
  - rtos
  - tcpip
  - tools
  - usb

– Board configurations
– Demo applications
– Integration guides, RM, User's Guides
– File System middleware
– 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
– OS abstraction layer code
– CMSIS startup
– System services
– Debug utilities
– RTOS kernel
– TCP/IP stack
– Eclipse KSDK component update for KDS & MQX plugin for IDE
– USB stack

freescale ™

Kinetis SDK 1.1 Directory Structure

```
▲ 📁 KSDK_1.1.0
  ▷ 📁 boards
  ▷ 📁 demos
  ▷ 📁 doc
  ▷ 📁 filesystem
  ▷ 📁 lib
  ▲ 📁 platform
    ▷ 📁 CMSIS
    ▷ 📁 composite
    ▷ 📁 drivers
    ▷ 📁 hal
    ▷ 📁 linker
    ▷ 📁 osa
    ▷ 📁 startup
    ▷ 📁 system
    ▷ 📁 utilities
  ▷ 📁 rtos
  ▷ 📁 tcpip
  ▷ 📁 tools
  ▷ 📁 usb
  ▷ 📁 utilities
```

- Board specific files
- Demo applications and projects
- Integration guides, RM, User's Guide
- File System middleware
- Library projects for supported SoCs
- Kinetis Platform Source
- CMSIS header, extension header and DSP Lib/Source codes
- Composite Drivers
- Peripheral Drivers
- Hardware abstractions
- Linker files
- OSA files
- CMSIS startup codes
- System service
- Debug utilities
- RTOS source codes
- TCP/IP stack.
- Processor expert support and other tools
- Freescale new unified USB stack
- Shell and standard library

freescale™

Kinetis SDK for MAPS Platform

- Customized software based on Kinetis SDK to support MAPS Platform, specifically for China customers
- Based on Kinetis SDK 1.0 for now, will upgrade to future Kinetis SDK version periodically
- Difference comparing to Kinetis SDK
  - Clean code only for particular SoC
  - Chinese document for Release Notes and HW / SW User Guide
  - Only support IAR / Keil, No GCC toolchain support for now
  - No Processor Expert Service Pack
  - Small package size

- K64 platform supported, more to come…

Q&A

www.Freescale.com