HANDS-ON WORKSHOP:

USING MCUXpresso SDK

CLARK JARVIS

MCUXpresso SOFTWARE AND TOOLS

PRODUCT MARKETER

AMF-DES-T2633 | JUNE 2017

SECURE CONNECTIONS
FOR A SMARTER WORLD

AGENDA

- MCUXpresso Software And Tools Overview
- MCUXpresso SDK Introduction
  - Web Builder
  - File Structure
  - Examples
- MCUXpresso SDK Deep Dive
  - Drivers
  - Interrupts
  - Porting
- FreeRTOS
- Conclusion

MCUXpresso Software and Tools

for Kinetis and LPC microcontrollers

MCUXpresso IDE

Edit, compile, debug and optimize in an intuitive and powerful IDE

MCUXpresso SDK

Runtime software including peripheral drivers, middleware, RTOS, demos and more

MCUXpresso Config Tools

Online and desktop tool suite for system configuration and optimization

MCUXpresso Software and Tools

- Common toolkit across Kinetis and LPC microcontrollers
- Easy to use
- High quality
- Shared software experience and broader portfolio support
- Offers easy migration and scalability
- Supports large ARM® Cortex®-M ecosystem
- Built on the ‘best of’ Kinetis SDK, LPCXpresso and Kinetis Design Studio IDEs

MCUXpresso Software & Tools — Products

Integrated Development Environment (IDE)

- Offers edit, compile, debug, and many more tools with an intuitive and powerful interface
- Brings “best of” legacy IDEs (LPCXpresso and Kinetis® Design Studio) together, including GNU tool integration and library, multicore capable debugger, as well as trace functionality
- Debug connections that support all Freedom, Tower®, and LPCXpresso development boards plus industry leading commercial debug probes

Software Development Kit (SDK)

- The software framework and reference for application development with NXP’s MCUs based on ARM® Cortex®-M cores
- Includes production-grade software with integrated RTOS, integrated stacks and middleware, reference software, and more
- Highest quality with MISRA compliance on all drivers; checked with Coverity® static analysis tools
- Available in custom downloads based on user selections of MCU, evaluation board, and optional software components

System Configuration Tools

- Integrated configuration and development tools for Kinetis, LPC and i.MX products
- A suite of evaluation and configuration tools that helps guide users from first evaluation to production software development
- Includes SDK builder, power estimator, pins and clocks tools
- Available in online and desktop versions

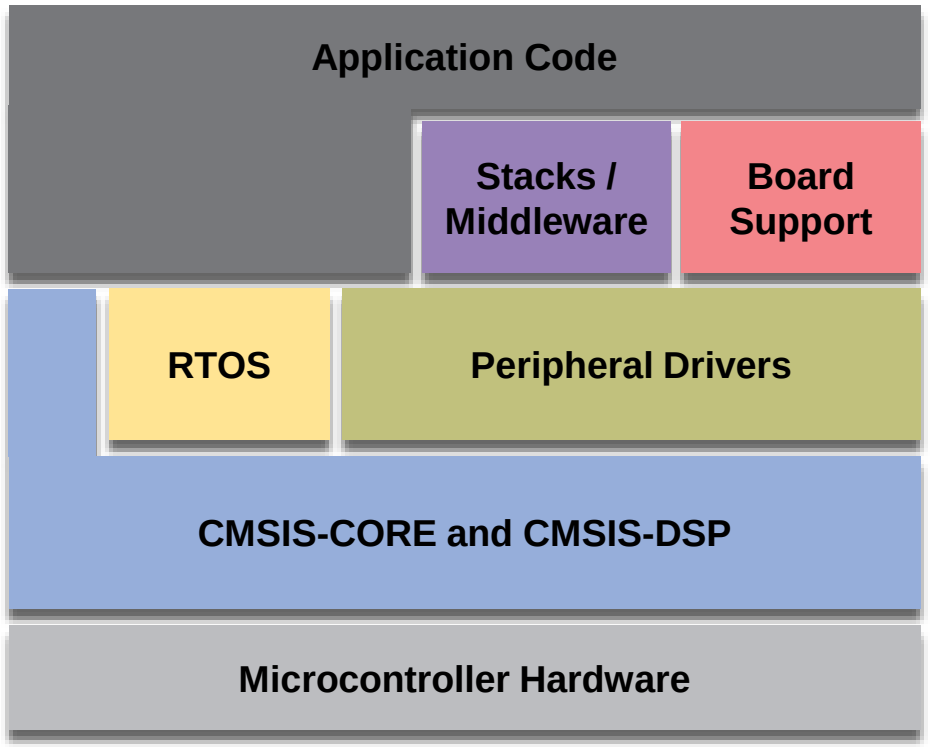
Origins of MCUXpresso Software & Tools

Kinetis and LPC SW Independent software and tools		**MCUXpresso Software and Tools** Supporting Kinetis & LPC Cortex-M MCUs
LPCXpresso IDE & Kinetis Design Studio	→	**MCUXpresso IDE**
Kinetis SDKv2	→	**MCUXpresso SDK**
Kinetis Expert / Processor Expert	→	**MCUXpresso Config Tools**

Learn more at: www.nxp.com/mcuxpresso/sdk

MCUXpresso SDK

The software framework and reference for Kinetis & LPC MCU application development



Product Features

Architecture:

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

Integrated RTOS:

- FreeRTOS v9
- RTOS-native driver wrappers

Integrated Stacks and 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:

- MCUXpresso IDE
- IAR®, ARM® Keil®, GCC w/ Cmake

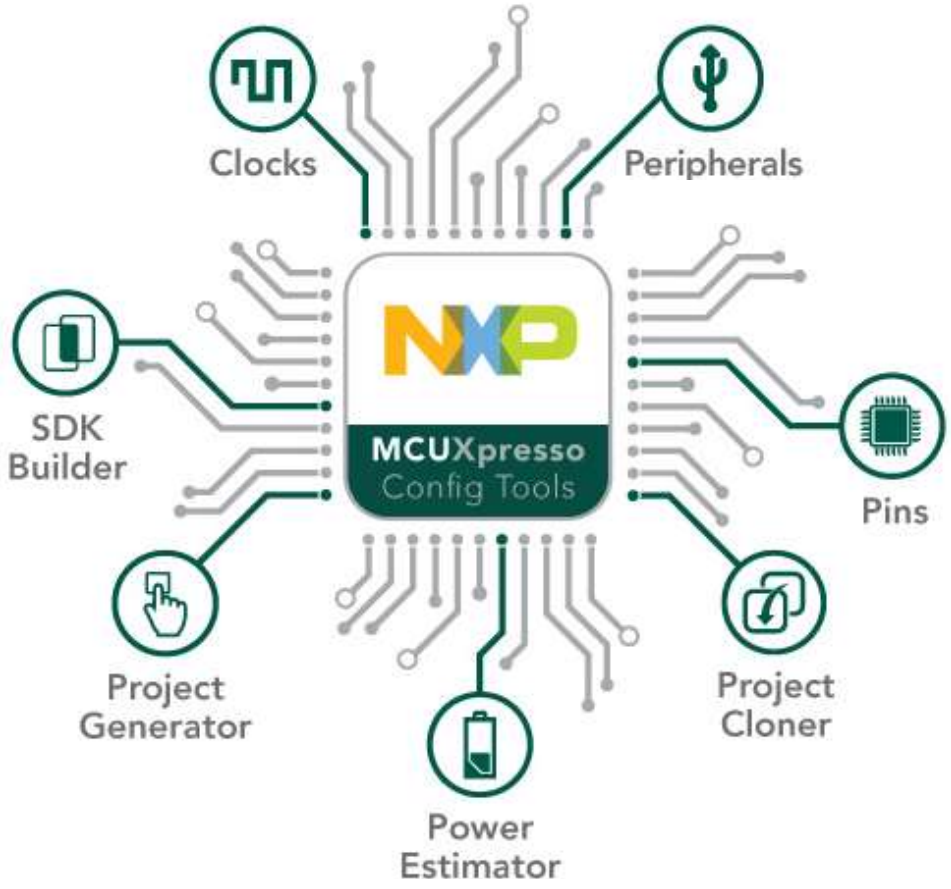
Quality

- Production-grade software
- MISRA 2004 compliance
- Checked with Coverity® static analysis tools

MCUXpresso Config Tools

Learn more at: www.nxp.com/mcuxpresso/config

Integrated configuration and development tools for LPC and Kinetis MCUs



MCUXpresso Config Tools is a suite of evaluation and configuration tools that helps guide users from first evaluation to production software development.

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

Pins, **Clocks**, and **Peripheral** tools generate initialization C code for custom board support. Features validation of inputs and cross-tool conflict resolution.

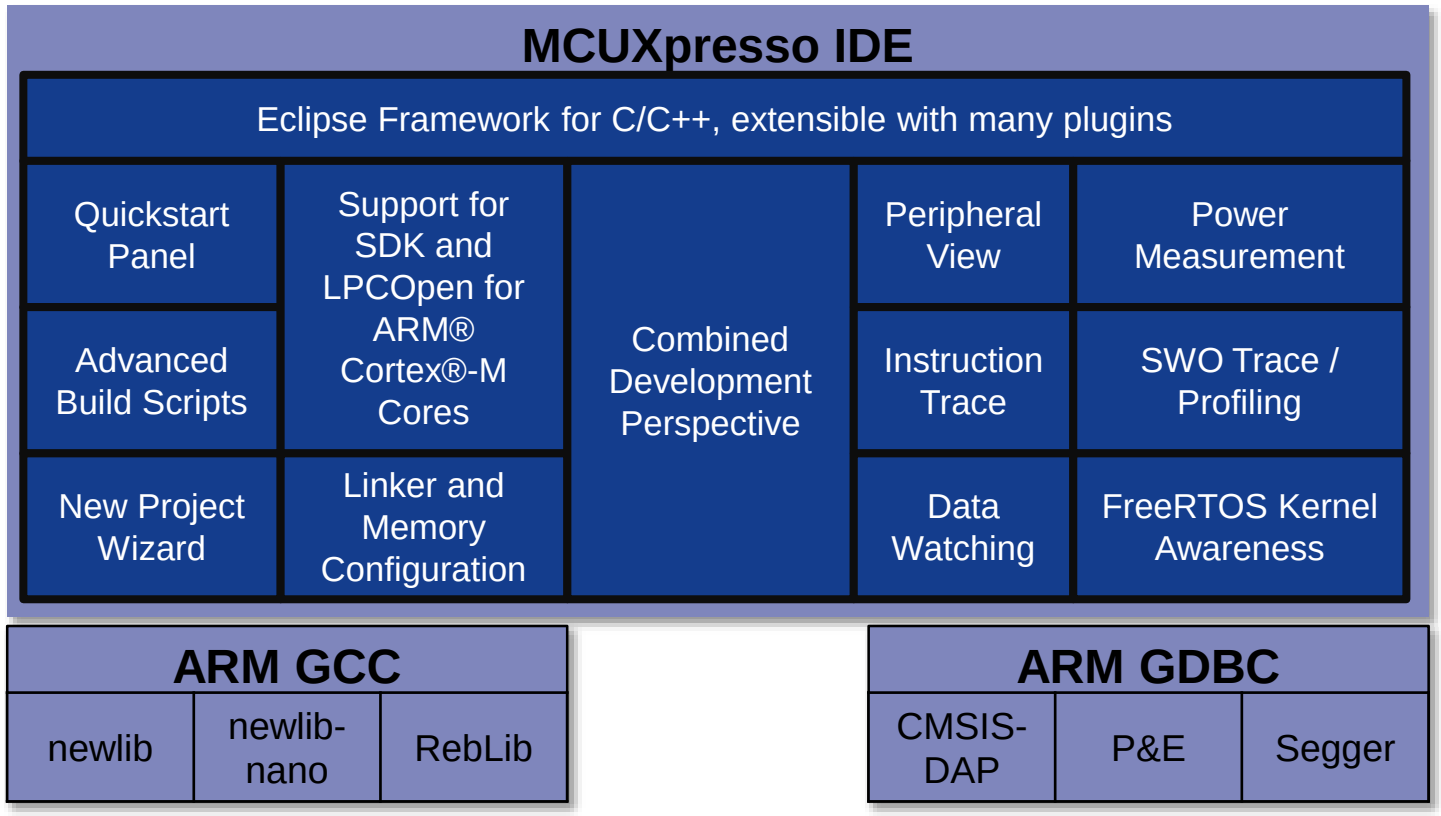
Project Generator creates new SDK projects with generated Pins and Clocks source files.

Project Cloning creates a standalone SDK project based on a example application available within SDK release.

Power Estimation tool provides energy and battery-life estimates based on a user's application model.
Available as a standalone tool for select devices.

MCUXpresso IDE

Free Eclipse and GCC-based IDE for C/C++ development on Kinetis and LPC MCUs



Product Features

- **Feature-rich**, **unlimited code size**, optimized for **ease-of-use**, based on **industry standard Eclipse** framework for **NXP's Kinetis** and **LPC MCUs**
- **Application development** with Eclipse and GCC-based IDE for advanced **editing**, **compiling** and **debugging**
- Supports **custom** development boards, **Freedom**, **Tower** and **LPCXpresso** boards with debug probes from **NXP**, **P&E** and **Segger**
- **Free Edition**: Full Featured, **unlimited Code Size**, no special activation needed, community based support
- **Pro Edition:** Email IDE support, **Advanced Trace Features**

MCUXpresso Software and Tools Additional Resources

Web pages

- MCUXpresso Software and Tools – www.nxp.com/mcuxpresso
  - MCUXpresso SDK – www.nxp.com/mcuxpresso/sdk
  - MCUXpresso IDE – www.nxp.com/mcuxpresso/ide
  - MCUXpresso Config Tools – www.nxp.com/mcuxpresso/config

Supported Devices

- Supported Devices Table (Community Doc)

Communities

- MCUXpresso Software and Tools – https://community.nxp.com/community/mcuxpresso
  - MCUXpresso SDK: https://community.nxp.com/community/mcuxpresso/mcuxpresso-sdk
  - MCUXpresso IDE: https://community.nxp.com/community/mcuxpresso/mcuxpresso-ide
  - MCUXpresso Config Tools: https://community.nxp.com/community/mcuxpresso/mcuxpresso-config

AGENDA

- MCUXpresso Software And Tools Overview
- MCUXpresso SDK Introduction
  - Web Builder
  - File Structure
  - Examples
- MCUXpresso SDK Deep Dive
  - Drivers
  - Interrupts
  - Porting
- FreeRTOS
- Conclusion

MCUXPRESSO SDK

MCUXpresso Homepage

https://mcuxpresso.nxp.com/en/welcome

Configuration

- What is a configuration?
  - A group of configured settings used across the MCUXpresso configuration tools (SDK builder, Pins, and Clocks)

- What is included in a configuration?
  - SDK builder configuration settings (e.g. Board/Processor, Toolchain, Host OS, etc.)
  - Pin assignments in the Pins Tool
  - Clock initializations in the Clocks Tool

- Configurations can be saved and shared as a .mex file

Get Started

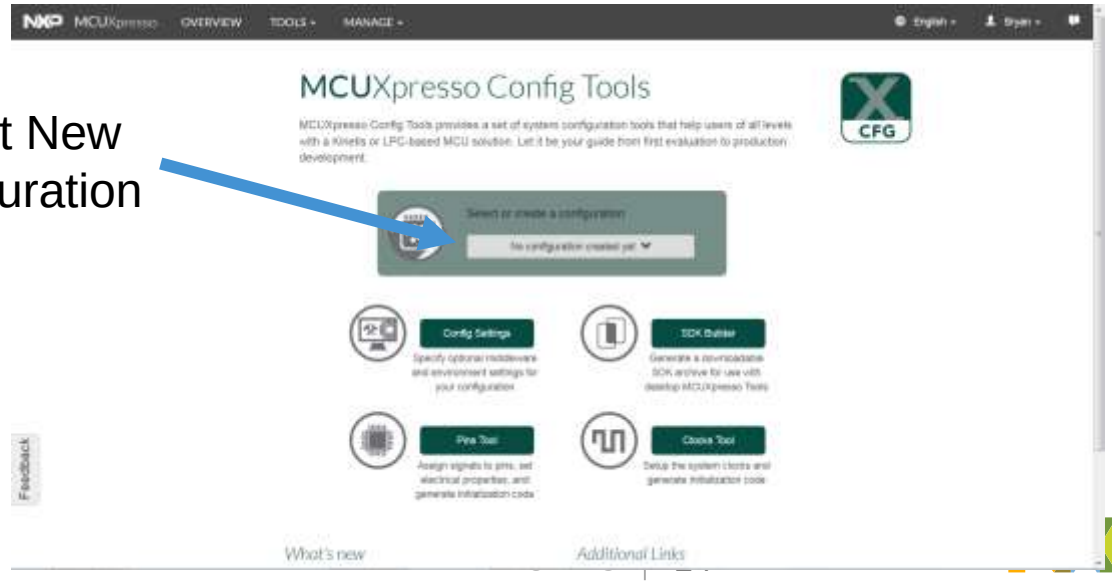
1. Login

Routed to nxp.com

2. Enter account info

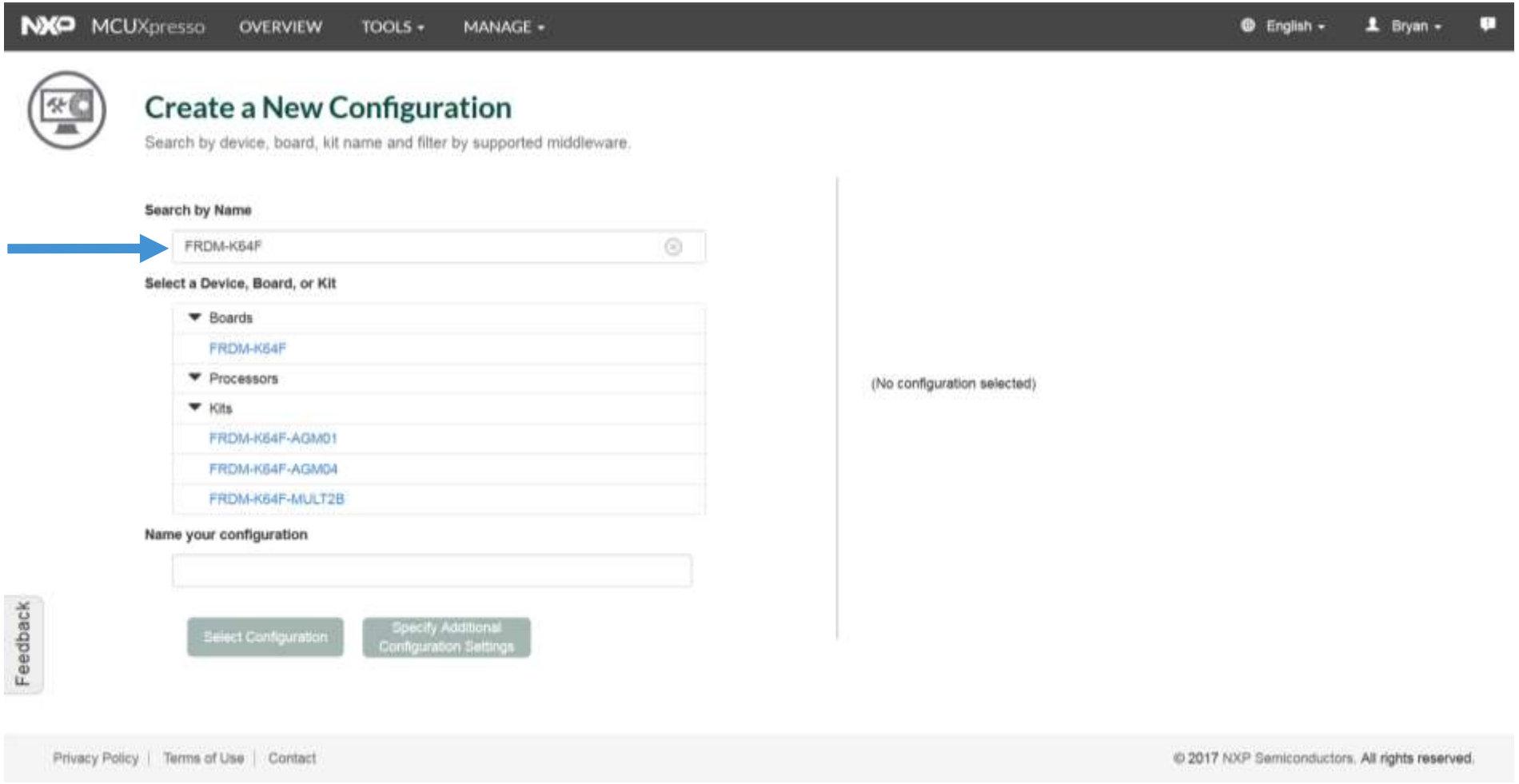
Return to mcuxpresso.nxp.com

3. Start New Configuration



Create a New Configuration (1/3)

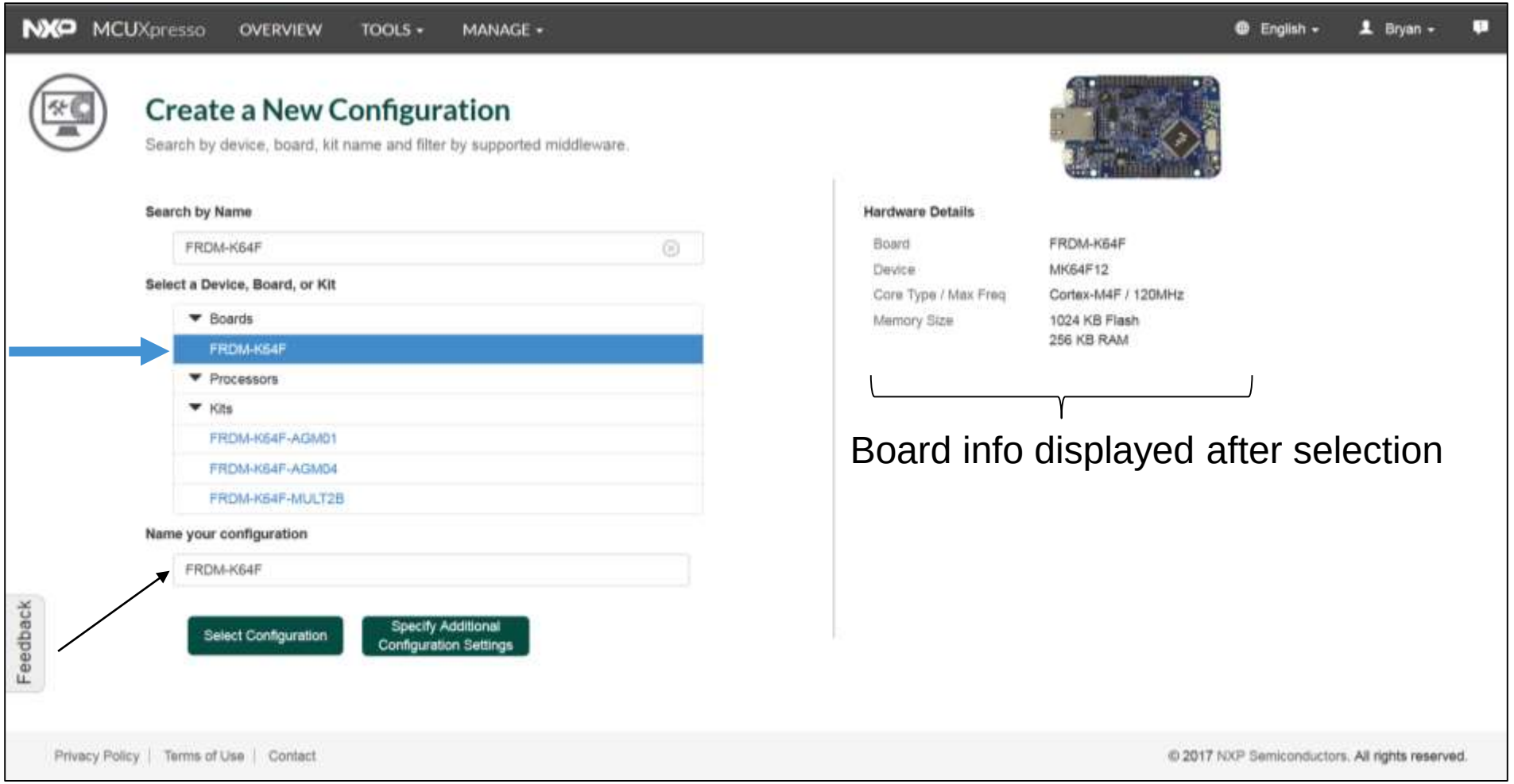
1. Type in search



Create a New Configuration (2/3)

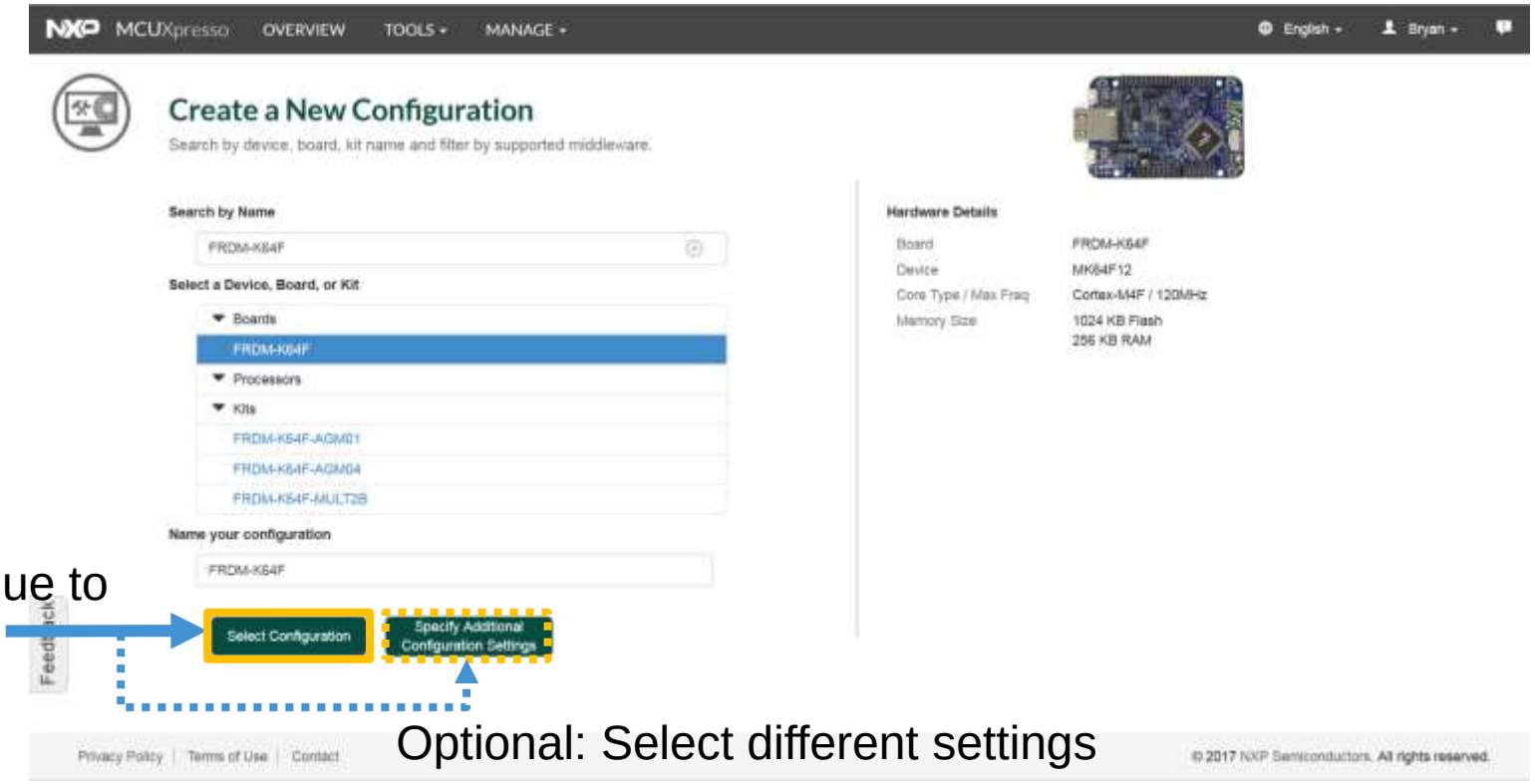
2. Make selection

Configuration name automatically assigned. Name can be modified



Board info displayed after selection

Create a New Configuration (3/3)



3. Continue to builder

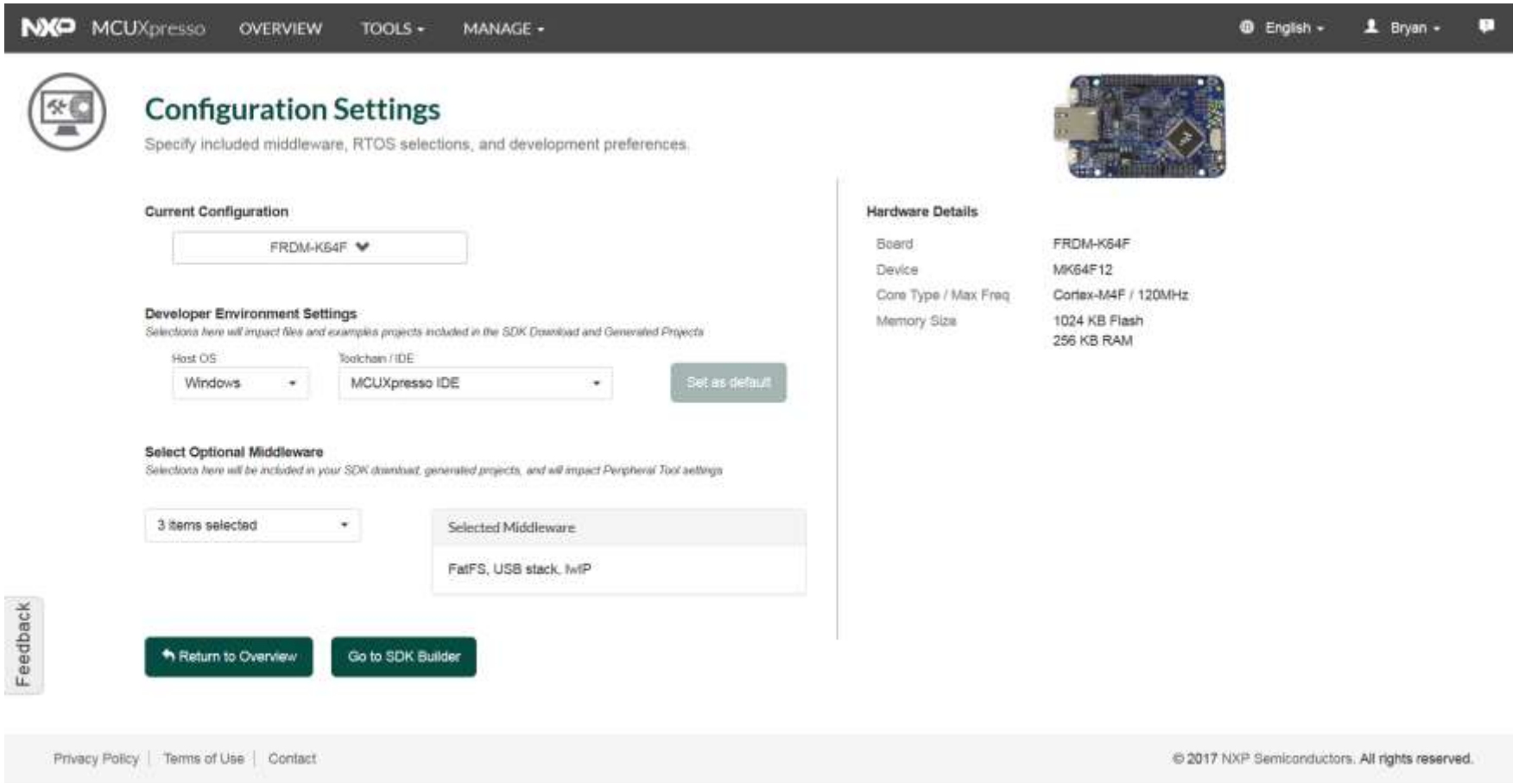
Optional: Select different settings

- Select Configuration
  - Proceed to builder with default options selected for toolchain, OS, and middleware

- Specify Additional Configuration Settings
  - Select toolchain, OS, and middleware other than default.

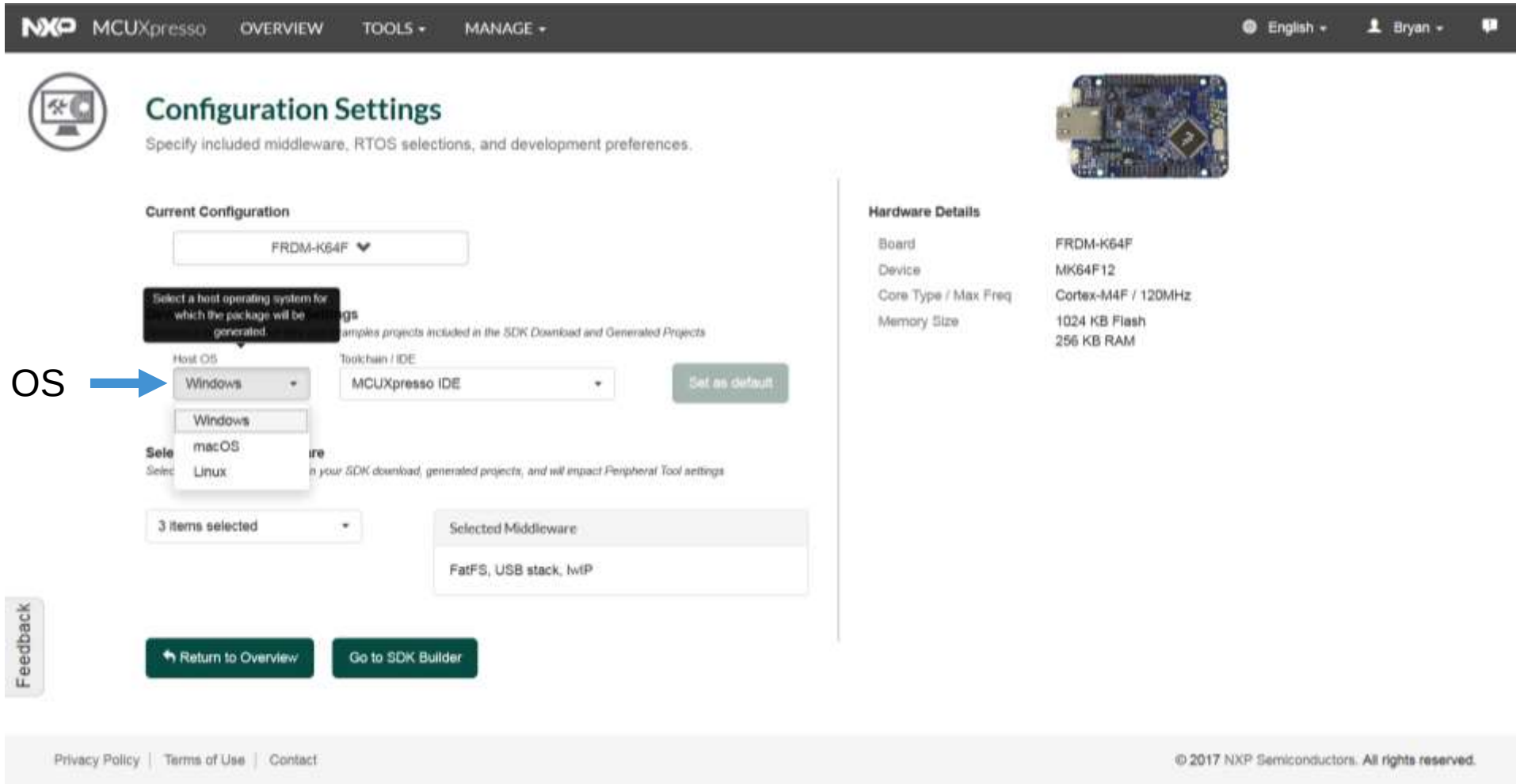
Additional Configuration Settings

- User selects:
  - Host OS
  - Toolchain/IDE
  - Middleware
- Defaults (first session):
  - Host OS -> **Windows**
  - IDE -> **MCUXpresso**
  - Middleware -> **FatFS**, **USB Stack***, **lwIP***

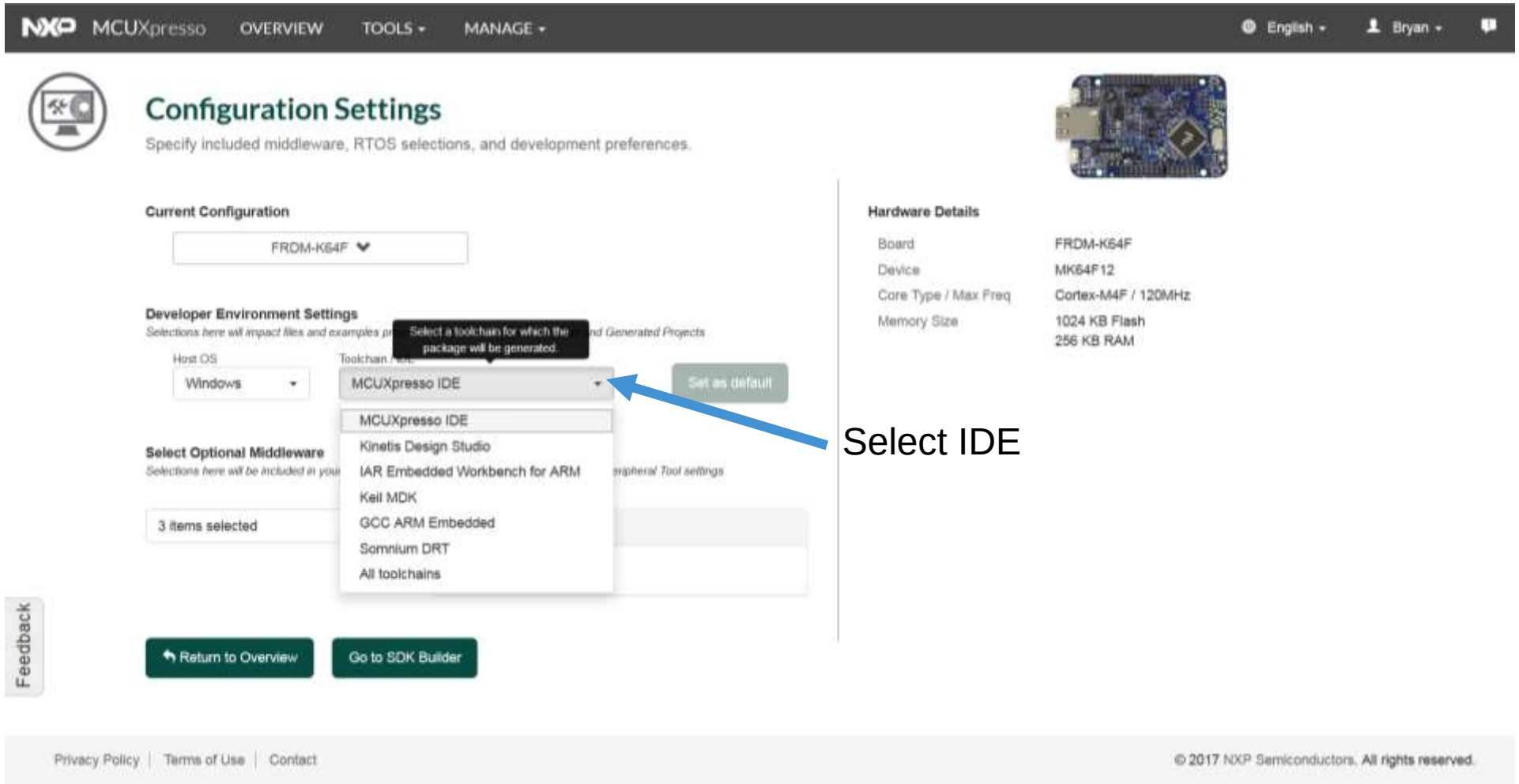


Additional Configuration Settings: Choose an OS

Select Host OS



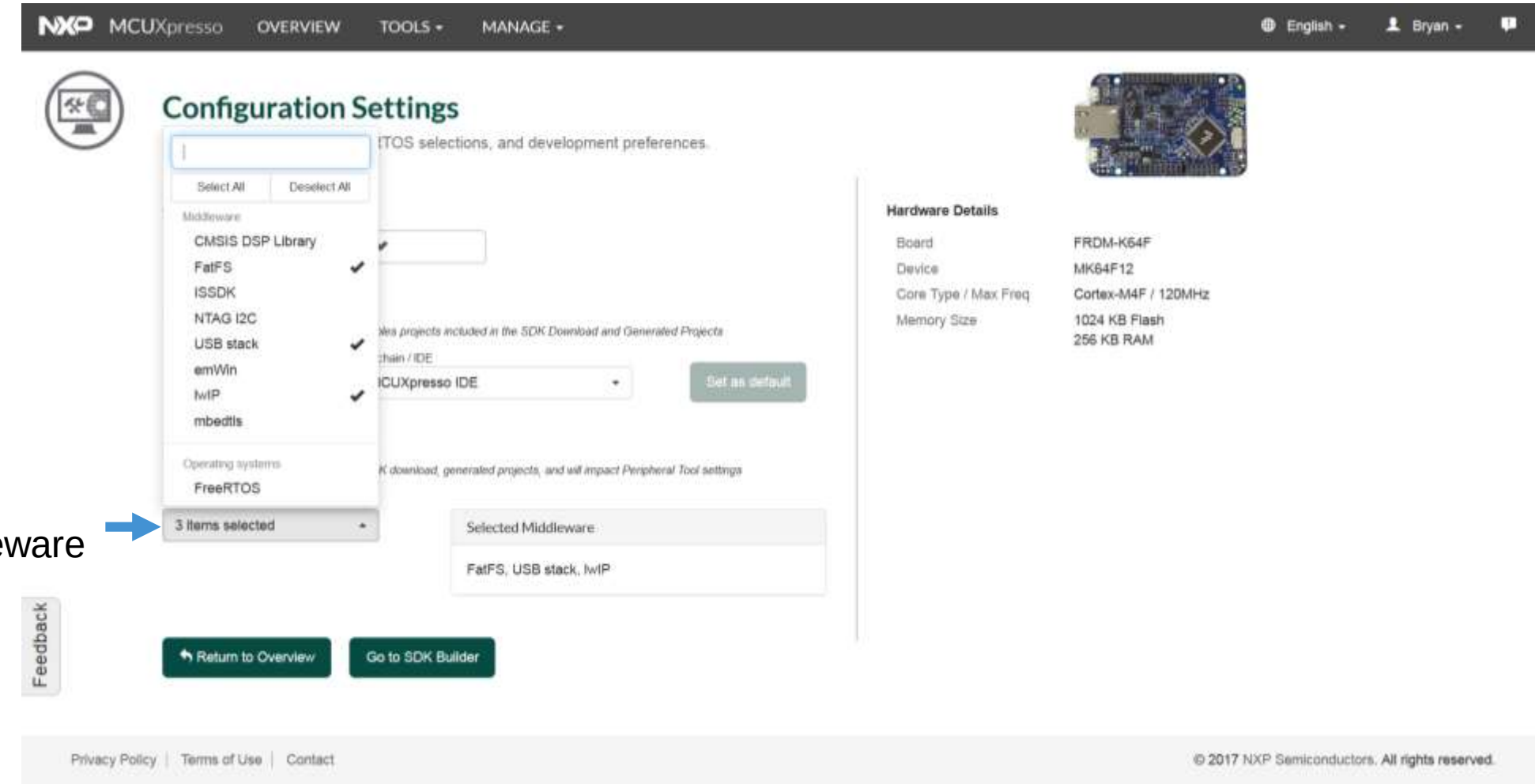
Additional Configuration Settings: Choose an IDE



Select IDE

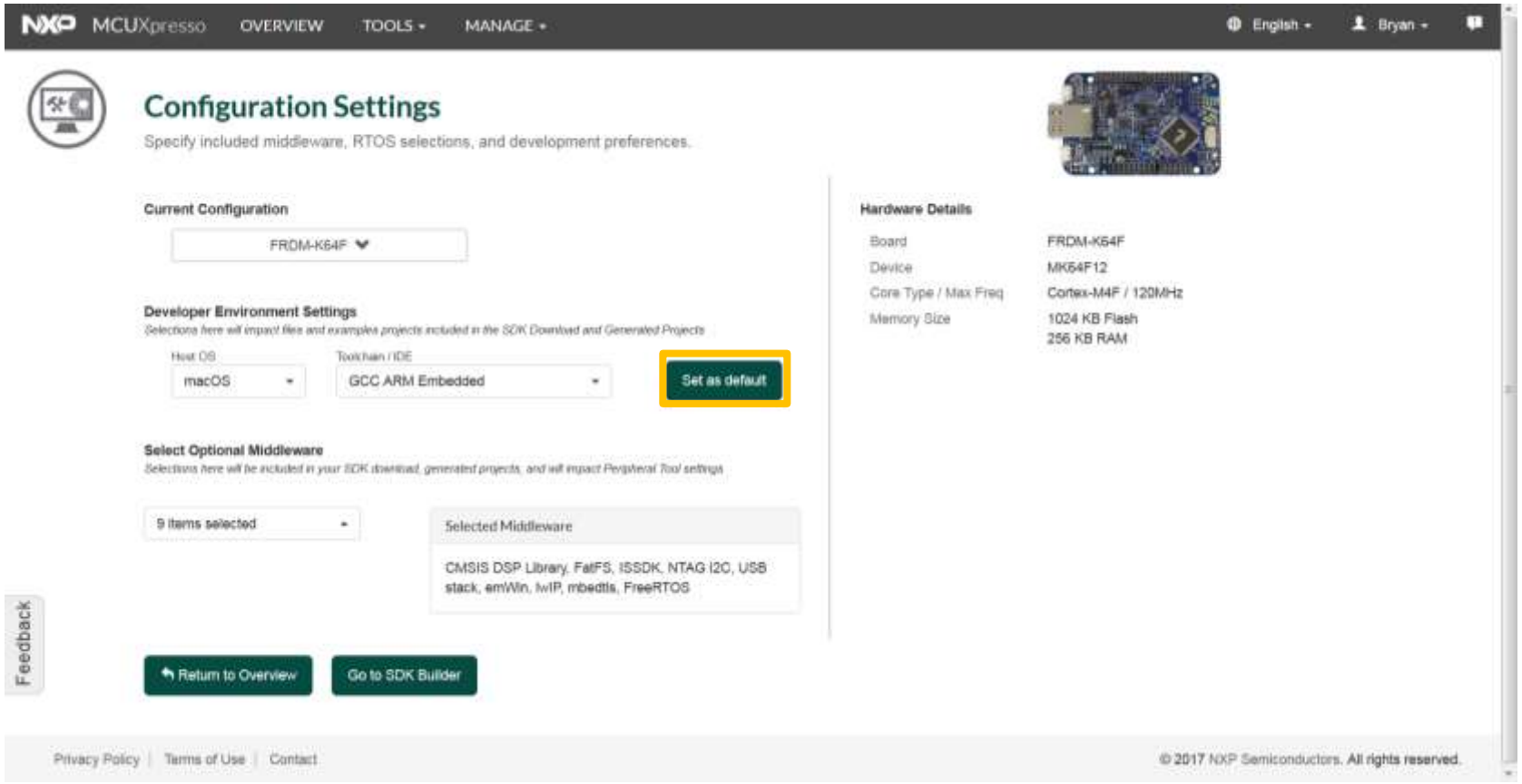
Additional Configuration Settings: Choose Middleware/RTOS

Select Middleware

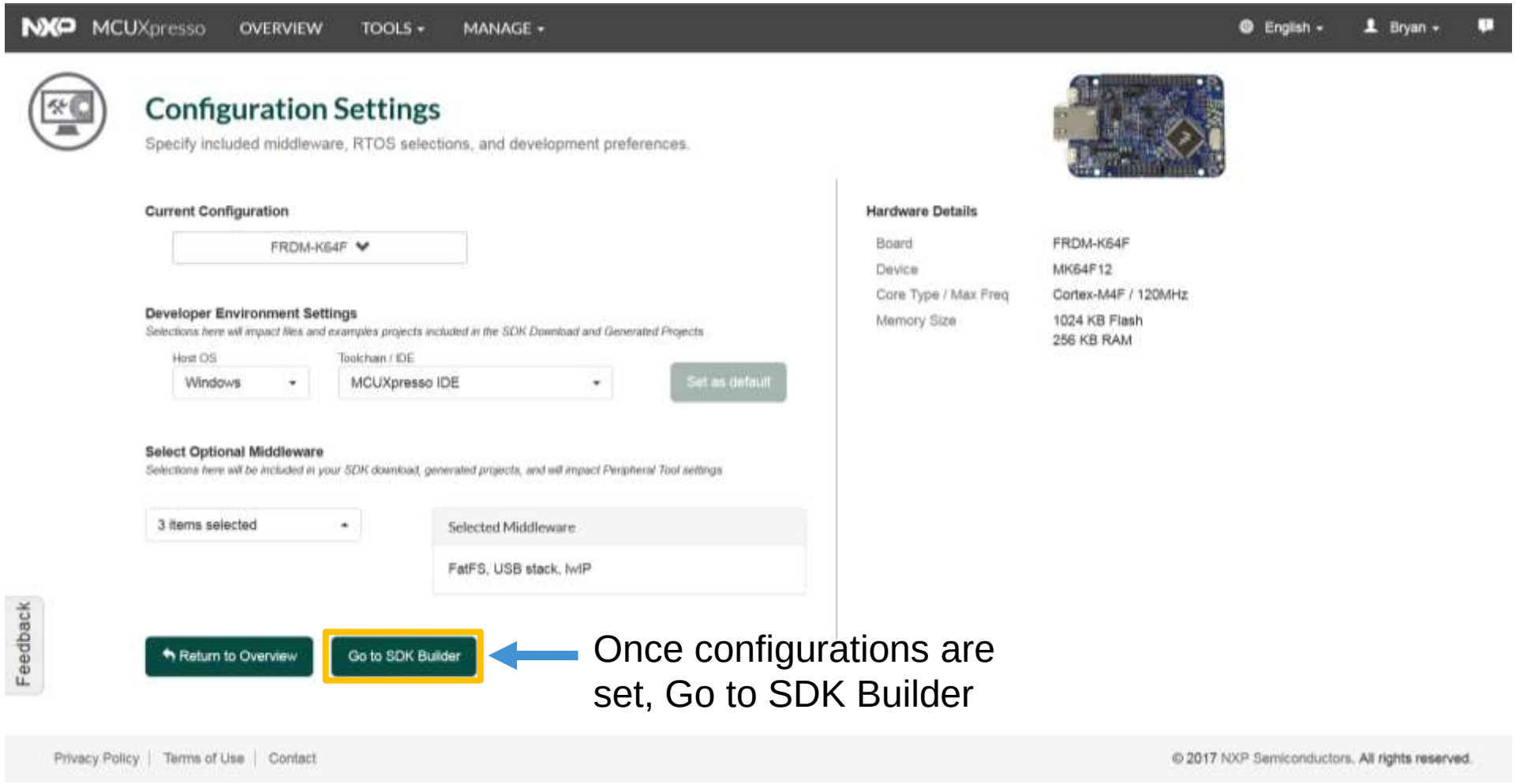


Change Default Build Settings

- Select “Set as Default” to save Host OS and Toolchain to preferences
- Future configurations will use these build settings as defaults

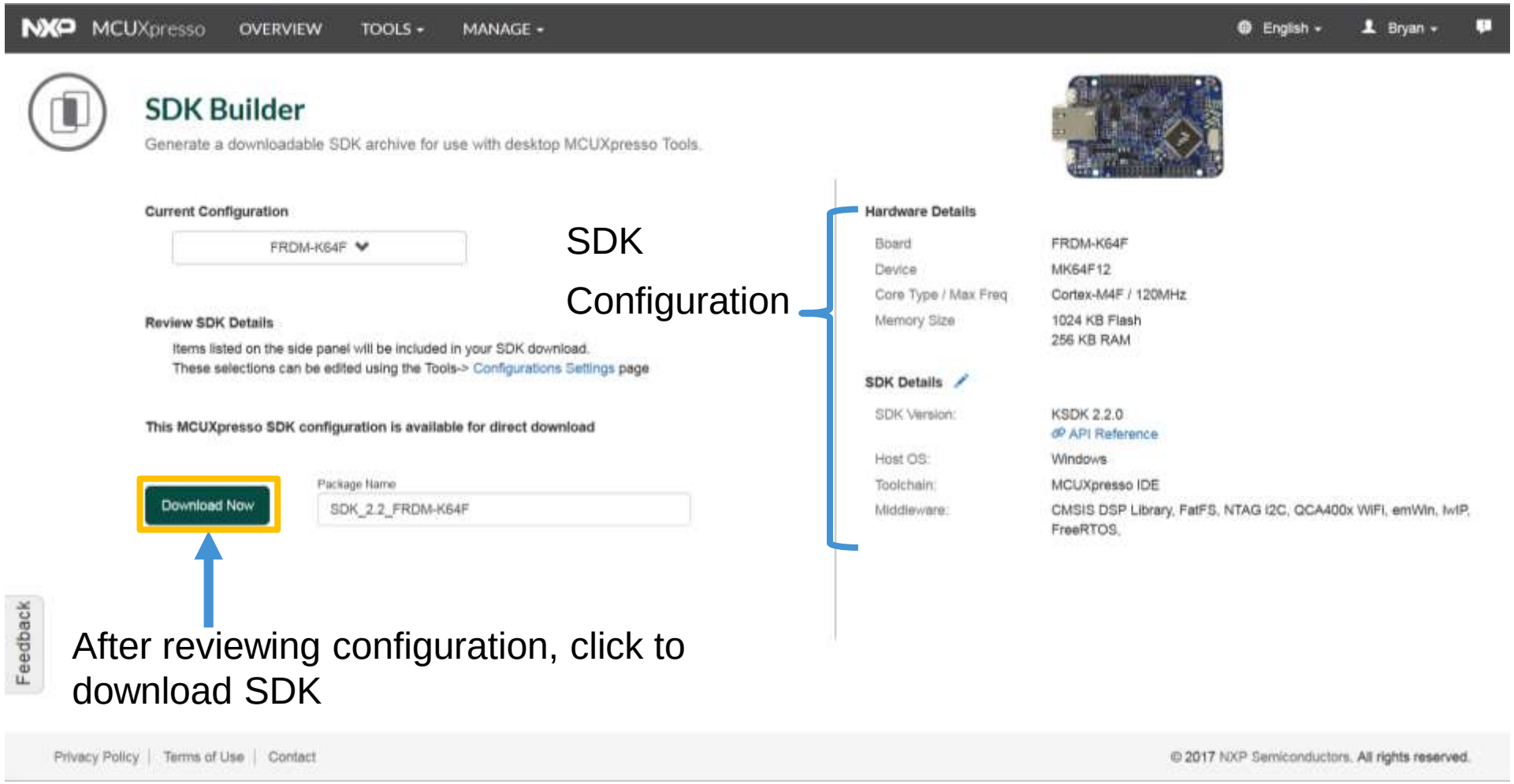


Finish Settings

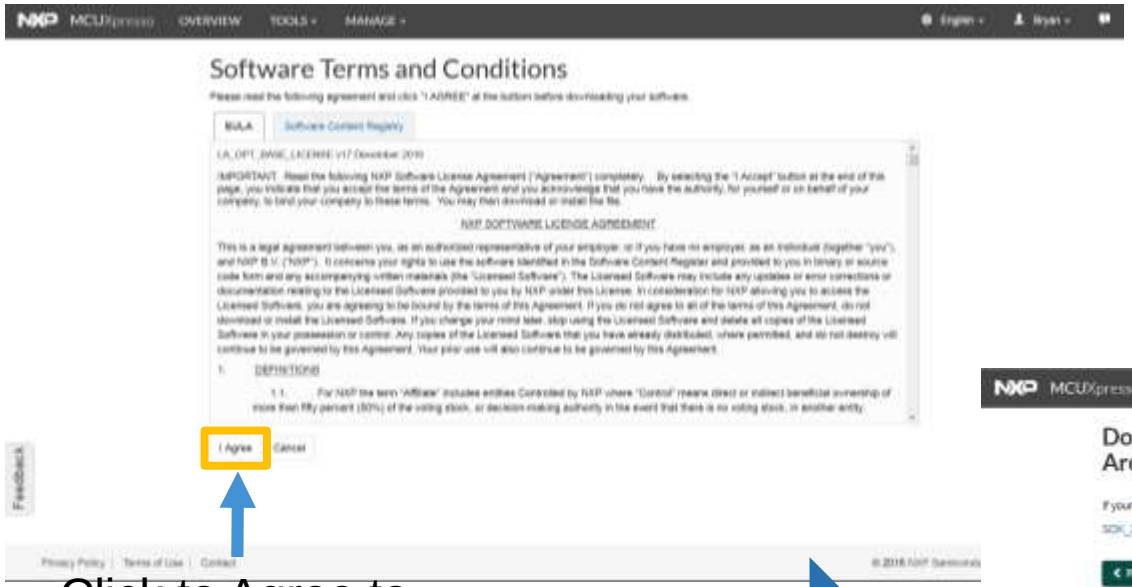


Once configurations are set, Go to SDK Builder

Build SDK

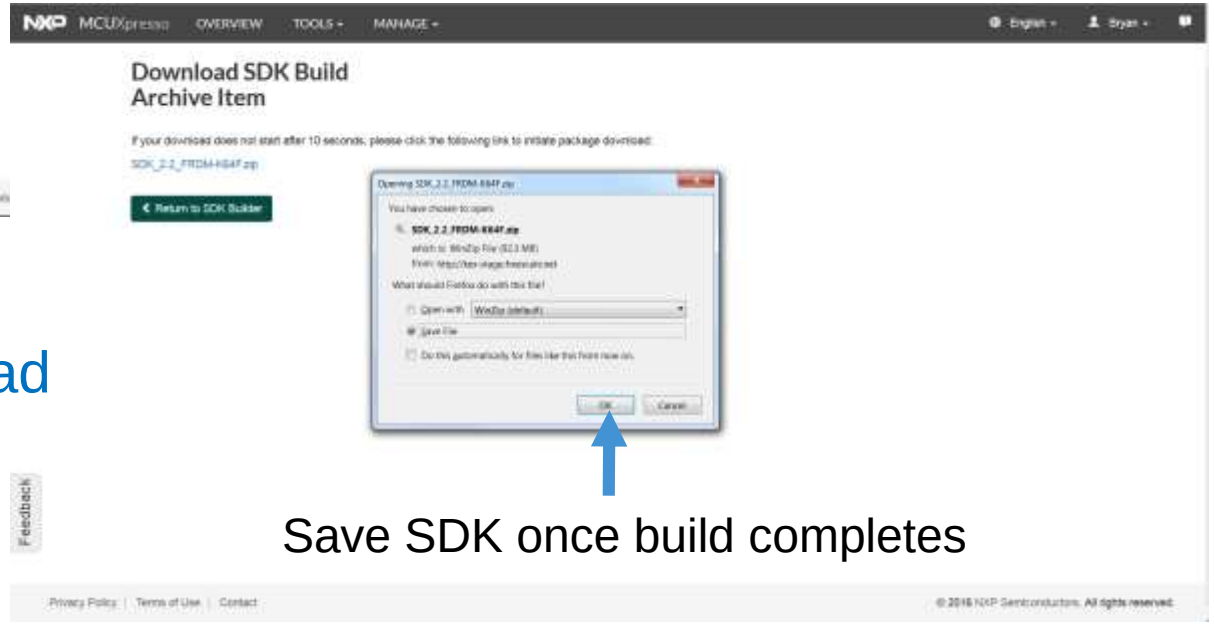

MCUXpresso
OVERVIEW
TOOLS
MANAGE
English
Bryan
SDK Builder
Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.
Current Configuration
FRDM-K64F
Review SDK Details
Items listed on the side panel will be included in your SDK download.
These selections can be edited using the Tools-> Configurations Settings page
This MCUXpresso SDK configuration is available for direct download
Download Now
Package Name
SDK_2.2_FRDM-K64F
SDK Configuration
Hardware Details
Board FRDM-K64F
Device MK64F12
Core Type / Max Freq Cortex-M4F / 120MHz
Memory Size 1024 KB Flash
256 KB RAM
SDK Details
SDK Version: KSDK 2.2.0
API Reference
Host OS: Windows
Toolchain: MCUXpresso IDE
Middleware: CMSIS DSP Library, FatFS, NTAG I2C, QCA400x WiFi, emWin, lwIP, FreeRTOS,
Feedback
After reviewing configuration, click to download SDK
Privacy Policy
Terms of Use
Contact
© 2017 NXP Semiconductors. All rights reserved.

Download SDK



Click to Agree to terms and conditions

SDK download will begin

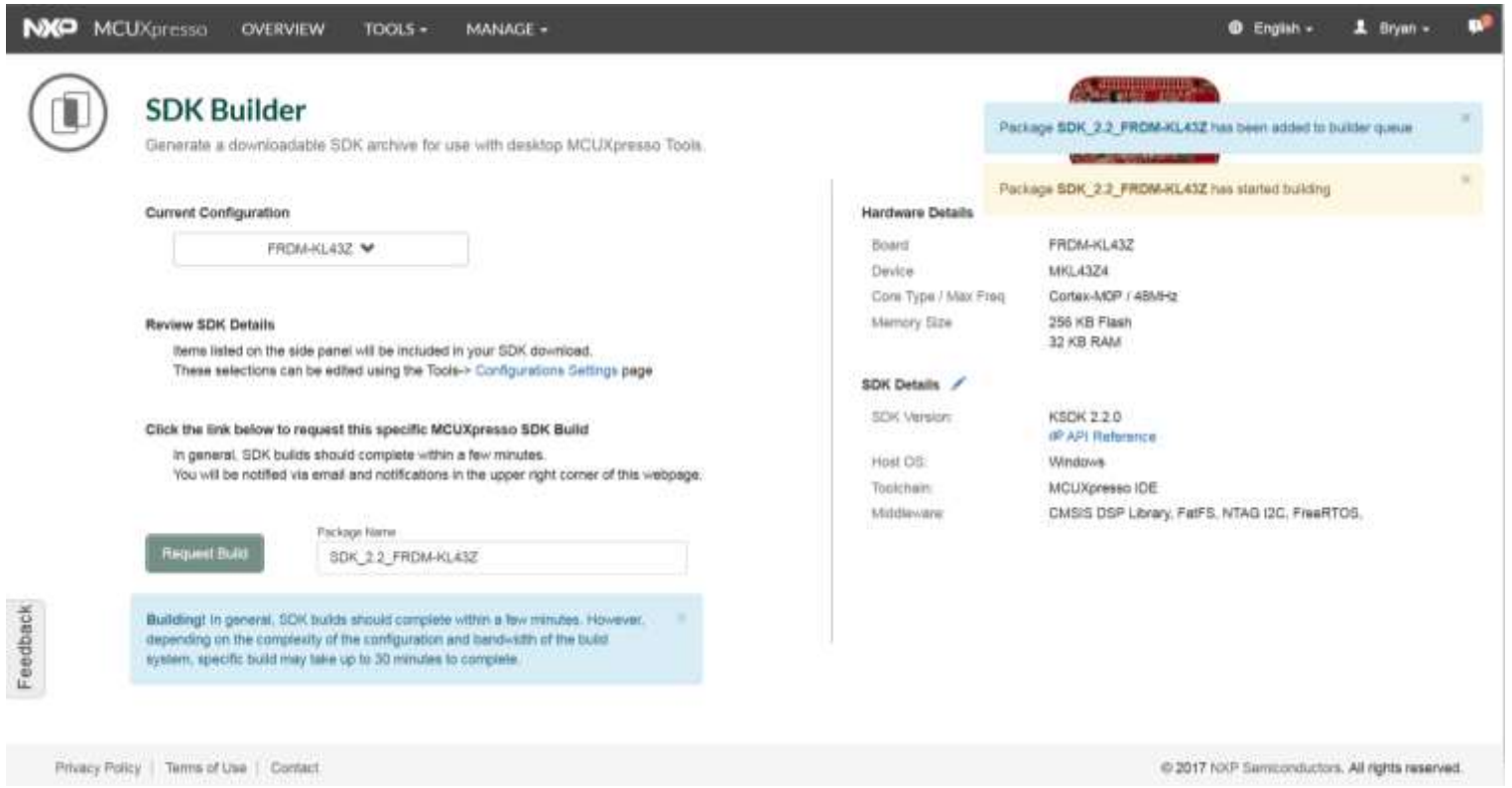


Save SDK once build completes

Request Build

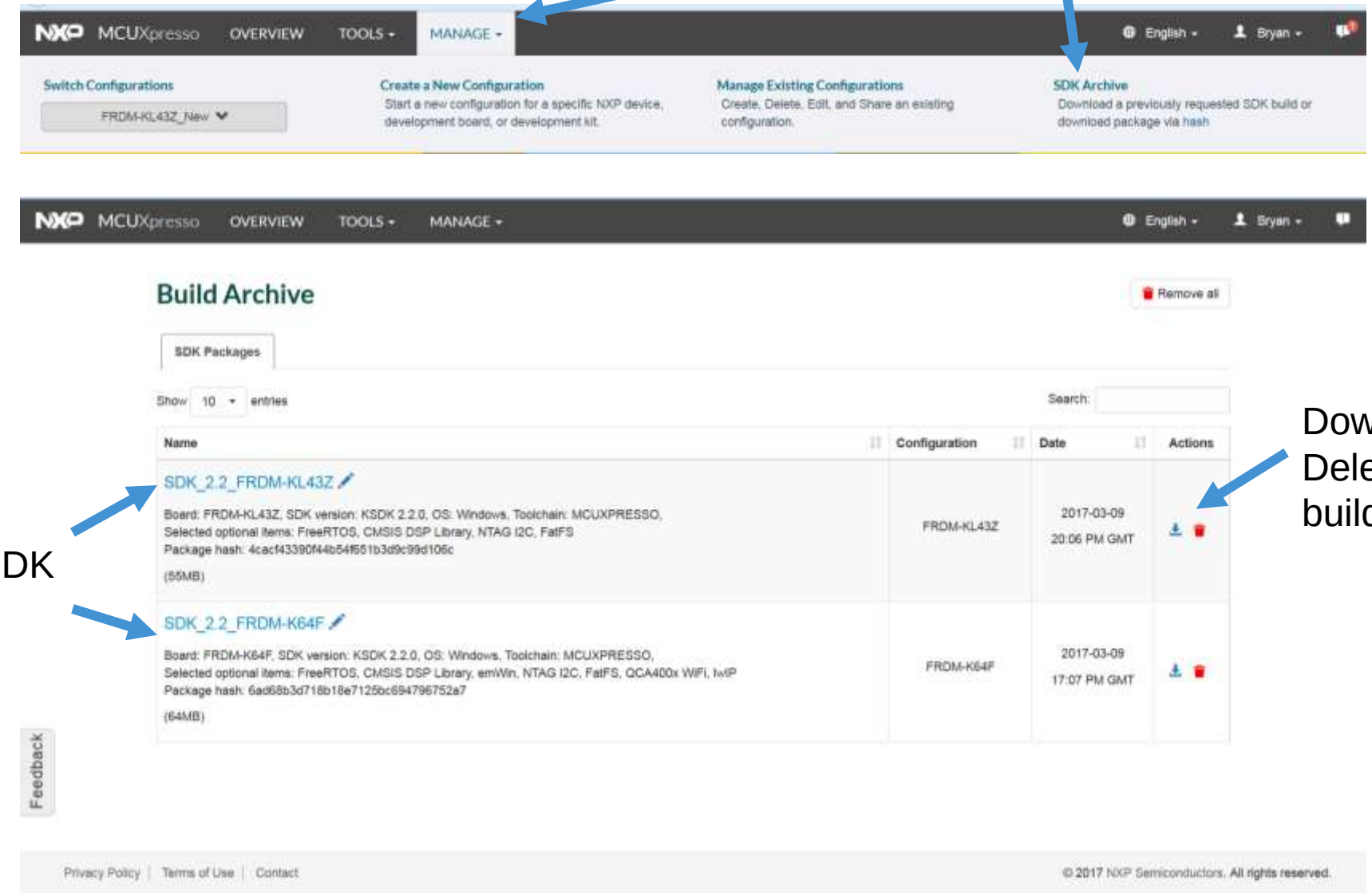
- In some occasions, if the SDK configuration has not previously been built, “Request Build” will be displayed in place of “Download Now”

- An email notification with direct link will be sent once the build is finished



Build Archive

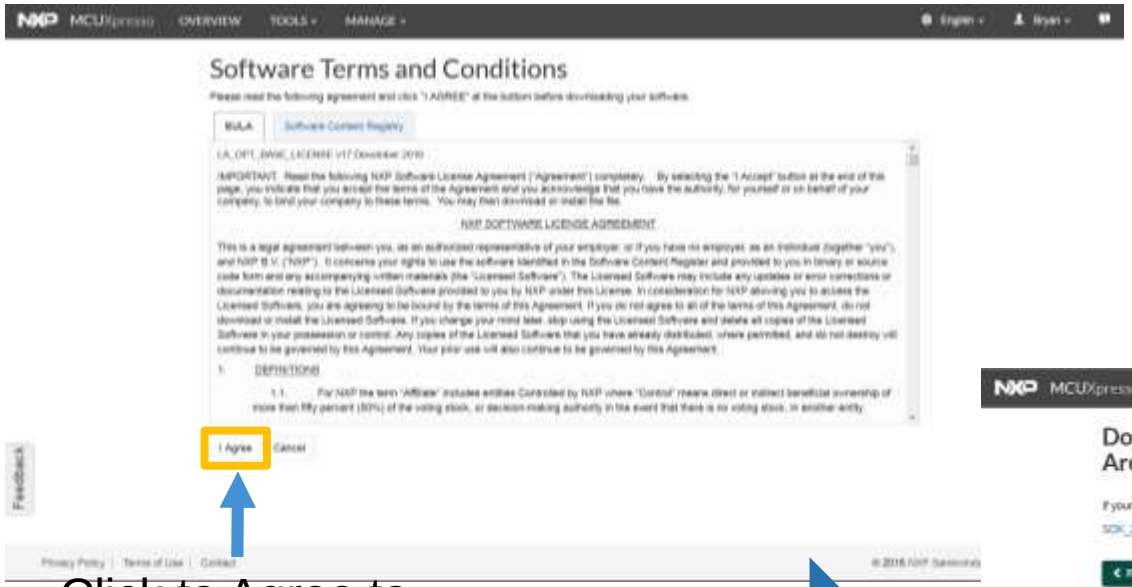
Access SDK Archive from Manage menu



Shows all SDK builds

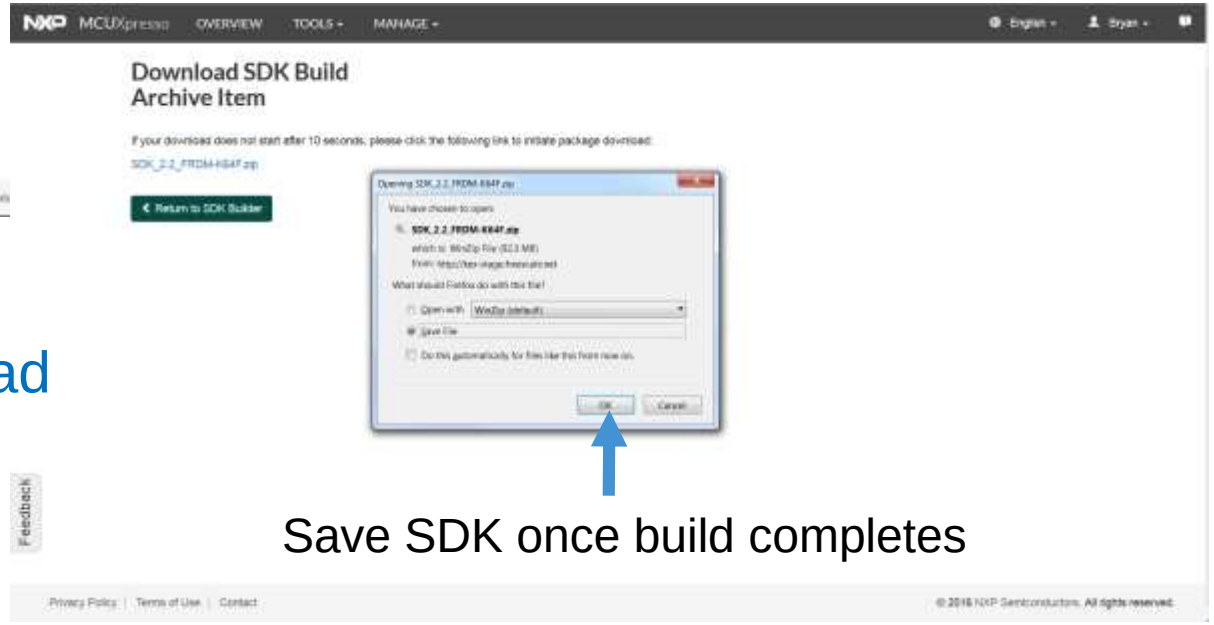
Download/ Delete SDK build

Download SDK



Click to Agree to terms and conditions

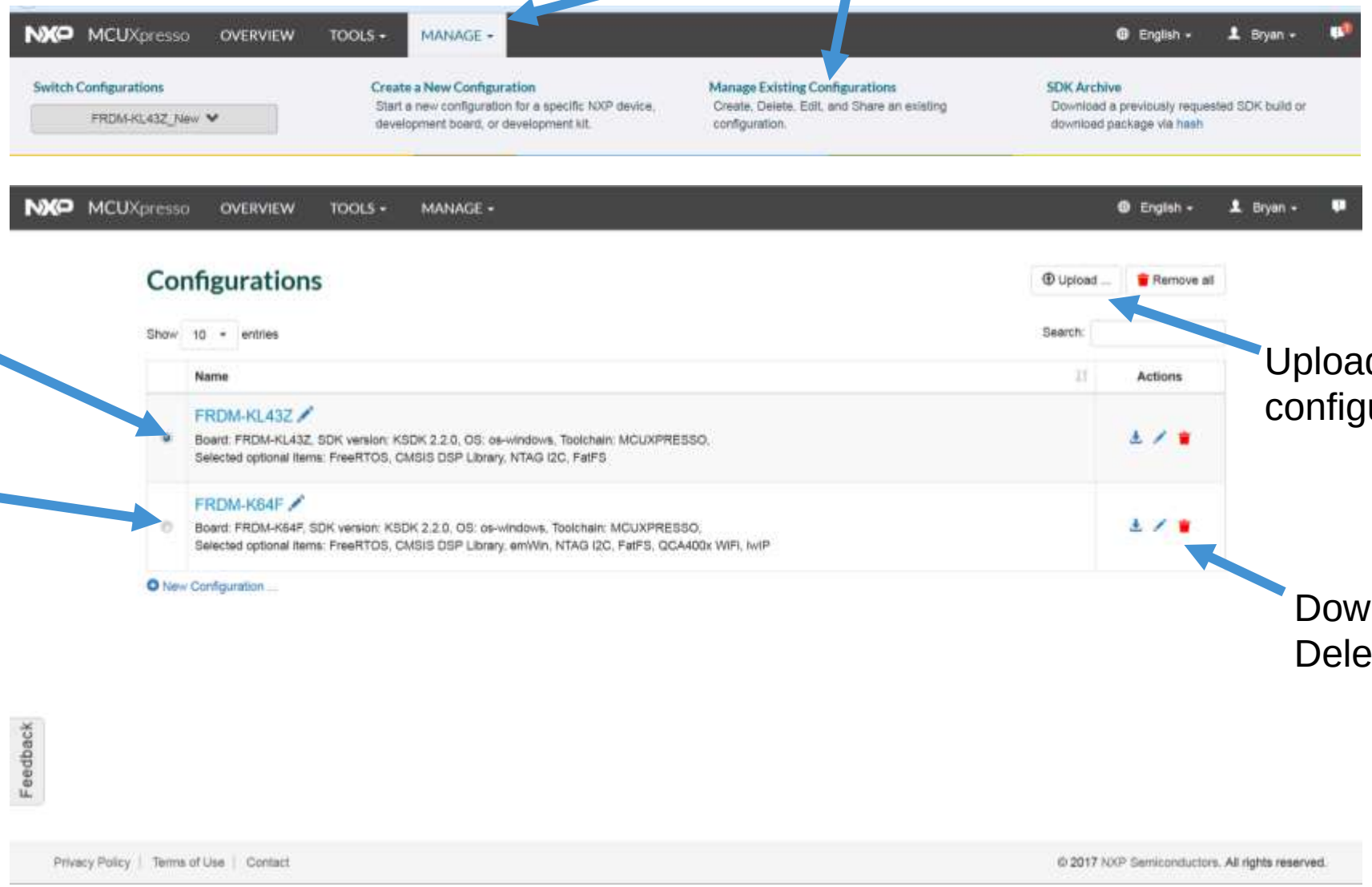
SDK download will begin



Save SDK once build completes

Configurations Archive

Access Configuration Archive from Manage menu



Current configuration

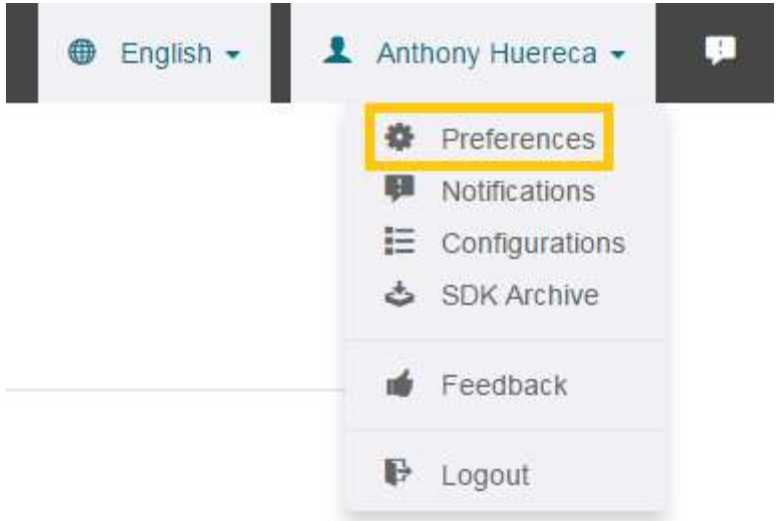
Previous configuration

Upload a configuration

Download/Edit/ Delete a configuration

Preferences

- First time users may see an error if they have not filled out profile in “Preferences" as required for export control compliance
  - Name
  - Company
  - Country
  - Project Description



Board Based SDK

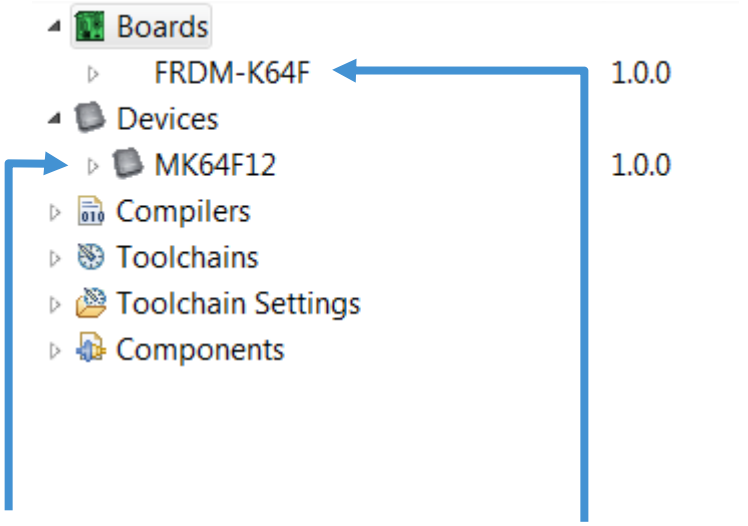
- Contains board-specific example projects and driver examples
- Contains processor support for device featured on board

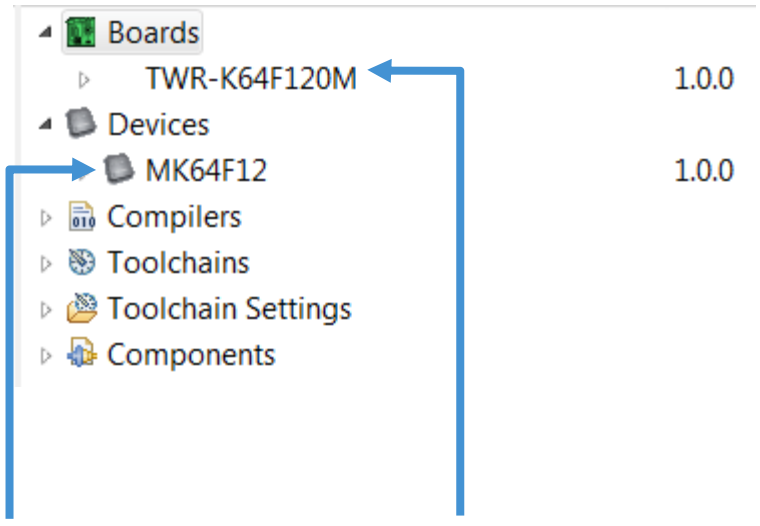
Hardware Details

Board	FRDM-K64F
Device	MK64F12
Core Type / Max Freq	Cortex-M4F / 120MHz
Memory Size	1024 KB Flash 256 KB RAM

Hardware Details

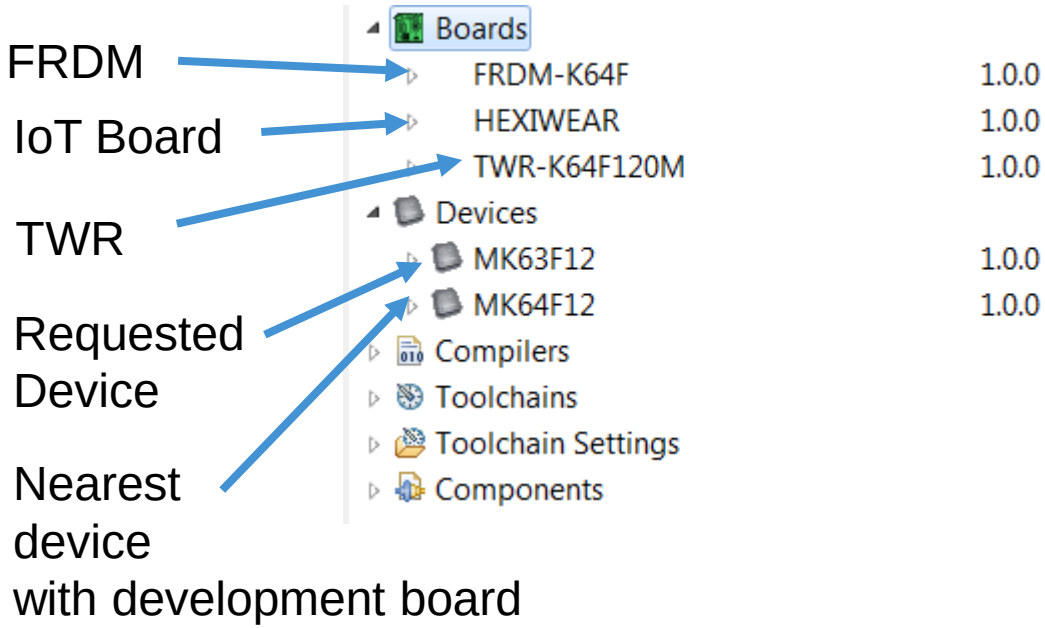
Board	TWR-K64F120M
Device	MK64F12
Core Type / Max Freq	Cortex-M4F / 120MHz
Memory Size	1024 KB Flash 256 KB RAM





Processor Based SDK

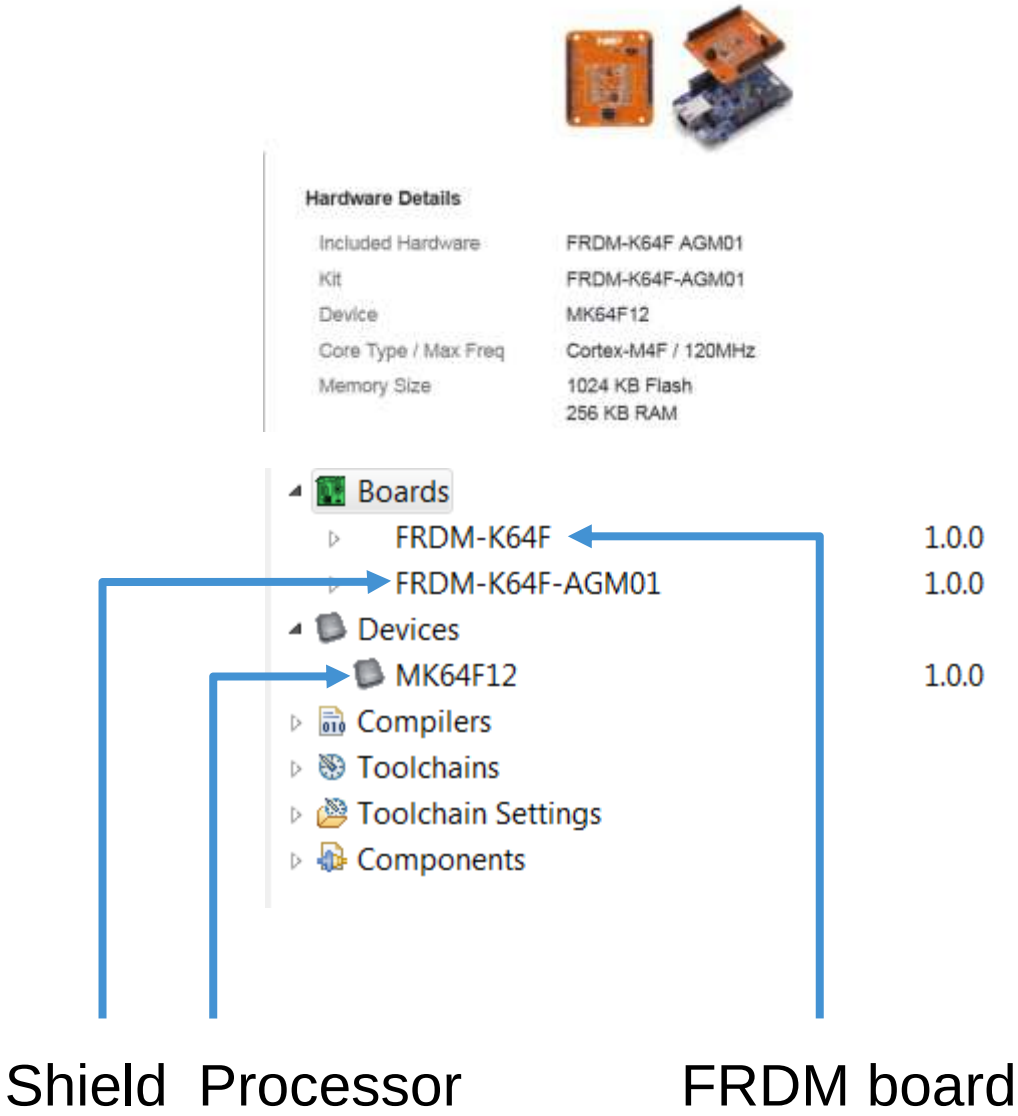
- Contains processor support and all recommended board support
- If a part is chosen that has no directly matching board, say the Kinetis MK63… then the generated SDK will contain:
  - part support for the requested part i.e. MK63...
  - part support for the recommended closest matching part that has an associated development board i.e. MK64...
  - board support packages for the above part i.e. Freedom and/or Tower MK64...



Kit Based SDK

- Contains example projects for kit

- Contains processor support for development board



MCUXpresso SDK Structure

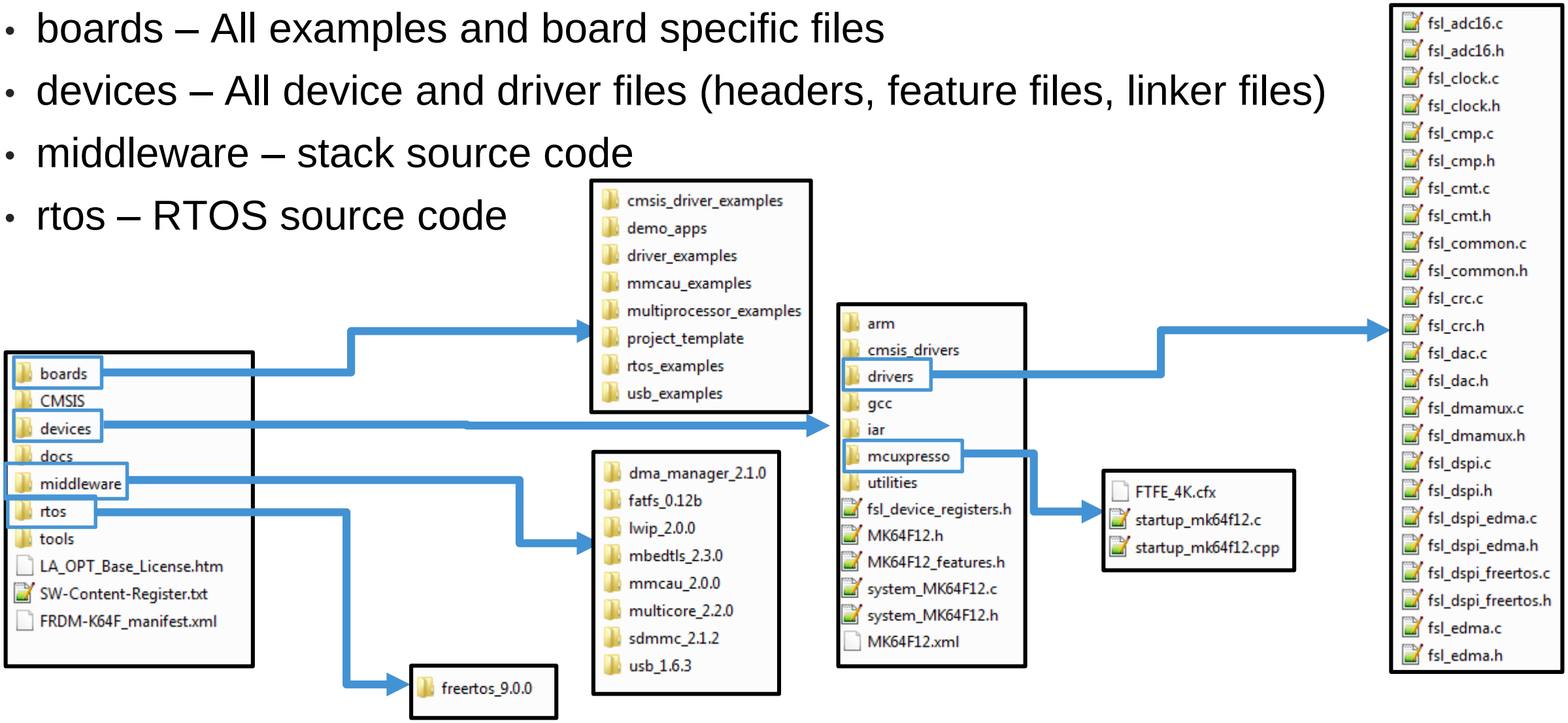
Zip or Unzip an SDK package

- SDK packages are downloaded as .zip files
- When using 3rd party IDEs, the SDK package must be unzipped
- For SDK support in the MCUXpresso Config Tools, the SDK package must also be unzipped

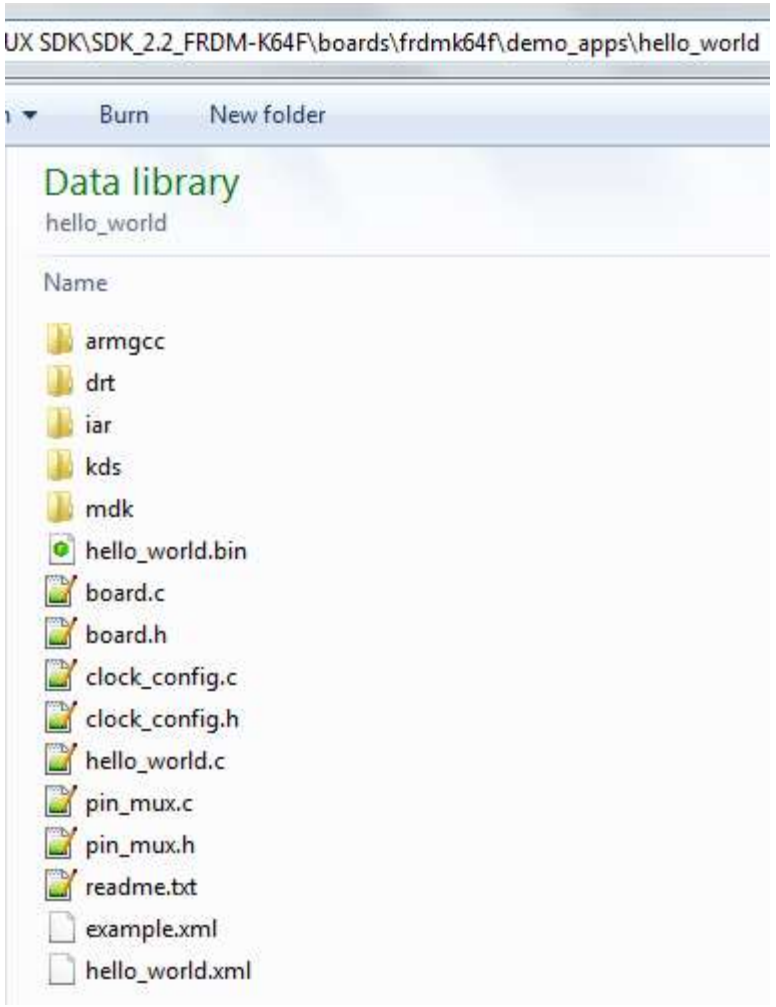
- MCUXpresso IDE can import SDK packages in either zipped or unzipped format.
  - Zipped SDKs:
    - When creating new projects or importing example projects, SDK source files are copied into the workspace (no linked references).
  - Unzipped SDKs:
    - When creating new projects or importing example projects, SDK source files can be copied into the workspace **or** referenced directly (linked references).
    - Requires additional time to unzip (one-time).
    - Provides speed improvement when many examples are imported to the workspace.

MCUXpresso SDK 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



MCUX Expresso SDK File Structure - Examples



- Each example application has its own unique copy of the board, pin_mux, and clock_config files.
- Also each example also contains a pre-compiled .bin file for easy drag-and-drop programming
- Readme.txt contains instructions on how to run the demo and pins used

MCUXpresso File Structure - Examples

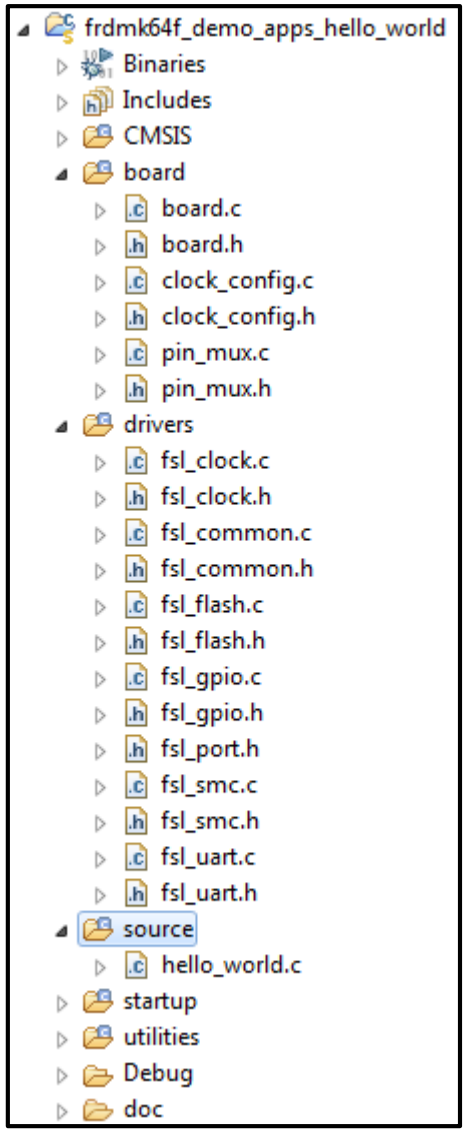
- Most configuration settings are in **board.h** file
  - UART module
  - UART baud
  - GPIO pins defined

```
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"
63
```

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

MCUXpresso SDK Projects

- All source files are included in the example application projects
- Drivers are found under the **drivers** folder
- Board specific files under the **board** folder
- Application specific files under **source** folder



MCUXpresso SDK Startup

- Reset_Handler found in \devices\<device>\<compiler>\startup_<device>.s
  - Called ResetISR for MCUXpresso IDE
- SystemInit() found at \devices\<device>\system_<device>.c is used to enable cache (if available) and disable the watchdog timer.
- Then jumps to main(), and three configuration functions run:
  - **BOARD_InitPins();**
  - **BOARD_BootClockRUN();**
  - **BOARD_InitDebugConsole();**

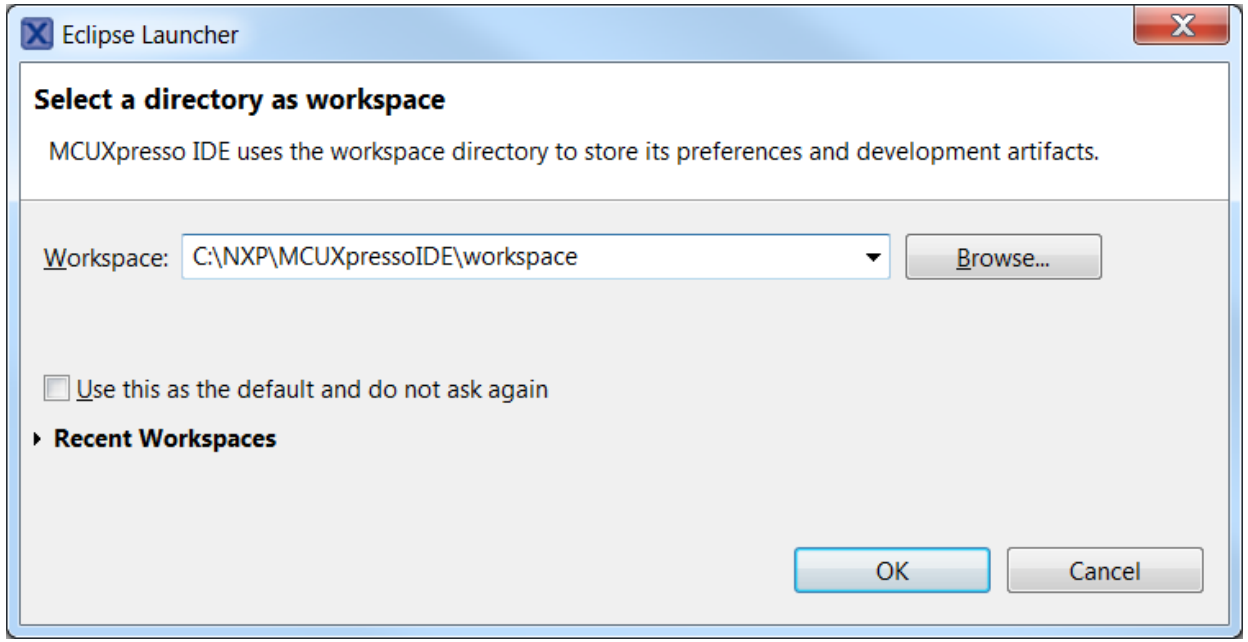
MCUXpresso SDK Notes

- As NPI devices are added, they will not support Kinetis Design Studio.

MCUXPRESSO IDE

Open MCUXpresso IDE

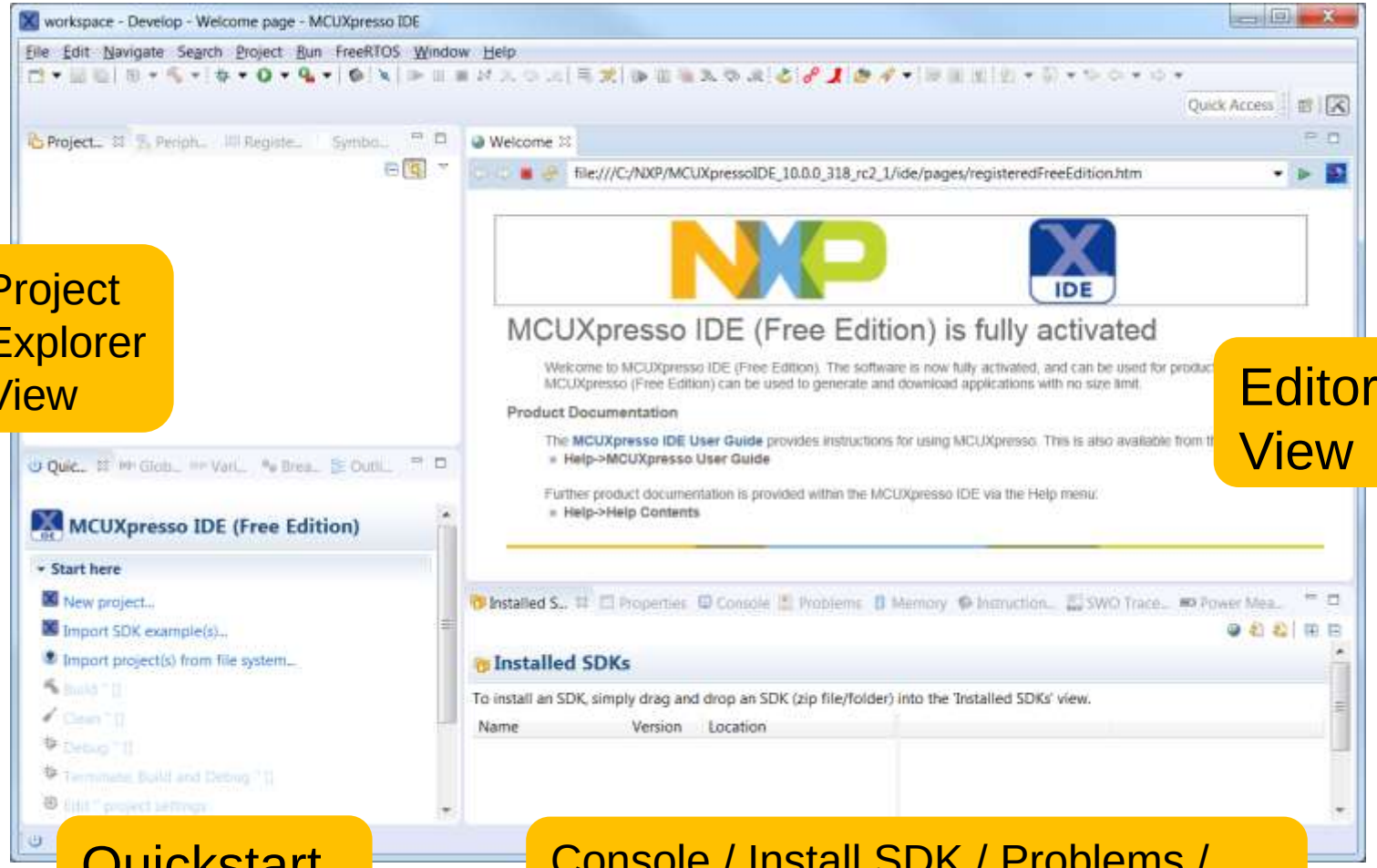
- Open MCUXpresso IDE on your system
- At the dialog box, enter a location for your workspace then click OK
  - Example) C:\NXP\MCUXpressoIDE\workspace
- Note: A workspace is a directory used to store projects that you want to actively work on during the IDE session



www.nxp.com/mcuxpresso/ide

Develop Perspective

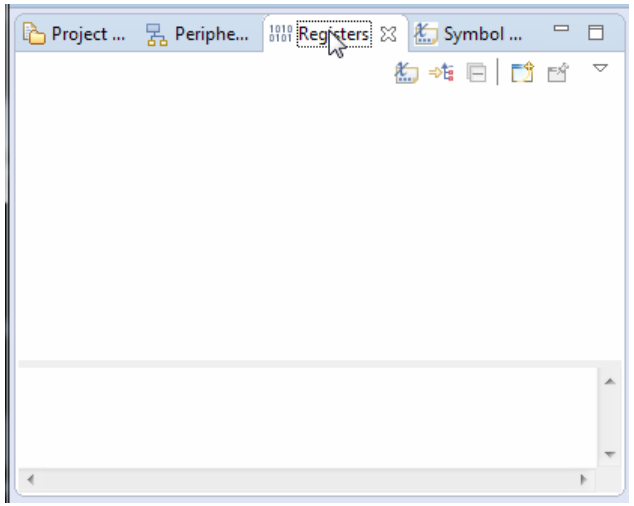
- MCUXpresso IDE will startup in a new workspace with no projects in the Develop Perspective
- A “perspective” is a collection of different “views”
- The Develop perspective provides a single combined project management and debugging view
- In addition to the default Develop perspective, the MCUXpresso IDE also supports traditional Eclipse C/C++ and Debug perspectives

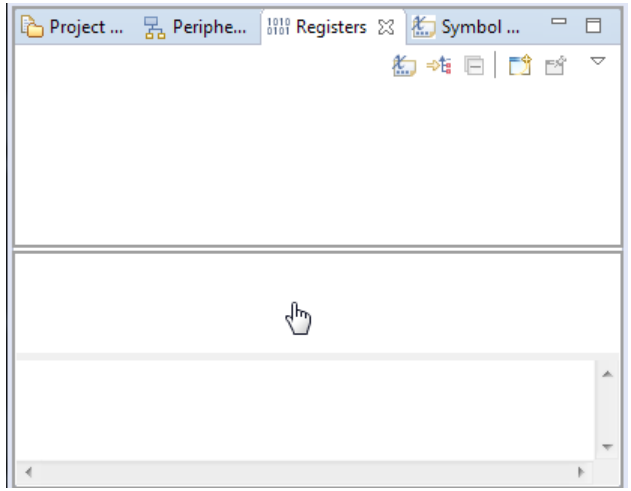


Changing the Layout of the Develop Perspective

- Layout of views within a perspective can be tailored to meet your personal needs
- For example, if we wanted to have the Registers view always visible…

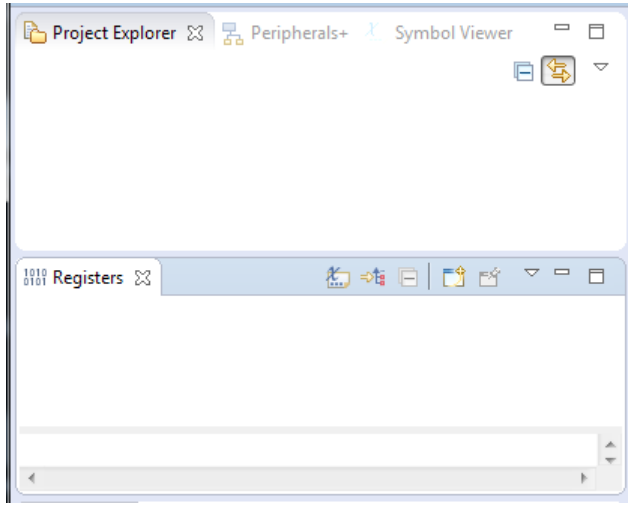
Click and hold down on the View you want to move

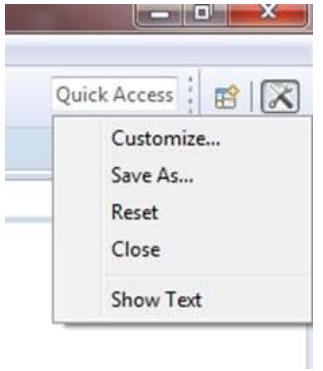




Continue to hold down and drag the cursor to the location you want to view to be displayed

Then release the mouse click, and the view will be placed at the required position



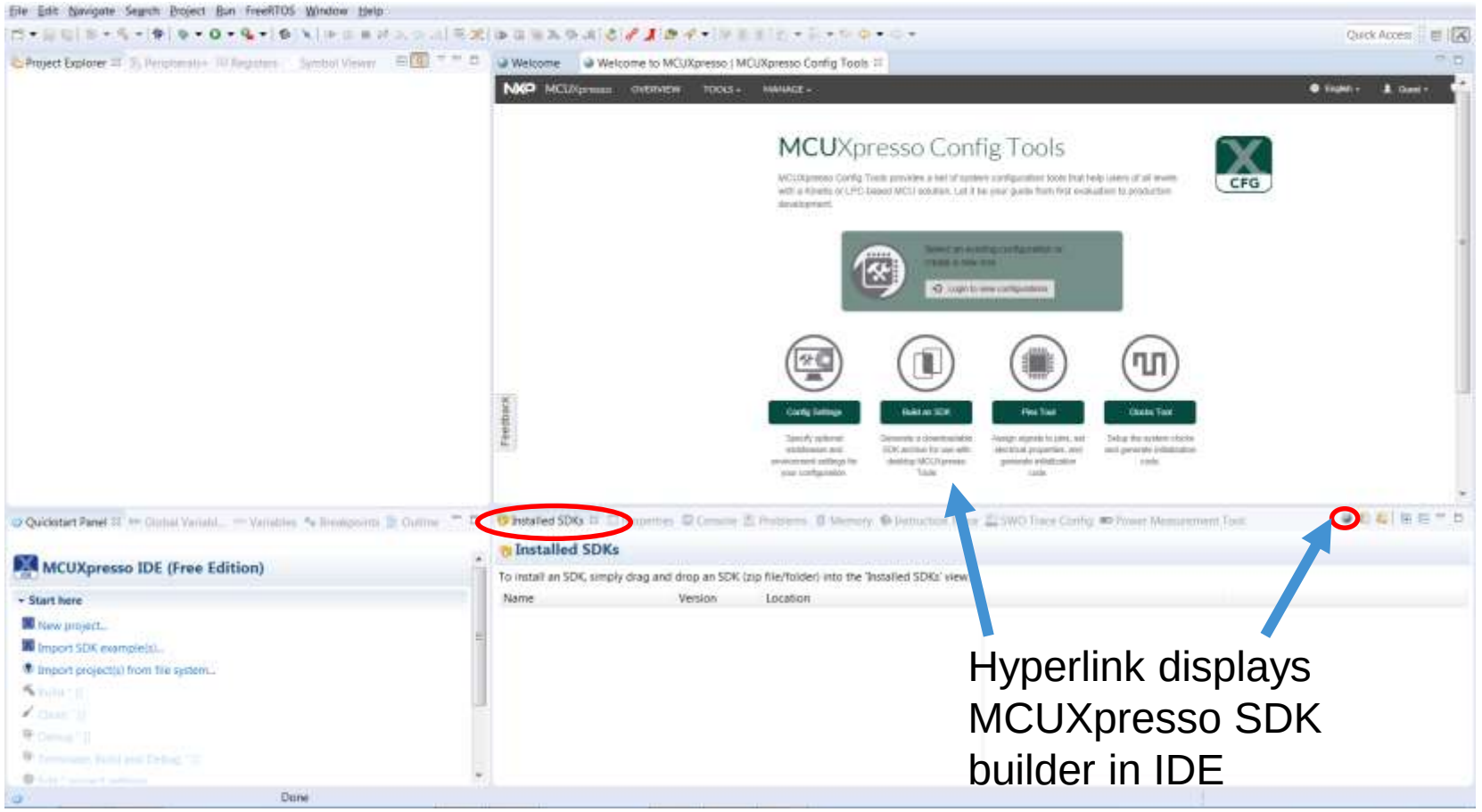


Right click on the Perspective button (top right of IDE window) to reset the layout back to the default

Install SDK

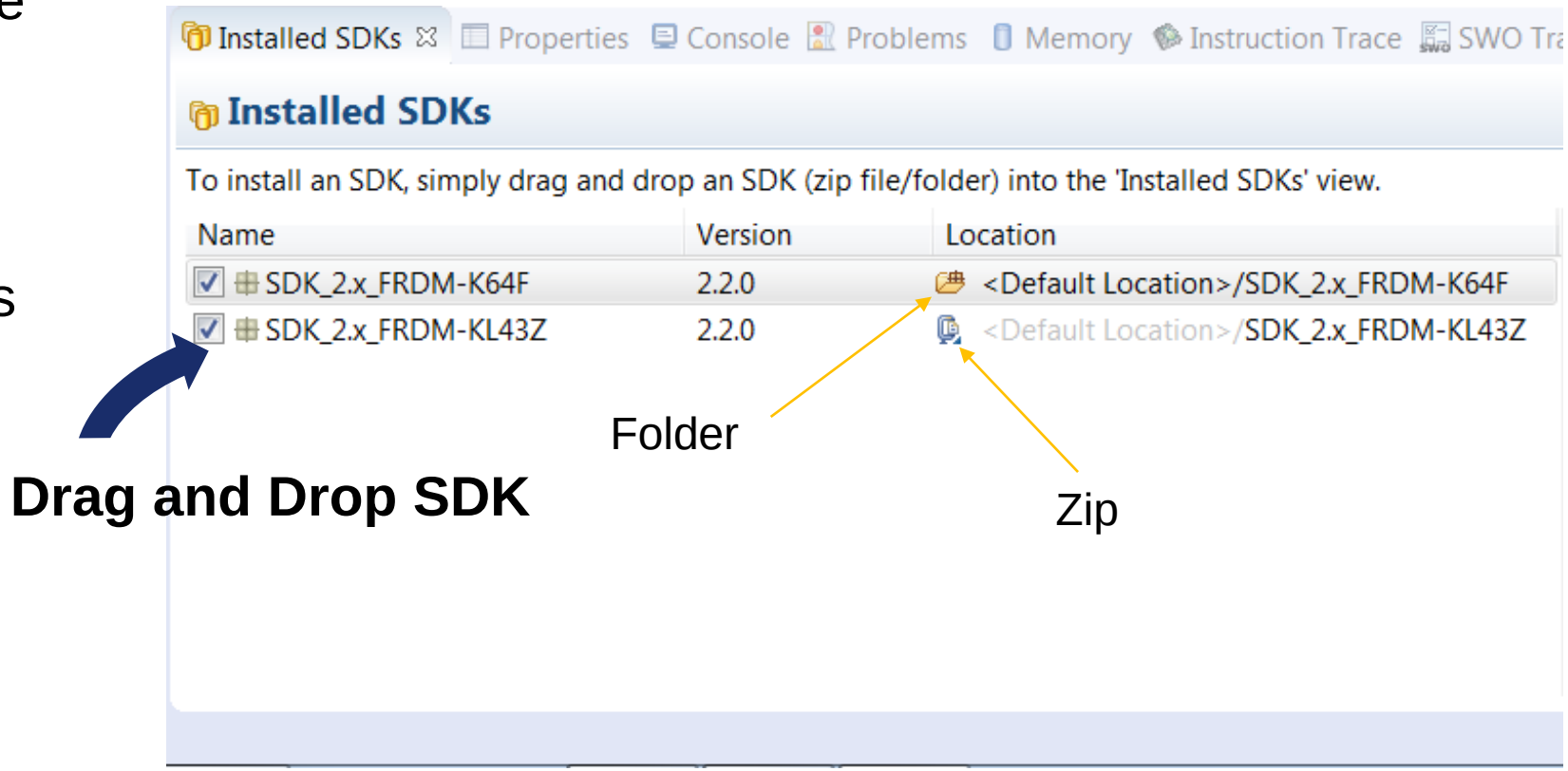
Installing an SDK in the IDE

- Part support is added by installing MCUXpresso SDKs into the IDE
- Allows example projects and driver examples from SDK to be easily imported
- New project generation based on board or processor in SDK
- The IDE is only compatible with SDKs built for MCUXpresso

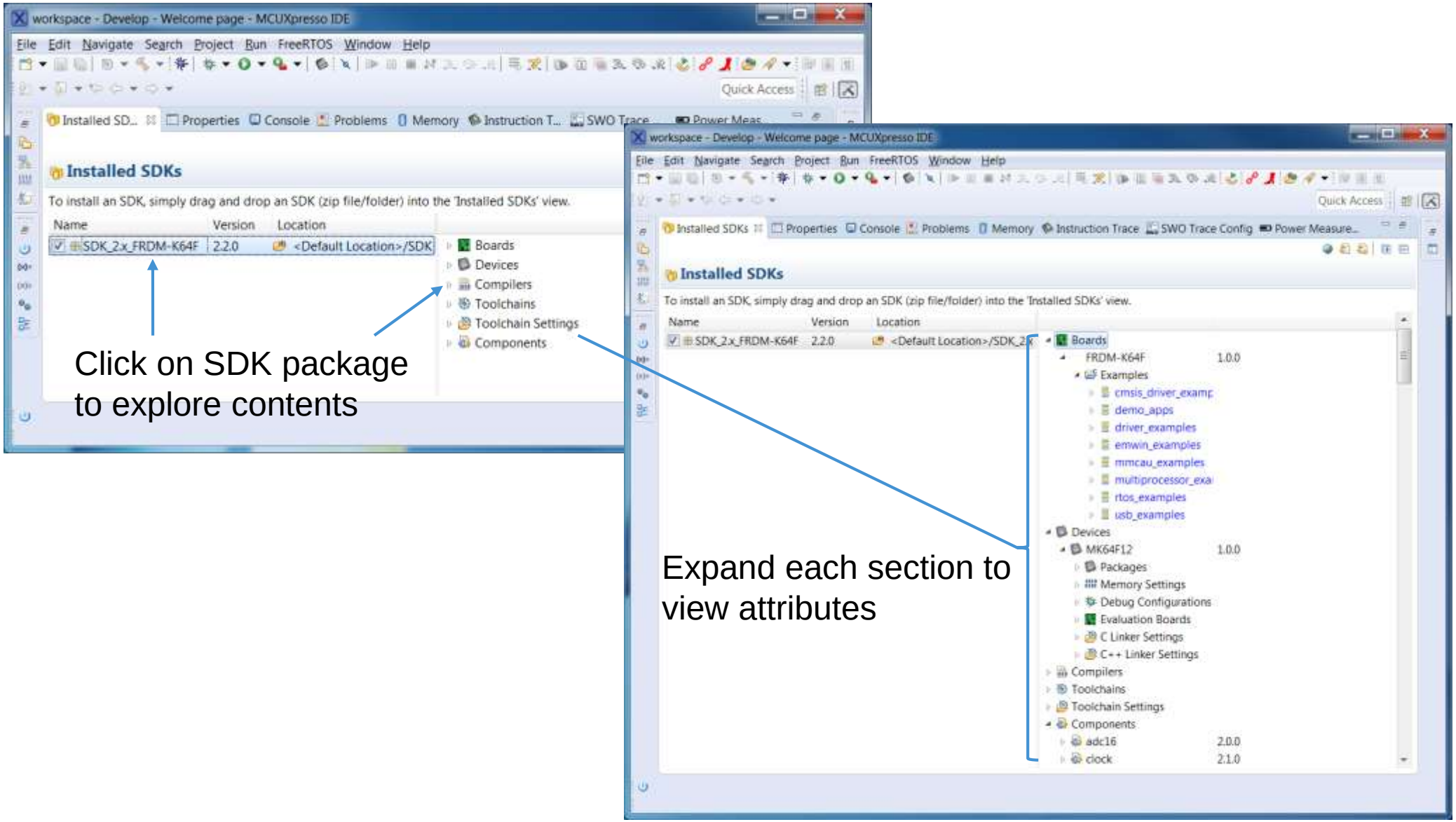


Install an SDK: Drag and Drop

- Drag/Drop SDK packages directly into the IDE in the **Installed SDKs** view
- Can drag SDK as folder or zip (archive). IDE uses separate icon for each type
- SDKs installed in the default location are shared across workspaces



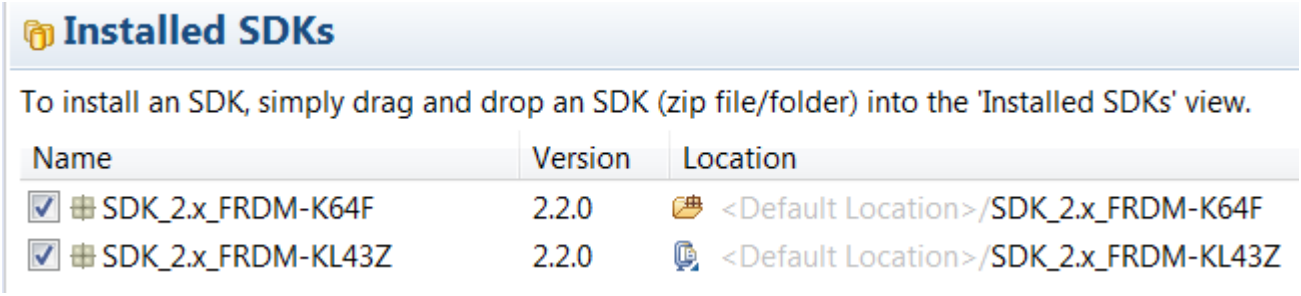
Inspect SDK

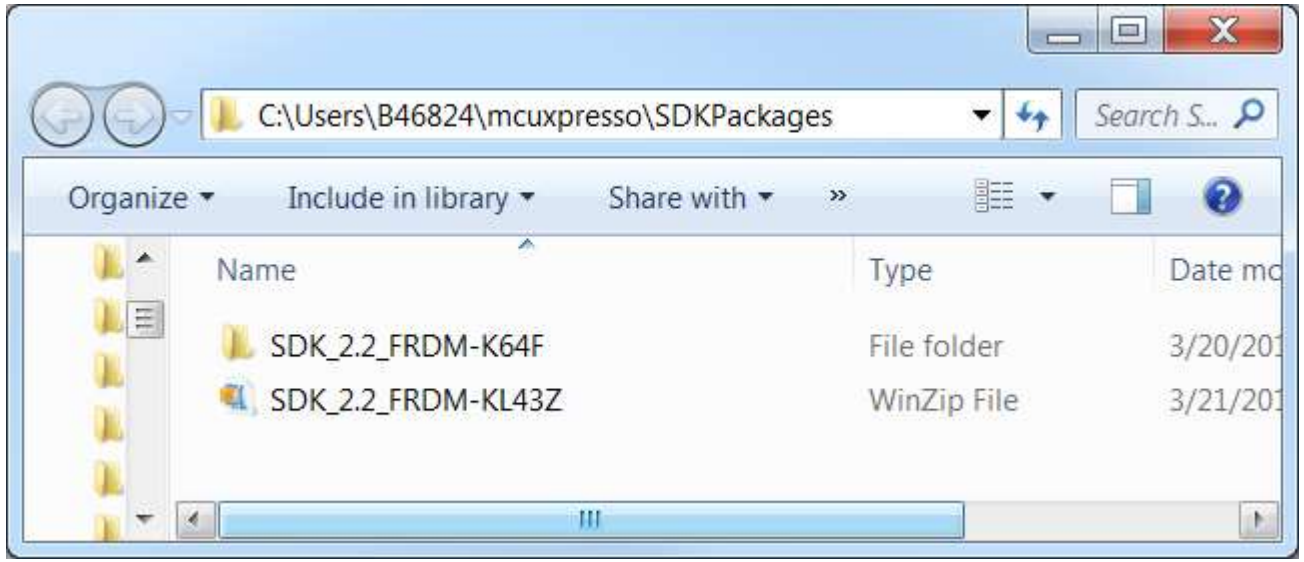

workspace - Develop - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project Run FreeRTOS Window Help
Quick Access
Installed SD... Properties Console Problems Memory Instruction T... SWO Trace Power Meas...
Installed SDKs
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view.
Name Version Location
SDK_2.x_FRDM-K64F 2.2.0 <Default Location>/SDK
Boards
Devices
Compilers
Toolchains
Toolchain Settings
Components
Click on SDK package to explore contents
workspace - Develop - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project Run FreeRTOS Window Help
Quick Access
Installed SDKs Properties Console Problems Memory Instruction Trace SWO Trace Config Power Measure...
Installed SDKs
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view.
Name Version Location
SDK_2.x_FRDM-K64F 2.2.0 <Default Location>/SDK_2.x
Boards
FRDM-K64F 1.0.0
Examples
cmsis_driver_examp
demo_apps
driver_examples
emwin_examples
mmcau_examples
multiprocessor_exa
rtos_examples
usb_examples
Devices
MK64F12 1.0.0
Packages
Memory Settings
Debug Configurations
Evaluation Boards
C Linker Settings
C++ Linker Settings
Compilers
Toolchains
Toolchain Settings
Components
adc16 2.0.0
clock 2.1.0
Expand each section to view attributes

Copy of SDK made in default path

- What happens when an SDK is dragged/dropped into the IDE?

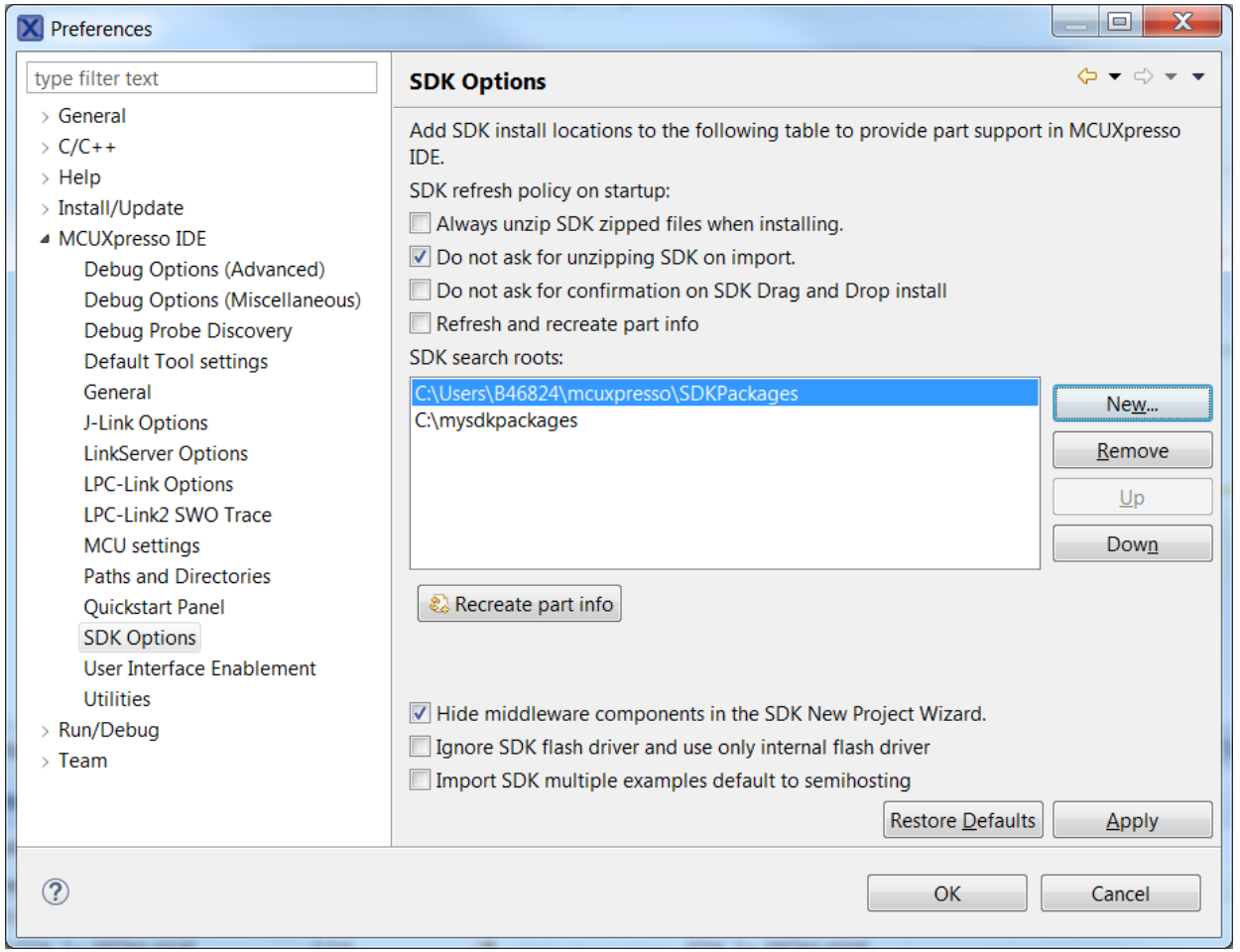
- The Drag/Drop feature creates a copy of the SDK located at **default path**: C:\Users\“user_name”\mcuxpresso\SDKPackages





Install an SDK: Advanced

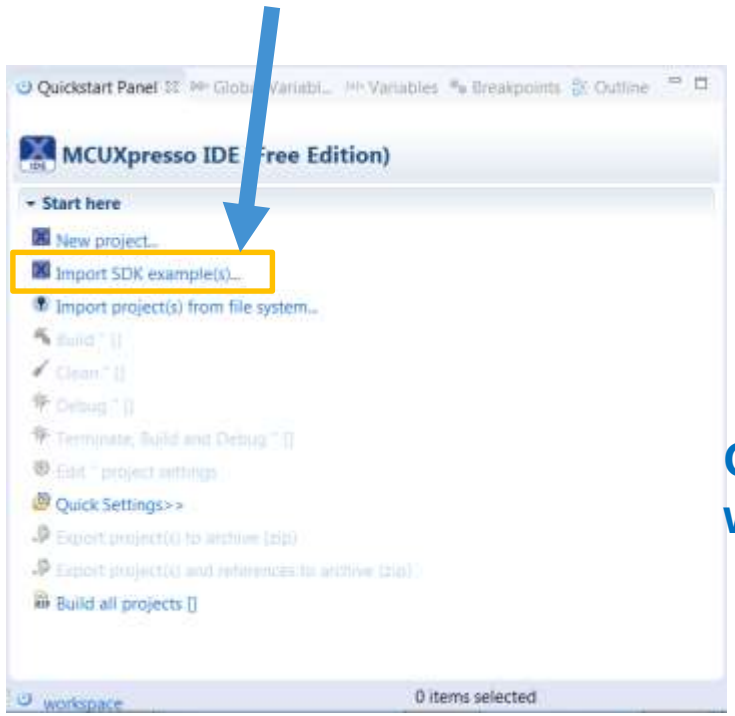
- Add paths to “SDK search roots:” for IDE to find current or future stored SDK packages
  - Window -> Preferences -> MCUXpresso IDE -> SDK Options
- SDKs can be zipped or unzipped
- For SDKs stored outside the default location:
  - “Delete SDK” function is disabled
  - Knowledge of SDKs is per workspace
- If multiple SDKs are found for the same device in various locations, you can choose which is loaded by reordering list (top has priority)
- Note: default location for drag/drop: C:\Users\”user_name”\mcuxpresso\SDKPackages



Importing SDK example

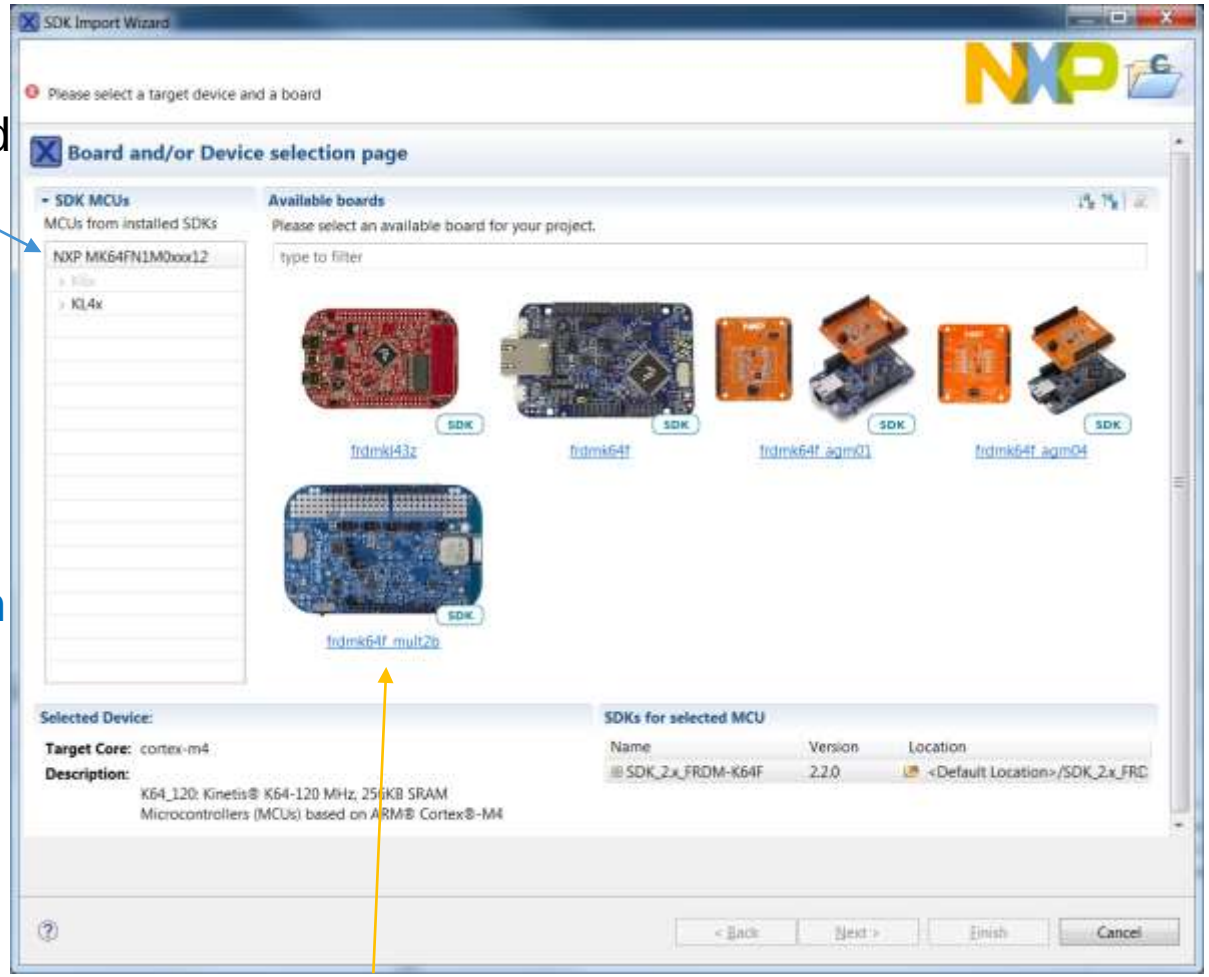
Import an SDK Example into the workspace

1. Click “Import SDK examples…” from Quickstart panel



Processors from installed SDKs

Opens selection wizard

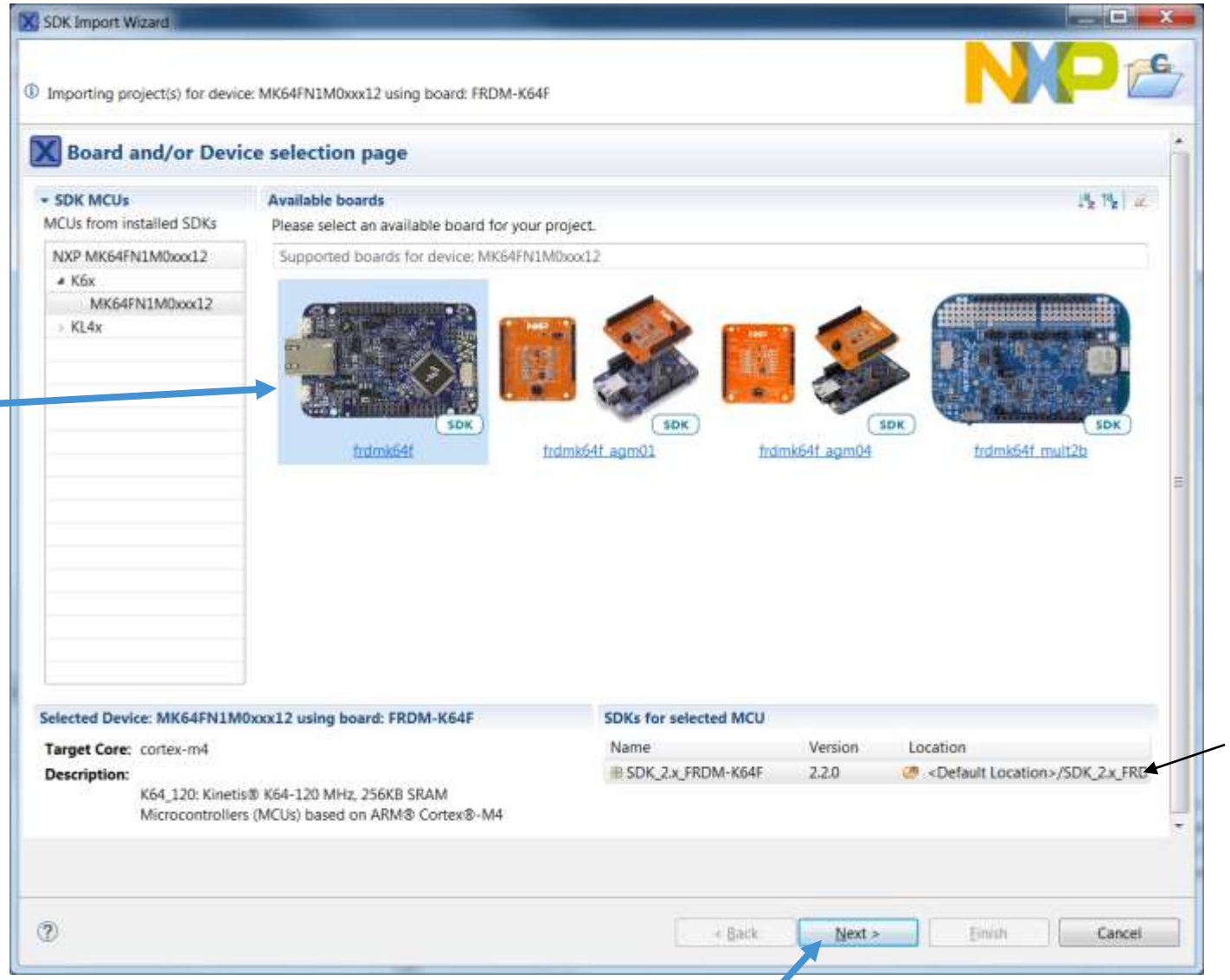


Boards from installed SDKs and preinstalled LPC boards

- SDK examples are board specific

Import an SDK Example into the workspace

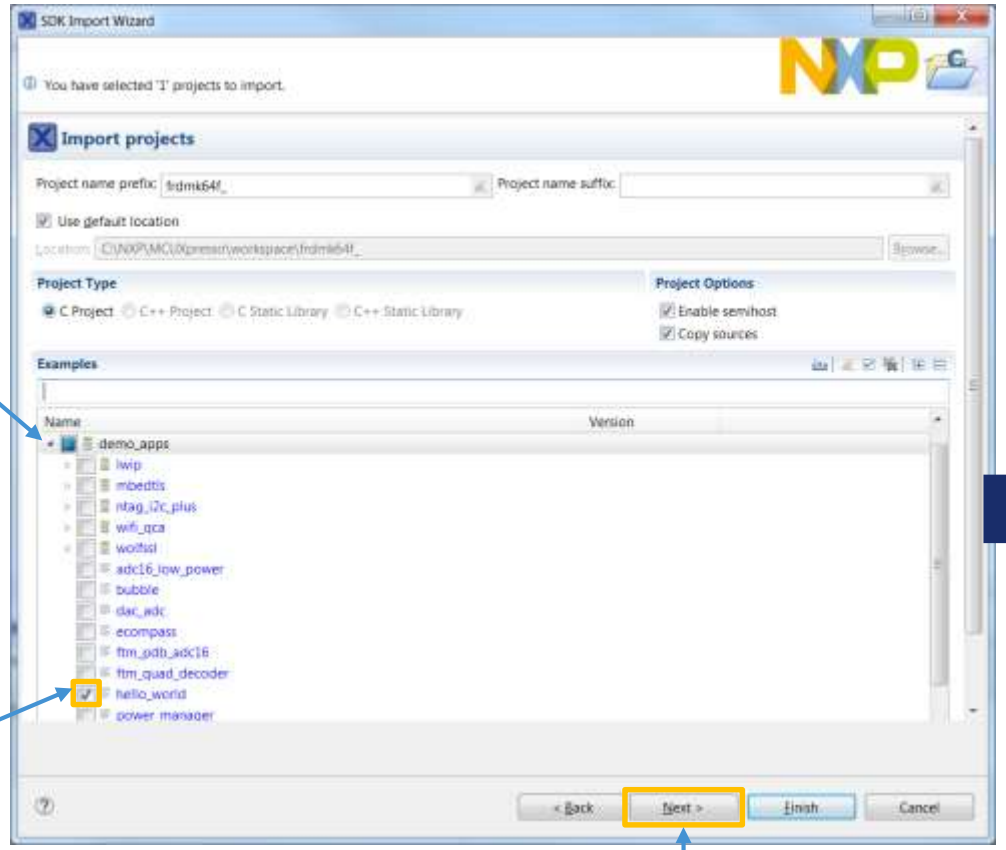
2. Click on board image to import an SDK example

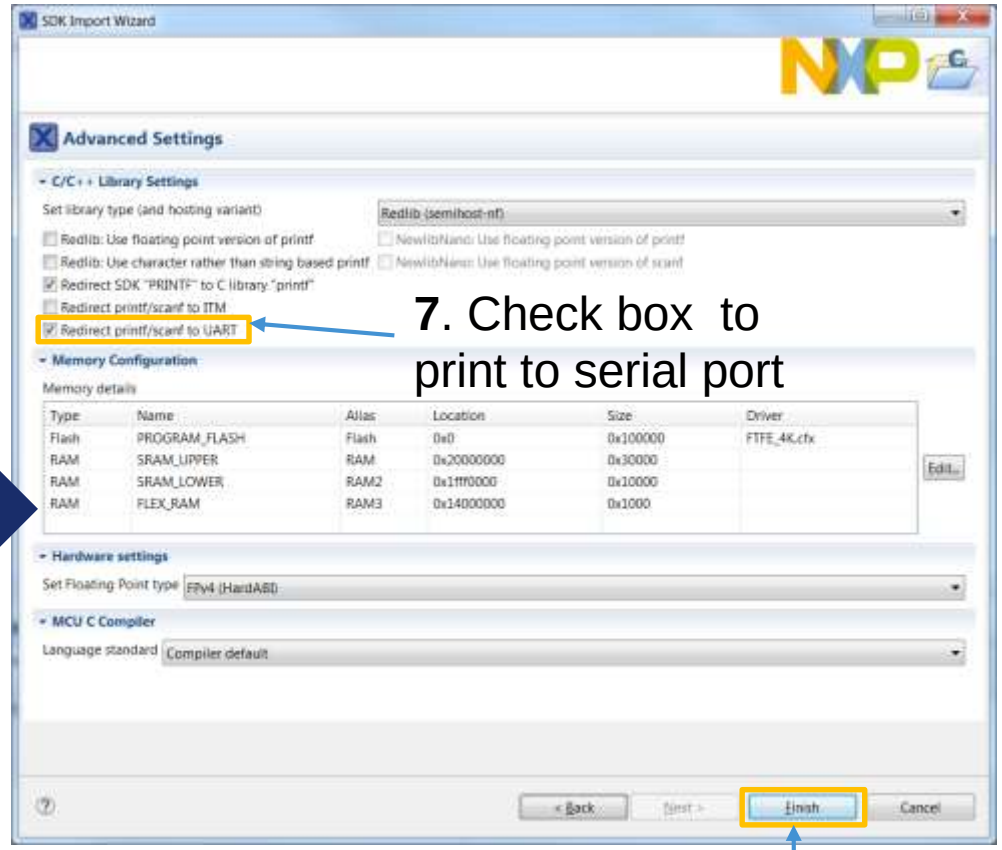


Installed SDK for selected board

3. Select Next to continue

SDK Example Import Wizard





4. Expand examples

5. Select project

6. Click Next

7. Check box to print to serial port

8. Click Finish

Copy Sources

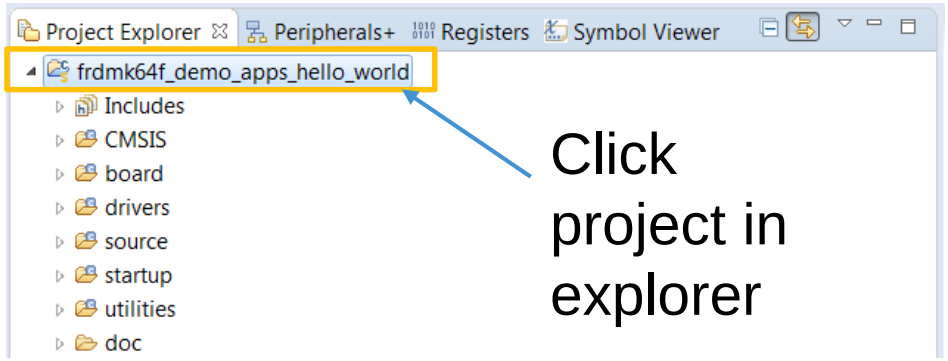
- If copy sources is selected, files needed for example project are copied from the installed SDK into the project folder located in your workspace
- If the SDK was zipped this option would be selected automatically and greyed out
- If Copy Sources is not selected, SDK source files used in the project are linked directly from the installed SDK
- **NOTE:** Linking sources will modify the installed SDK

Sharing Projects

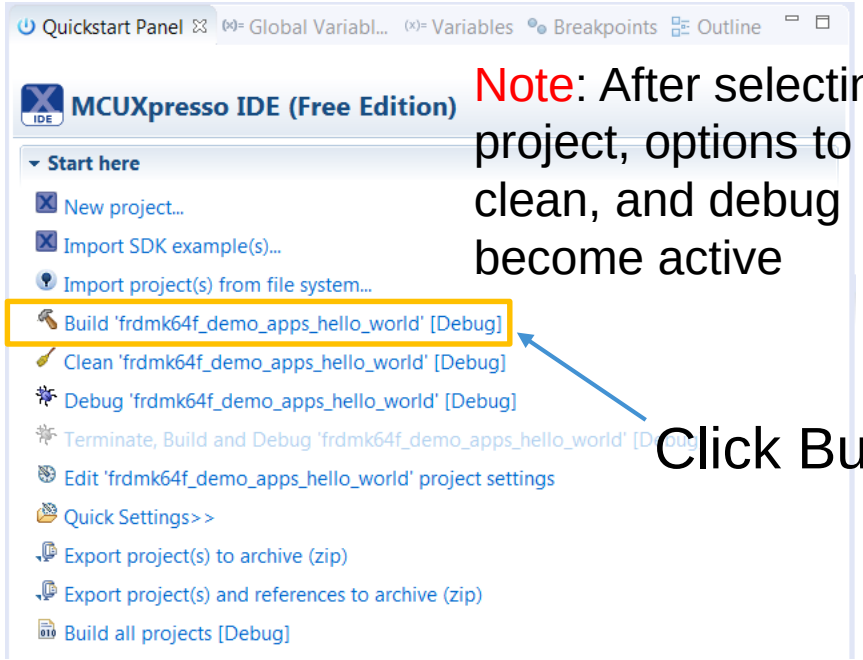
- If a project is built using part support from an SDK and is then exported – for example to share the project with a colleague who also uses MCUXpresso IDE, then the colleague must also install an SDK providing part support for the project's MCU.

- **Note**: Because device support is included in the SDK, it is recommended that any required SDKs are installed before a project requiring SDK part support is imported. However, if this is not done beforehand, simply select the imported project in the project explorer and right click and select: *C/C++ Build -> MCU settings* ensure the correct MCU is selected and click **Refresh MCU Cache**.

Building an Example



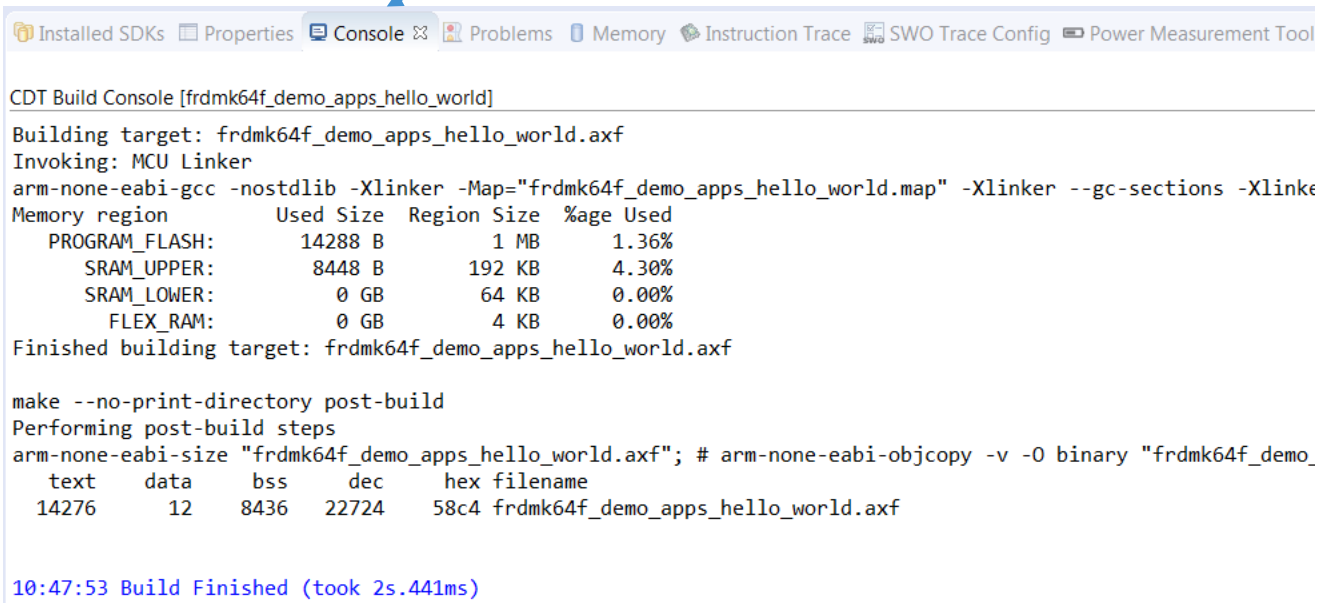
Click project in explorer



Note: After selecting a project, options to build, clean, and debug become active

Click Build

View Build Status in Console



Memory Usage in Build Console

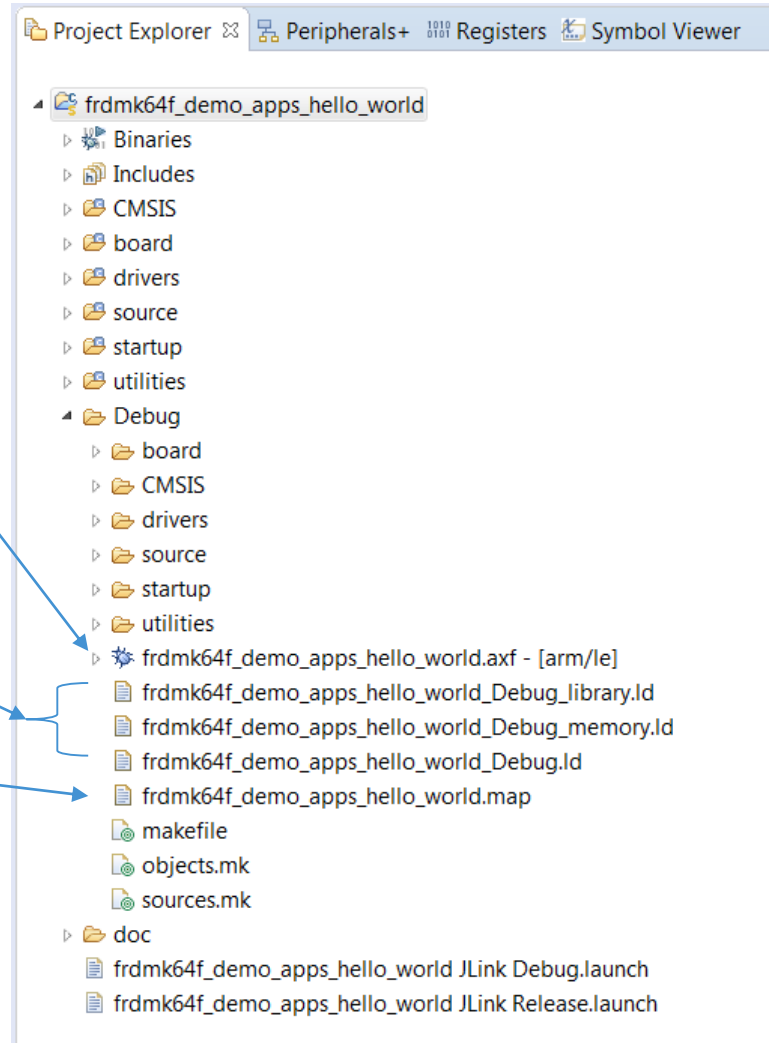
Installed SDKs | Properties | Console | Problems | Memory | Instruction Trace | SWO Trace Config | Power Measurement Tool | Search

```
CDT Build Console [frdmk64f_demo_apps_hello_world]
14:06:12 **** Build of configuration Debug for project frdmk64f_demo_apps_hello_world ****
make -r -j8 all
Building target: frdmk64f_demo_apps_hello_world.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -Xlinker -Map="frdmk64f_demo_apps_hello_world.map" -Xlinker --gc-sections -Xlinker -print-memory-usage
Memory region         Used Size  Region Size  %age Used
   PROGRAM_FLASH:      14288 B         1 MB      1.36%
      SRAM_UPPER:       8448 B       192 KB      4.30%
      SRAM_LOWER:          0 GB        64 KB      0.00%
        FLEX_RAM:          0 GB         4 KB      0.00%
Finished building target: frdmk64f_demo_apps_hello_world.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "frdmk64f_demo_apps_hello_world.axf"; # arm-none-eabi-objcopy -v -O binary "frdmk64f_demo_apps_hello_world.axf"
   text    data     bss     dec     hex filename
  14276      12    8436   22724    58c4 frdmk64f_demo_apps_hello_world.axf


14:06:13 Build Finished (took 1s.80ms)
```

- **text** - shows the code and read-only data in your application (in decimal)
- **data** - shows the read-write data in your application (in decimal)
- **bss** - show the zero initialized ('bss' and 'common') data in your application (in decimal)
- **dec** - total of 'text' + 'data' + 'bss' (in decimal)
- **hex** - hexadecimal equivalent of 'dec'

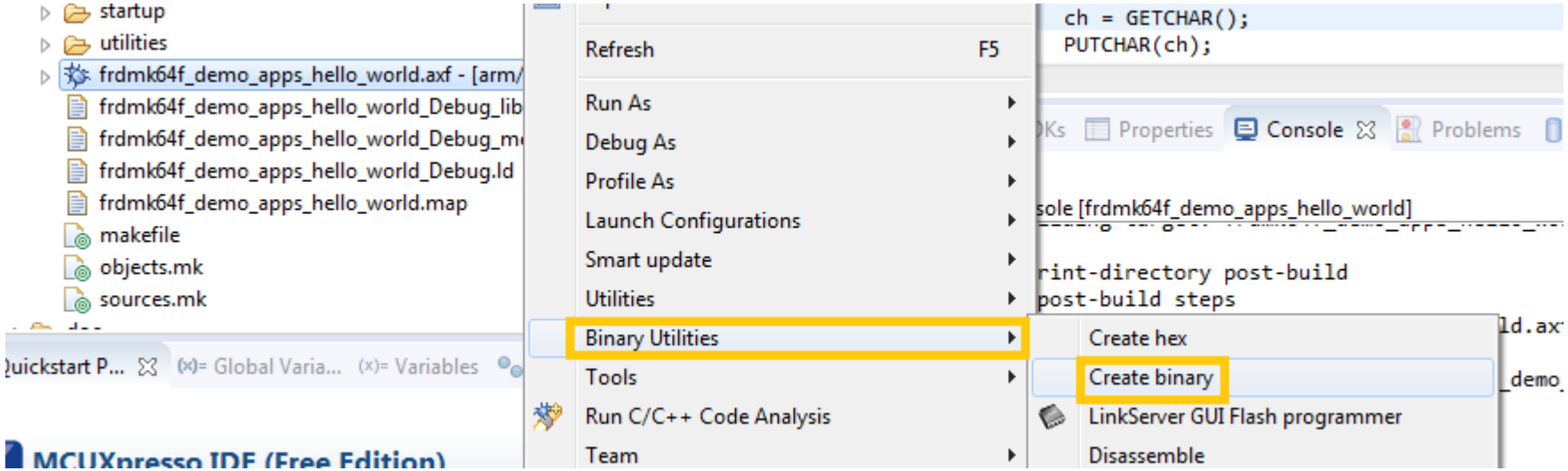
Build Results

- Link step will generate an AXF file
  - Standard ARM Executable Format – ELF/DWARF
  - MCUXpresso IDE can directly download to target
  - Post build step can be used to convert to other formats, such as binary or hex (using arm-none-eabi-objcopy)
- Linker scripts, controlling placement of code and data in memory, generated automatically by IDE
- MAP file generated by linker can be very useful too
  - Shows where code and data has been placed, and sizes of individual sections

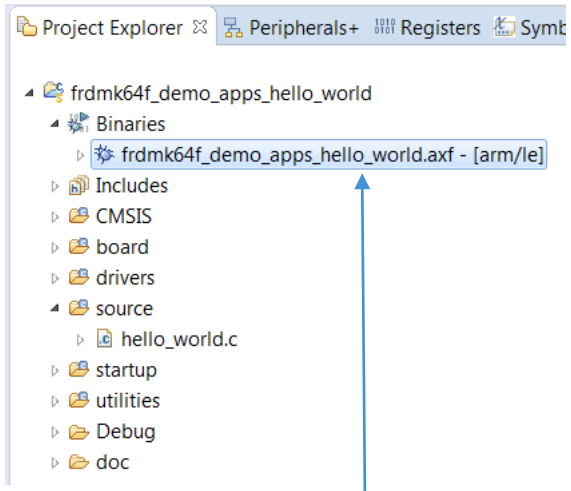


Create Binary

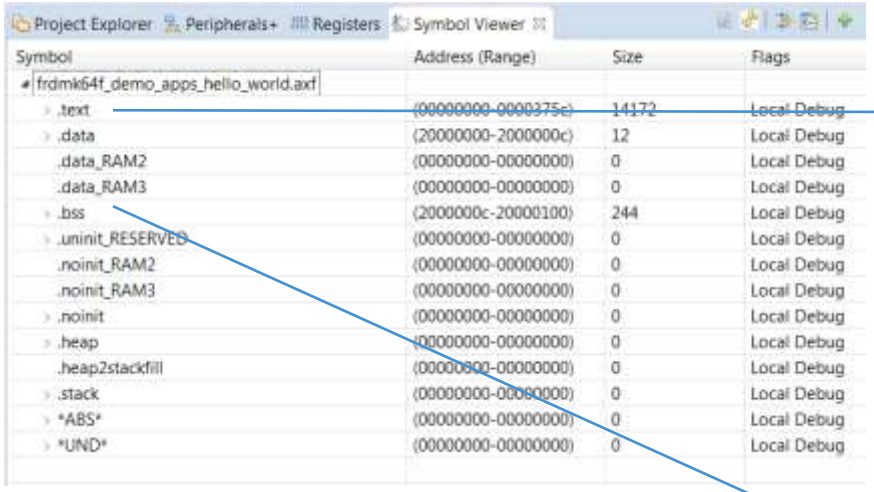
- Useful for drag-and-drop programming via OpenSDA
- Right Click on .axf file: Binary Utilities -> Create Binary



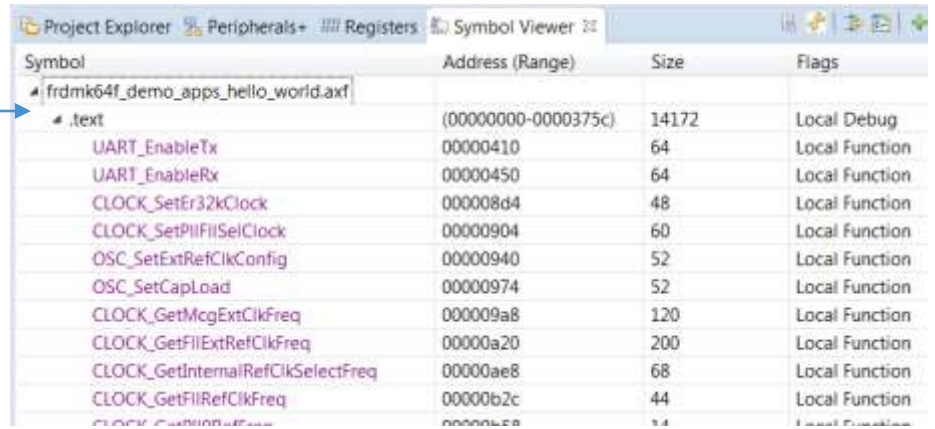
Symbol Viewer

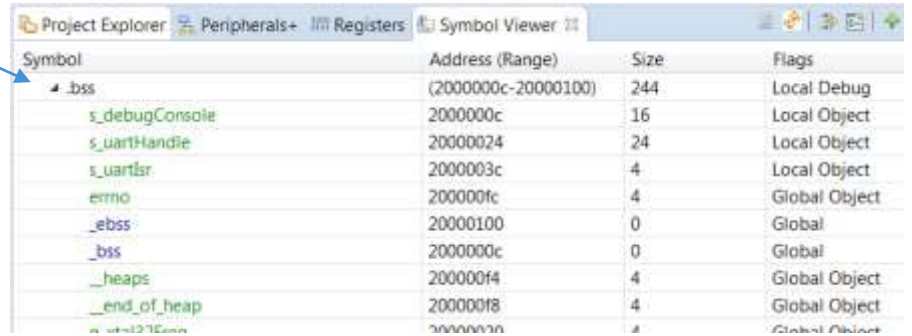


- Right-click .axf file in explorer, then select Tools > View Symbols



- Symbol viewer will move to front of view





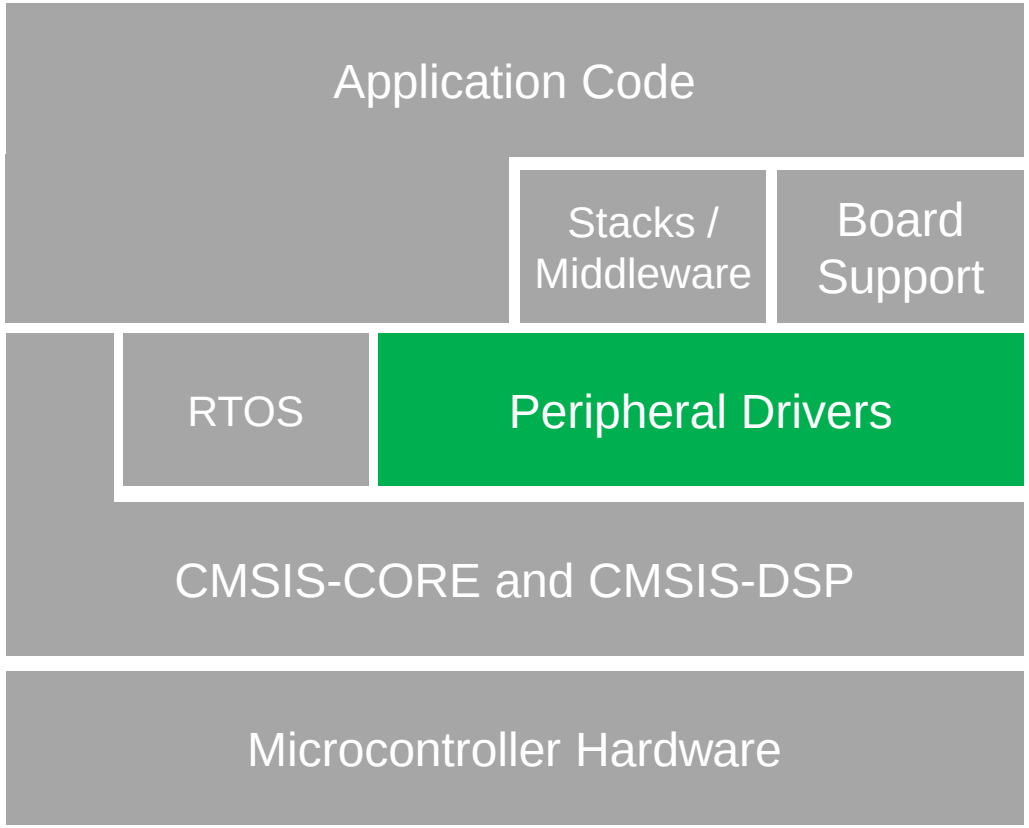
- Expand each section to examine its symbols

AGENDA

- MCUXpresso Software And Tools Overview
- MCUXpresso SDK Introduction
  - Web Builder
  - File Structure
  - Examples
- MCUXpresso SDK Deep Dive
  - Drivers
  - Interrupts
  - Porting
- FreeRTOS
- Conclusion

MCUXpreso SDK Drivers

MCUXpresso SDK – Peripheral Drivers



- MCUXpresso 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

MCUXpresso SDK Drivers

- Designed to simplify the most common driver use-cases
  - Does not cover every peripheral feature.
- The **MCUXpresso SDK API Reference Manual** contains the details of the API.
- Examples found in **<MCUXpresso SDK>\boards\<board_name>\driver_examples** are the best way to learn how to use a particular driver

Low-Power with MCUXpresso SDK

- The SMC drivers allow MCUXpresso 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

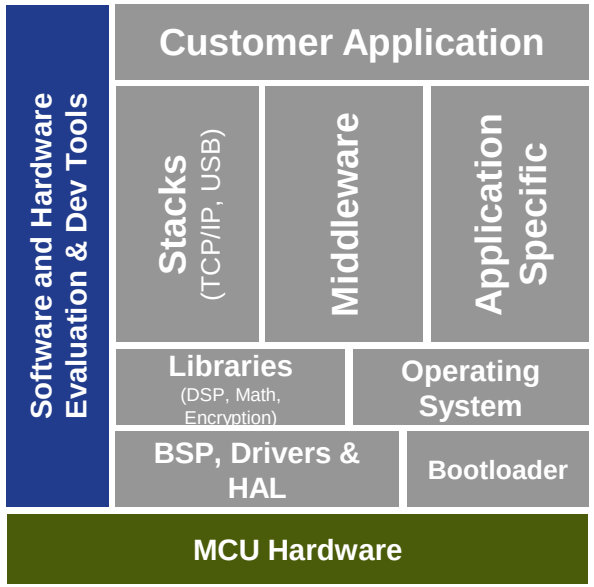
Lab #1

Freedom Development Platforms

Learn more at: www.freescale.com/freedom
www.freescale.com/mbed

Low-cost/low-power development hardware

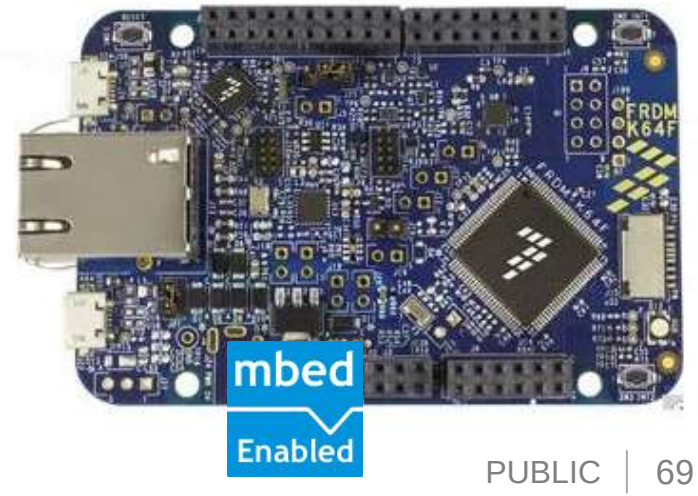
Enables quick application prototyping and demonstration of Kinetis MCU families



Product Features

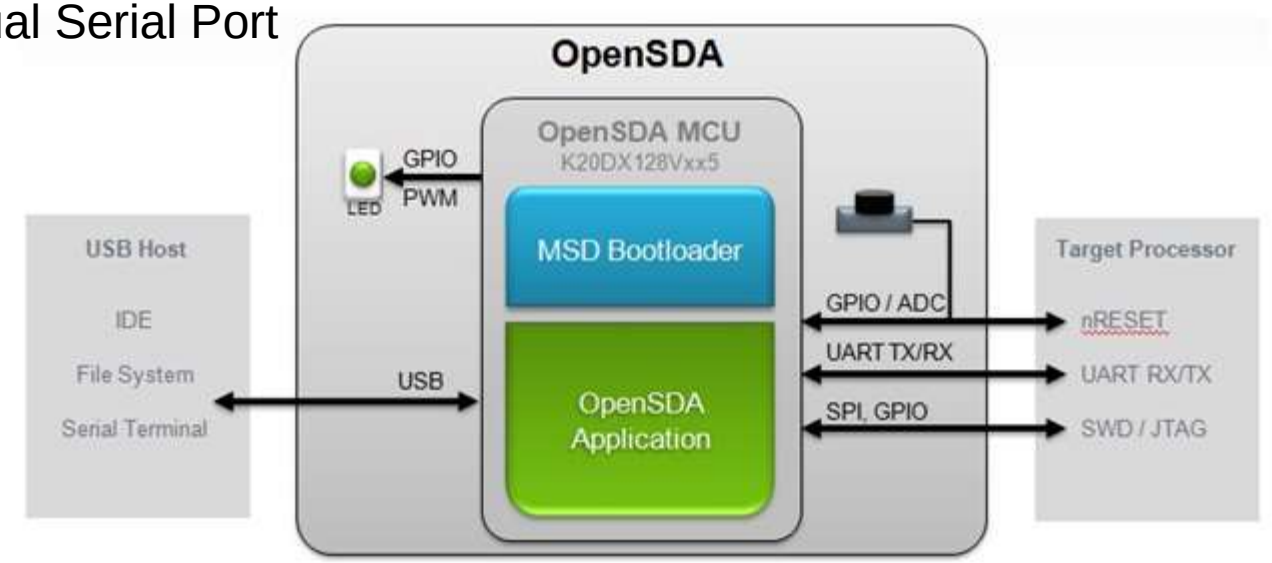
- Low–cost (starting at $12.95 USD)
- Designed in an industry-standard compact form factor (Arduino R3)
- Easy access to the MCU I/O pins
- Integrated open-standard serial and debug interface (OpenSDA)
- Compatible with a rich-set of third-party expansion boards

FRDM-K64F:

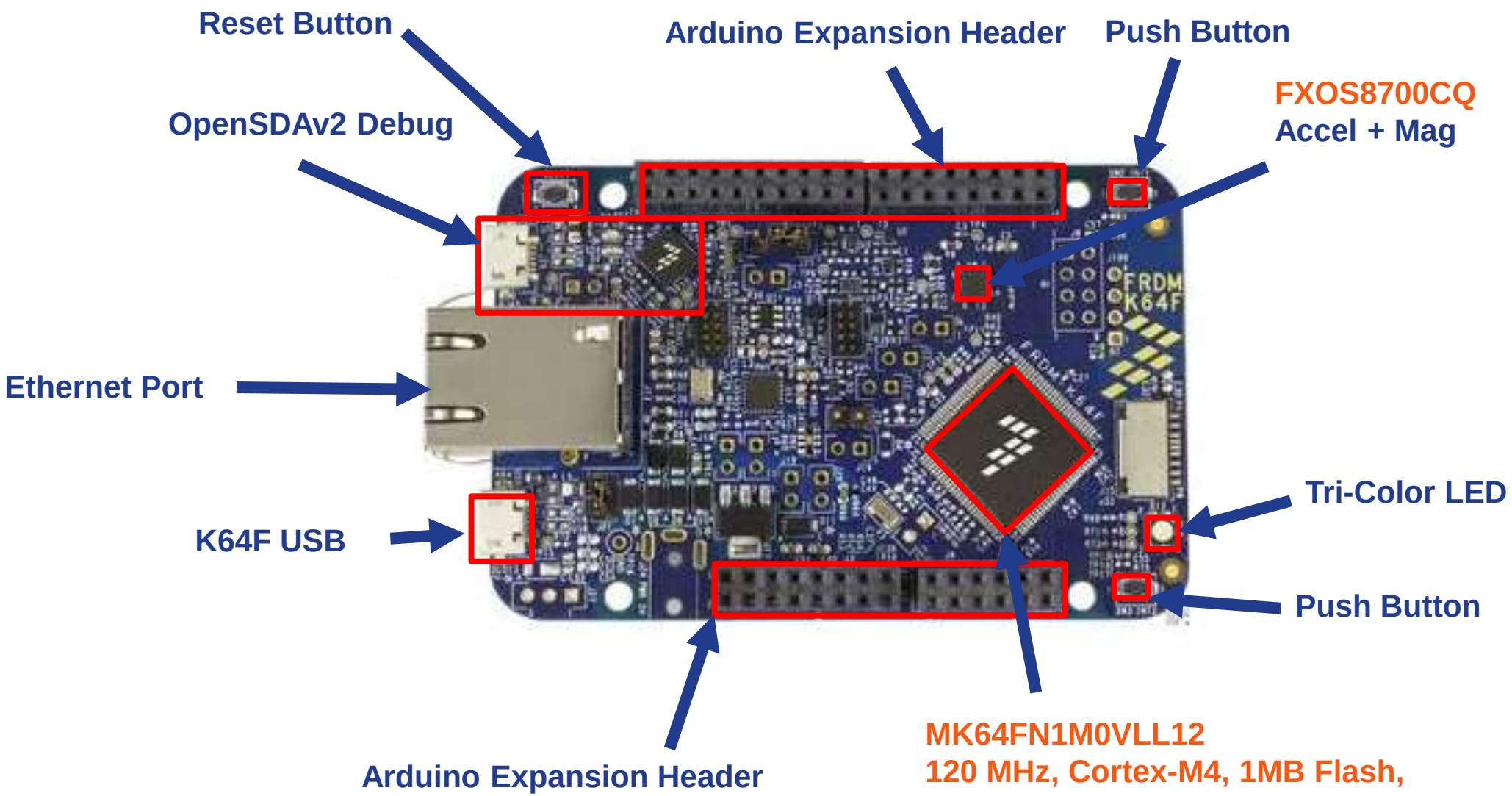


OpenSDA

- OpenSDA is a circuit built into Freescale evaluation boards to provide a bridge between your computer and the embedded target processor
- Purpose is to provide inexpensive debug tool for Freescale evaluation boards
- Different apps can be loaded via a bootloader
- Default CMSIS-DAP app does:
  - Drag-and-drop flashing via a Mass Storage Device
  - Debug via CMSIS-DAP protocol
  - Virtual Serial Port



FRDM-K64F Hardware Overview



Lab 1 Overview

Objective:

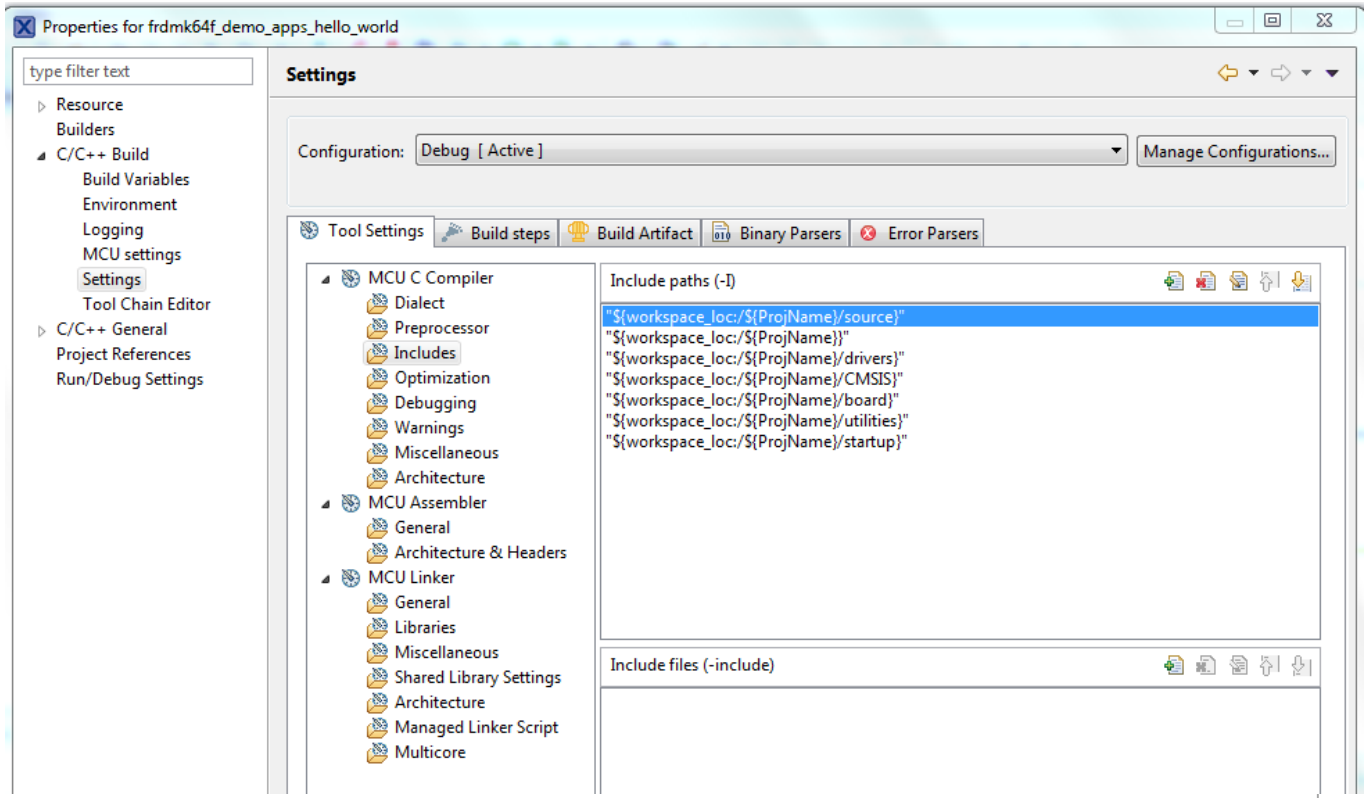
This lab explains how to import and build the demos that are bundled with MCUXpresso SDK and modify it to blink an LED

- **Lab Flow**:
  - Importing demo project
  - Build Demo
  - Download and Debug
  - Modify code to blink LED
  - Explore Project Options

- **Required Hardware and Software**:
  - FRDM-K64F Board configured with CMSIS-DAP Debugger
  - MCUXpresso IDE
  - MCUXpresso SDK
  - Mbed serial driver installed
  - Terminal Program (ie TeraTerm or PuTTY)
  - Micro USB Cable

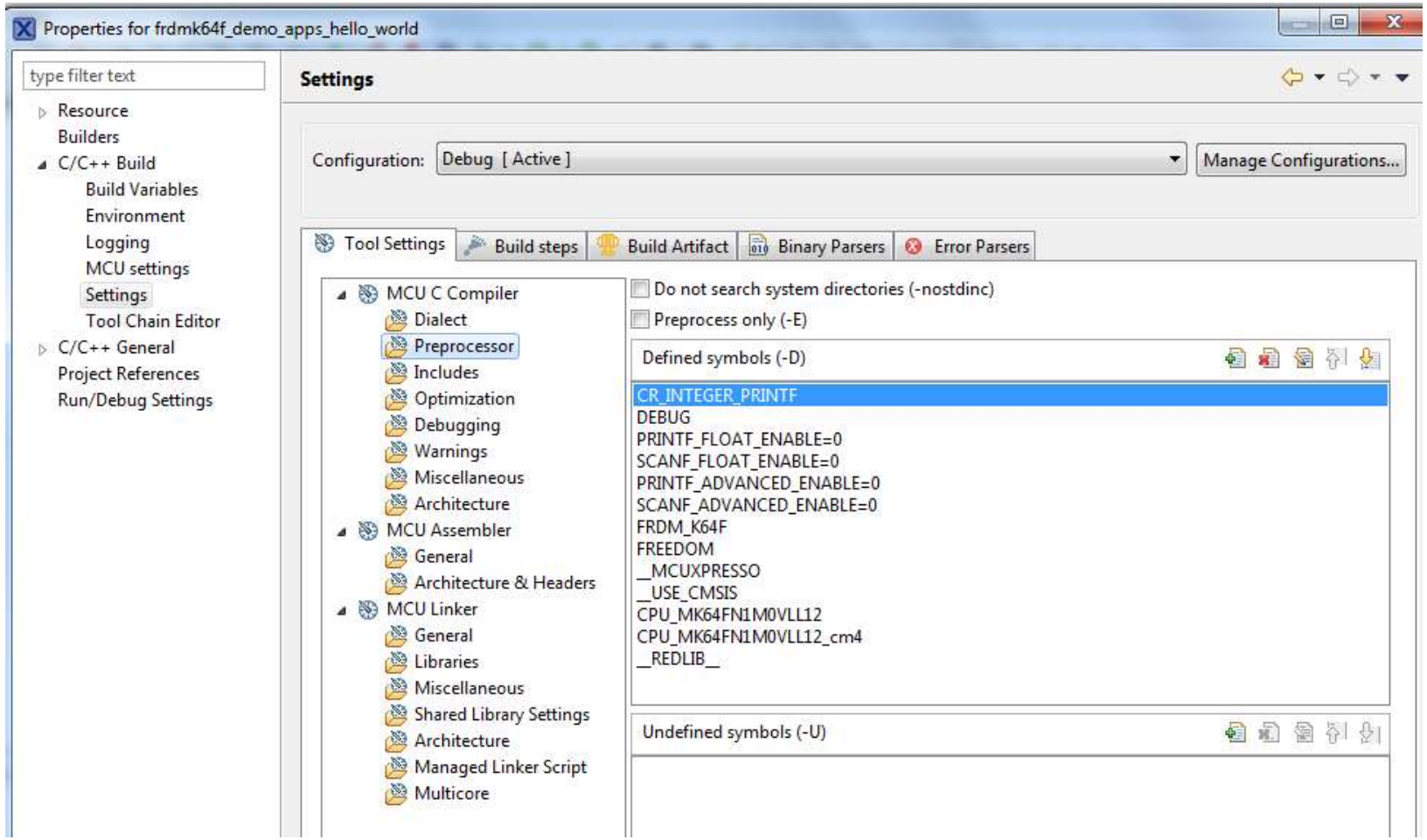
MCUXpresso SDK Project Information

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



MCUXpresso SDK Project Information Continued

- Look at the Preprocessor screen to see the various MCUXpresso SDK defines



Interrupts

Interrupt Handling with MCUXpresso

- Default interrupt handler is a while(1) loop for each IRQ

```
605 WEAK_AV void IntDefaultHandler(void)
606 { while(1) {}
607 }
```

- Redefine the handler for a module in the user application

```
42 #define LPTMR_LED_HANDLER LPTMR0_IRQHandler
```

- Examples of handing interrupts for each module are in driver examples

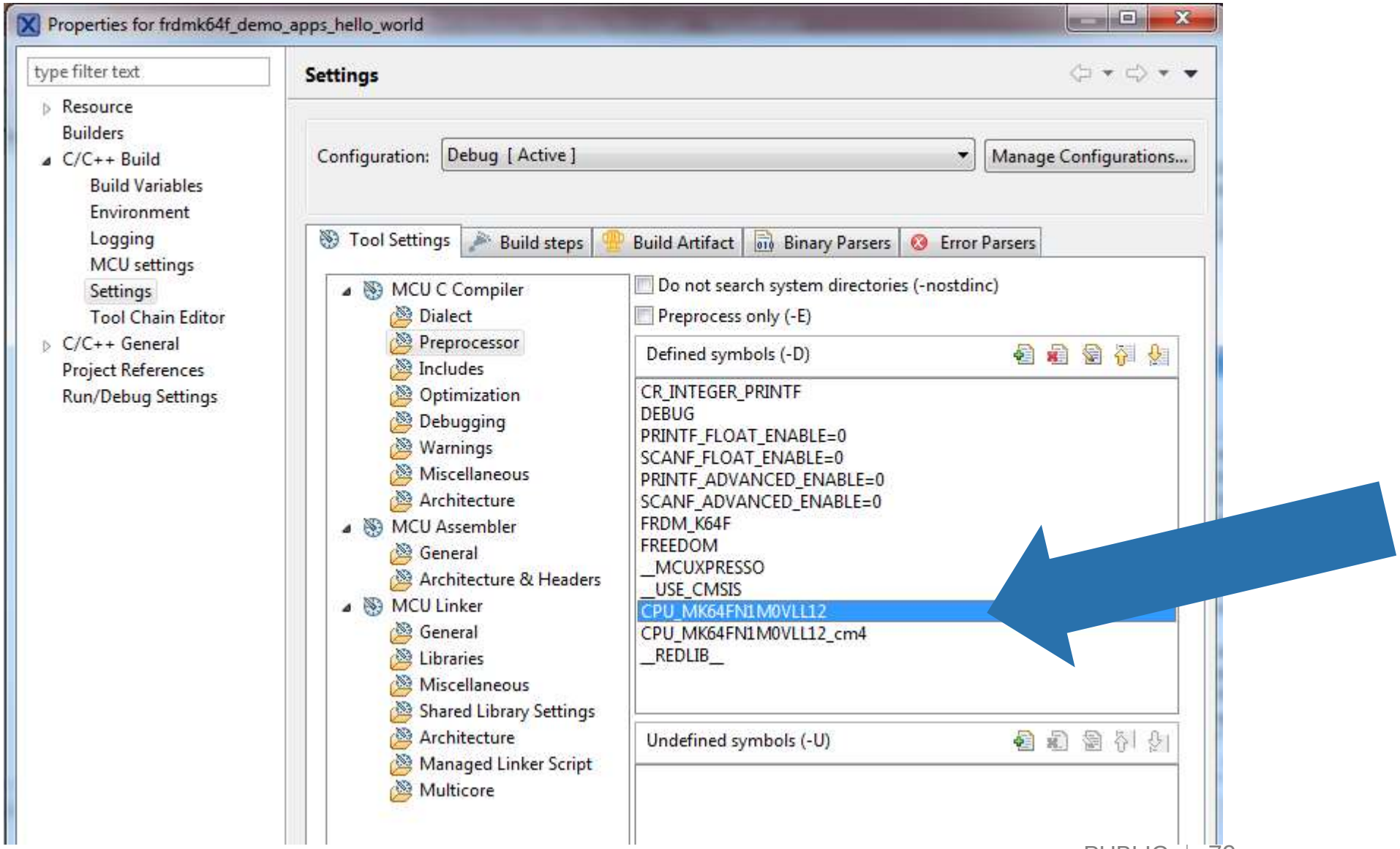
```
63 void LPTMR_LED_HANDLER(void)
64 {
65     LPTMR_ClearStatusFlags(LPTMR0, kLPTMR_TimerCompareFlag);
66     lptmrCounter++;
67     LED_TOGGLE();
```

Porting to new device

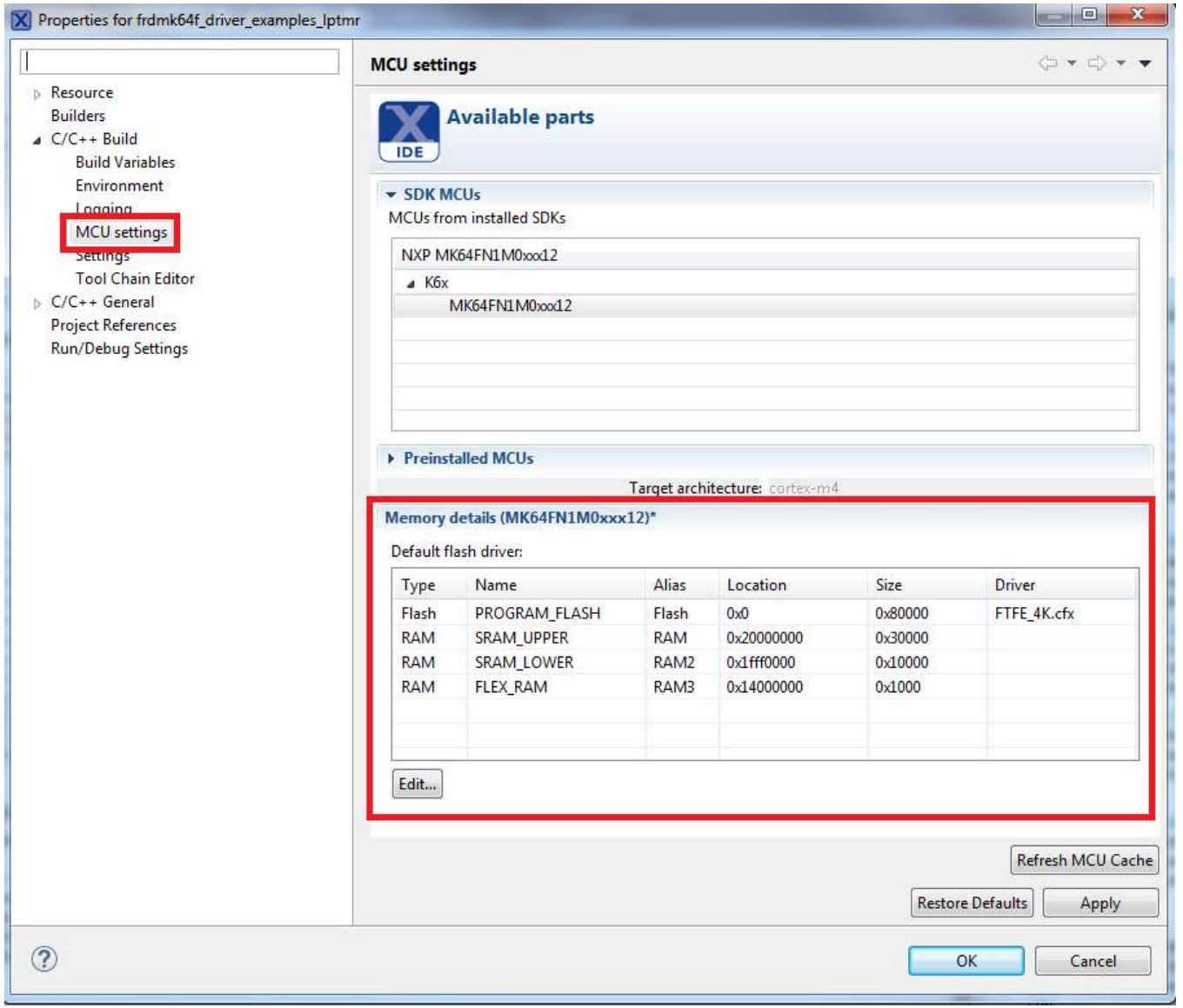
Changing to Kinetis Subset Derivative

- MCUXpresso SDK makes changing to a subset derivative easy
- MCUXpresso SDK already has derivative information in source code
  - Macros used at compile time
- MCUXpresso SDK uses compiler preprocessor definition to specify derivative.
  - Change in project and rebuild

Modify the CPU name



Modify the Flash/RAM size in the project settings



Derivative Details

- The symbol to use for derivative based on Kinetis part number, like CPU_MK64FN1M0VMD12

- Change part number in project settings (Preprocessor and Linker)

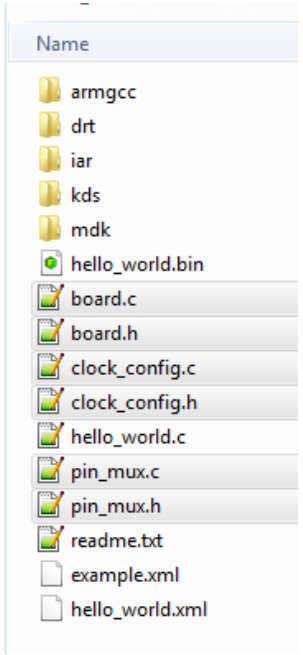
- MCUXpresso SDK already includes supported derivatives
  - Can find all derivative options in <MCUXpressoSDK_PATH> \devices\<device_name>\fsl_device_registers.h

- Porting to a new family is not supported. Only derivatives.
  - Full list of supported derivatives can be found in the Release Notes

Porting to custom hardware

Custom Board Configuration

- Each example in MCUXpresso SDK has board configuration files
- Found in <MCUXpressoSDK_PATH>\boards\<board_name>\demo_apps\<demo_name>
  - Contains board-specific details for MCUXpresso 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();
```

board.h file

- Defines debug UART peripheral and pins
  - For stdin/stdout functions, like printf()
- Mainly used for MCUXpresso 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

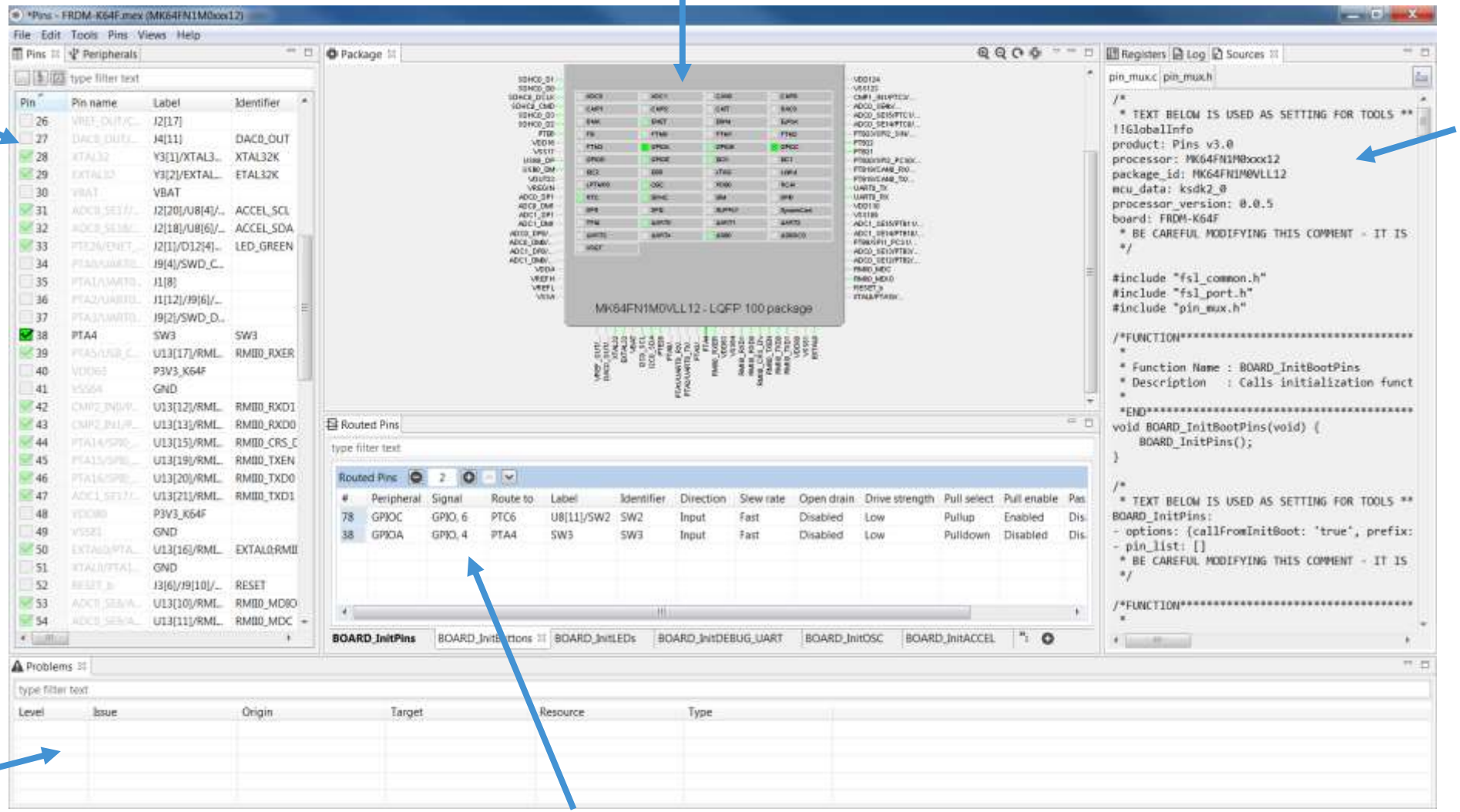
- Kinetis devices provide great flexibility in muxing signals
  - Each digital port pin has up to 8 signals muxed on pin
  - Some peripherals route same signals to multiple pins
- BOARD_InitPins() inside pin_mux.c is used to initialize pins
  - Used to set pin mux options for all pins used on board

10.3.1 K64 Signal Multiplexing and Pin Assignments

144 LQFP	144 MAP BGA	121 XFBGA	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

MCUXpresso Config Tools - Pins



GPIO Driver Uses Those 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

MCUXpresso 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

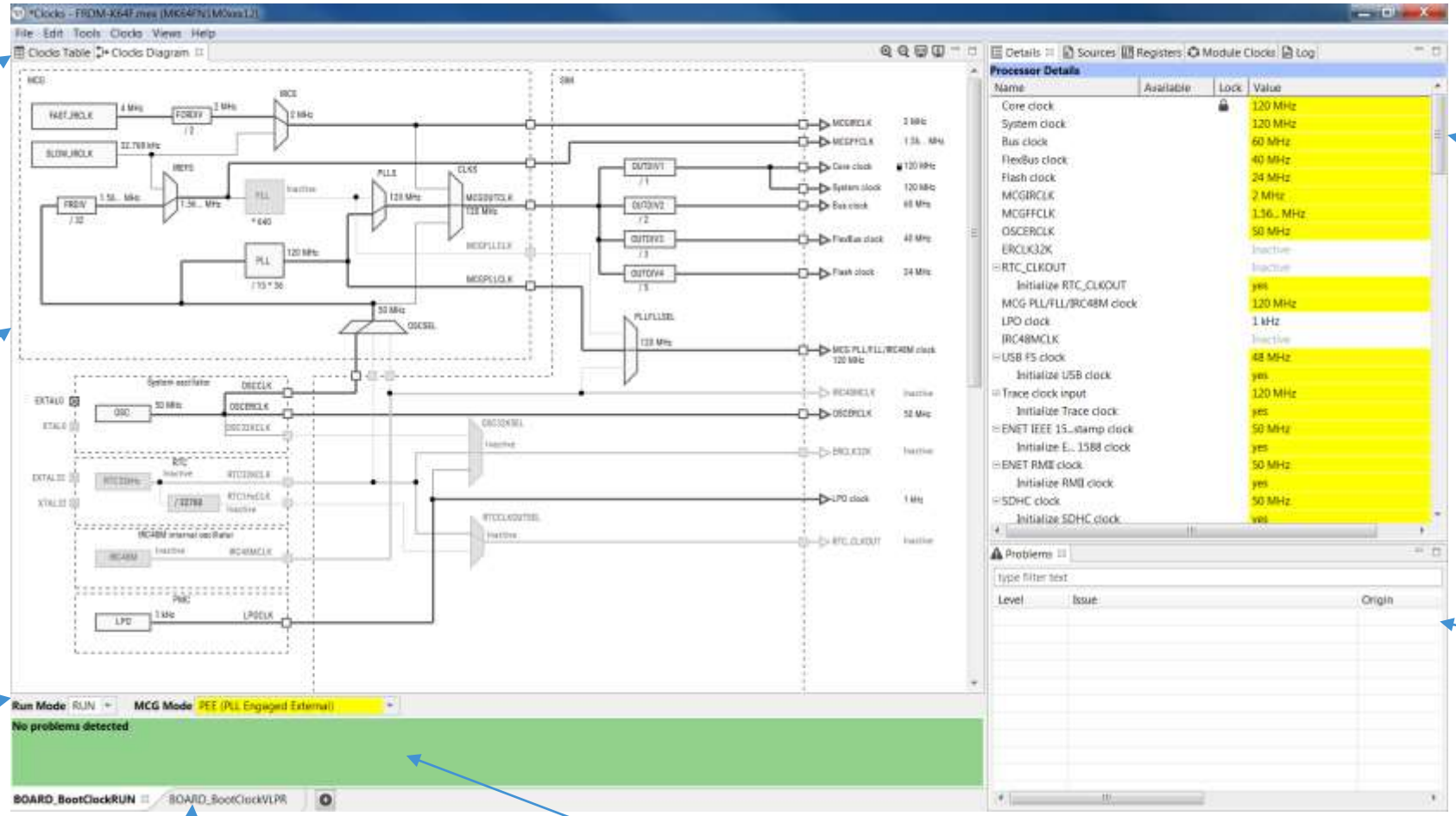
```
72   PORT_SetPinMux(PORTB, PIN16_IDX, kPORT_MuxAlt3);          /* PORTB16 (pin 62) is configured as UART0_RX */
73   PORT_SetPinMux(PORTB, PIN17_IDX, kPORT_MuxAlt3);          /* PORTB17 (pin 63) is configured as UART0_TX */
```

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.

```
187 /*******************************************************************************
188  * Code for BOARD_BootClockRUN configuration
189  ******************************************************************************/
190 void BOARD_BootClockRUN(void)
191 {
192     /* Set the system clock dividers in SIM to safe value. */
193     CLOCK_SetSimSafeDivs();
194     /* Initializes OSC0 according to board configuration. */
195     CLOCK_InitOsc0(&oscConfig_BOARD_BootClockRUN);
196     CLOCK_SetXtal0Freq(oscConfig_BOARD_BootClockRUN.freq);
197     /* Configure the Internal Reference clock (MCGIRCLK). */
198     CLOCK_SetInternalRefClkConfig(mcgConfig_BOARD_BootClockRUN.irclkEnableMode,
199                                   mcgConfig_BOARD_BootClockRUN.ircs,
200                                   mcgConfig_BOARD_BootClockRUN.fcrdiv);
201     /* Configure FLL external reference divider (FRDIV). */
202     CLOCK_CONFIG_SetFllExtRefDiv(mcgConfig_BOARD_BootClockRUN.frdiv);
203     /* Set MCG to PEE mode. */
204     CLOCK_BootToPeeMode(mcgConfig_BOARD_BootClockRUN.oscsel,
205                         kMCG_PllClkSelPll0,
206                         &mcgConfig_BOARD_BootClockRUN.pll0Config);
207     /* Set the clock configuration in SIM module. */
208     CLOCK_SetSimConfig(&simConfig_BOARD_BootClockRUN);
209     /* Set SystemCoreClock variable. */
```

MCUXpresso Config Tools - Clocks



USB Hardware Porting

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

```
59  /* USB clock source and frequency*/
60  #define USB_FS_CLK_SRC kCLOCK_UsbSrcIrc48M
61  #define USB_FS_CLK_FREQ 48000000U
```

```
539     CLOCK_EnableUsbfs0Clock(USB_FS_CLK_SRC, USB_FS_CLK_FREQ);
```

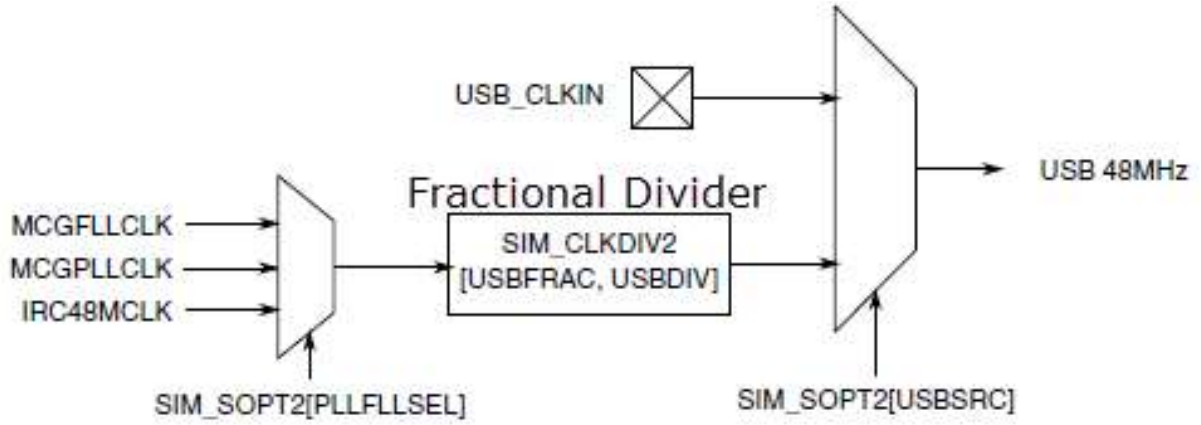
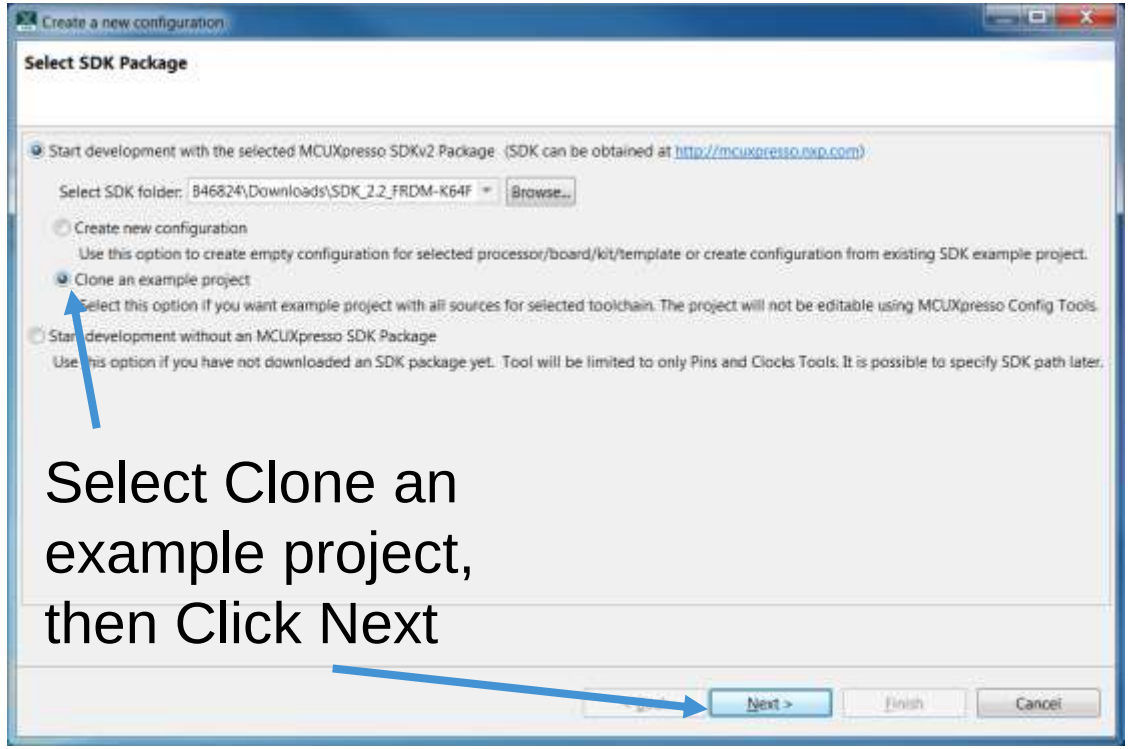
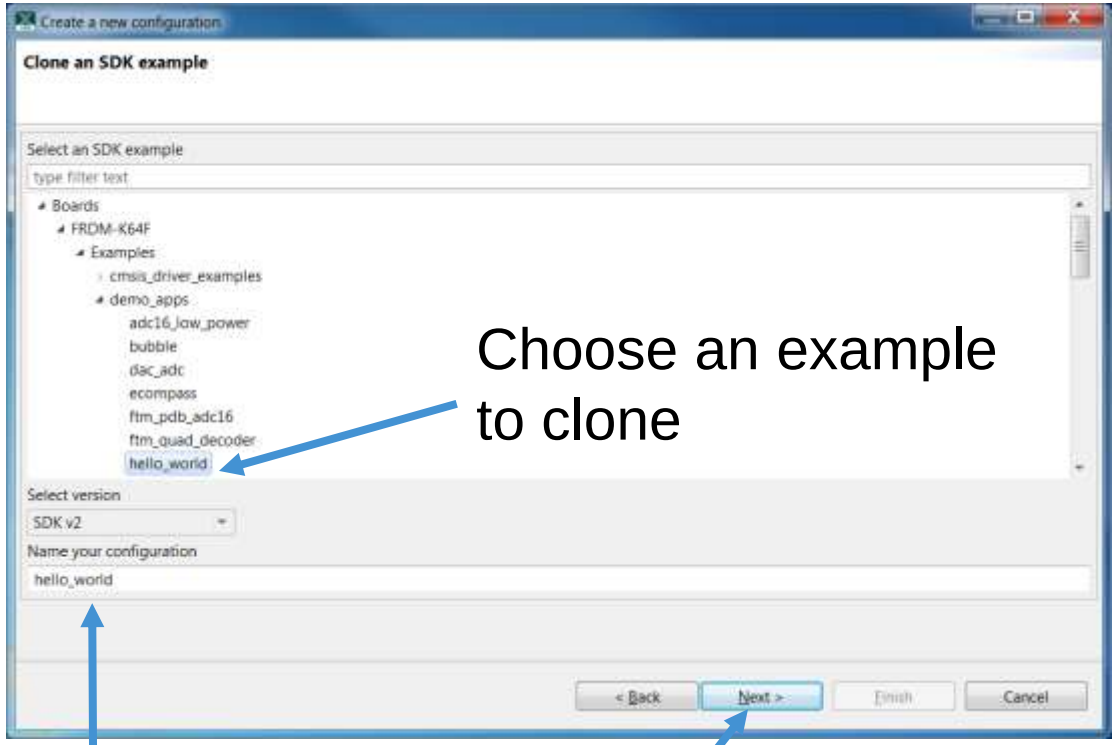


Figure 5-7. USB 48 MHz clock source

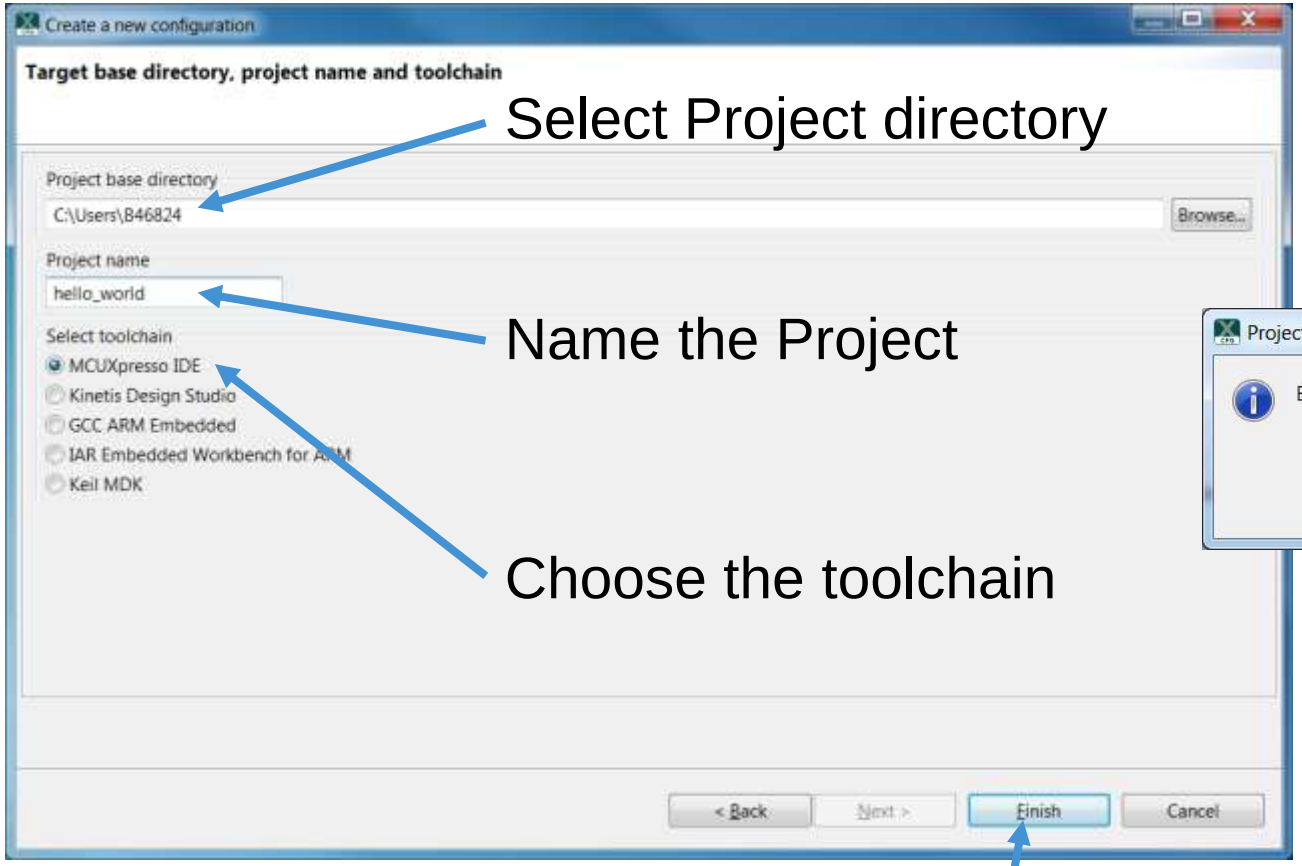
Clone a project

Clone an Example for non-MCUXpresso IDE





Clone Project

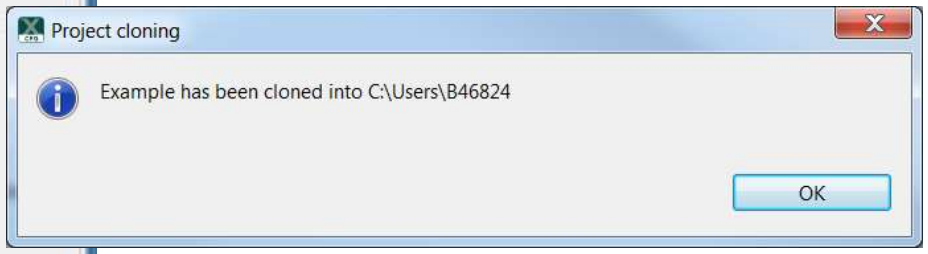


Select Project directory

Name the Project

Choose the toolchain

Click Finish



Once finished, dialog box will confirm cloning has completed

Lab #2

Lab 2 Overview

Objective:
Modify the project to use a push button switch to trigger an interrupt

- **Lab Flow**:
  - Modify code to enable interrupts on a GPIO pin.
  - Expand project to turn on different colors depending on the switch pressed

- **Required Hardware and Software**:
  - FRDM-K64F Board configured with CMSIS-DAP Debugger
  - MCUXpresso IDE
  - MCUXpresso SDK
  - Mbed serial driver installed
  - Terminal Program (ie TeraTerm or PuTTY)
  - Micro USB Cable

AGENDA

- MCUXpresso Software And Tools Overview
- MCUXpresso SDK Introduction
  - Web Builder
  - File Structure
  - Examples
- MCUXpresso SDK Deep Dive
  - Drivers
  - Interrupts
  - Porting
- FreeRTOS
- Conclusion

FreeRTOS Overview

FreeRTOS

“It is the fate of operating systems to become free.”

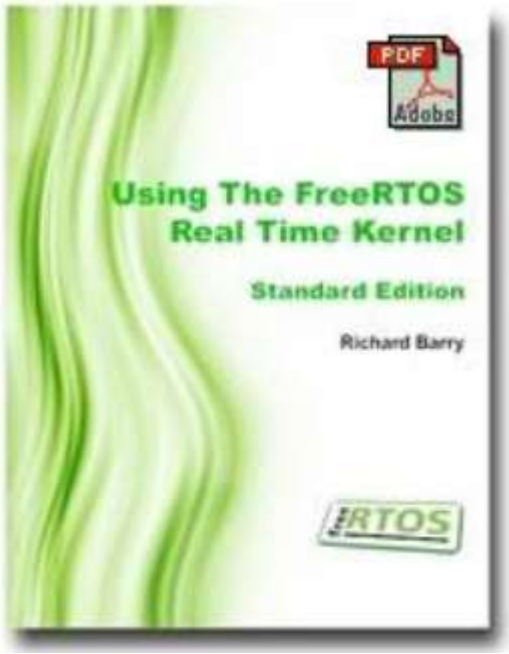
— Neal Stephenson

Outline

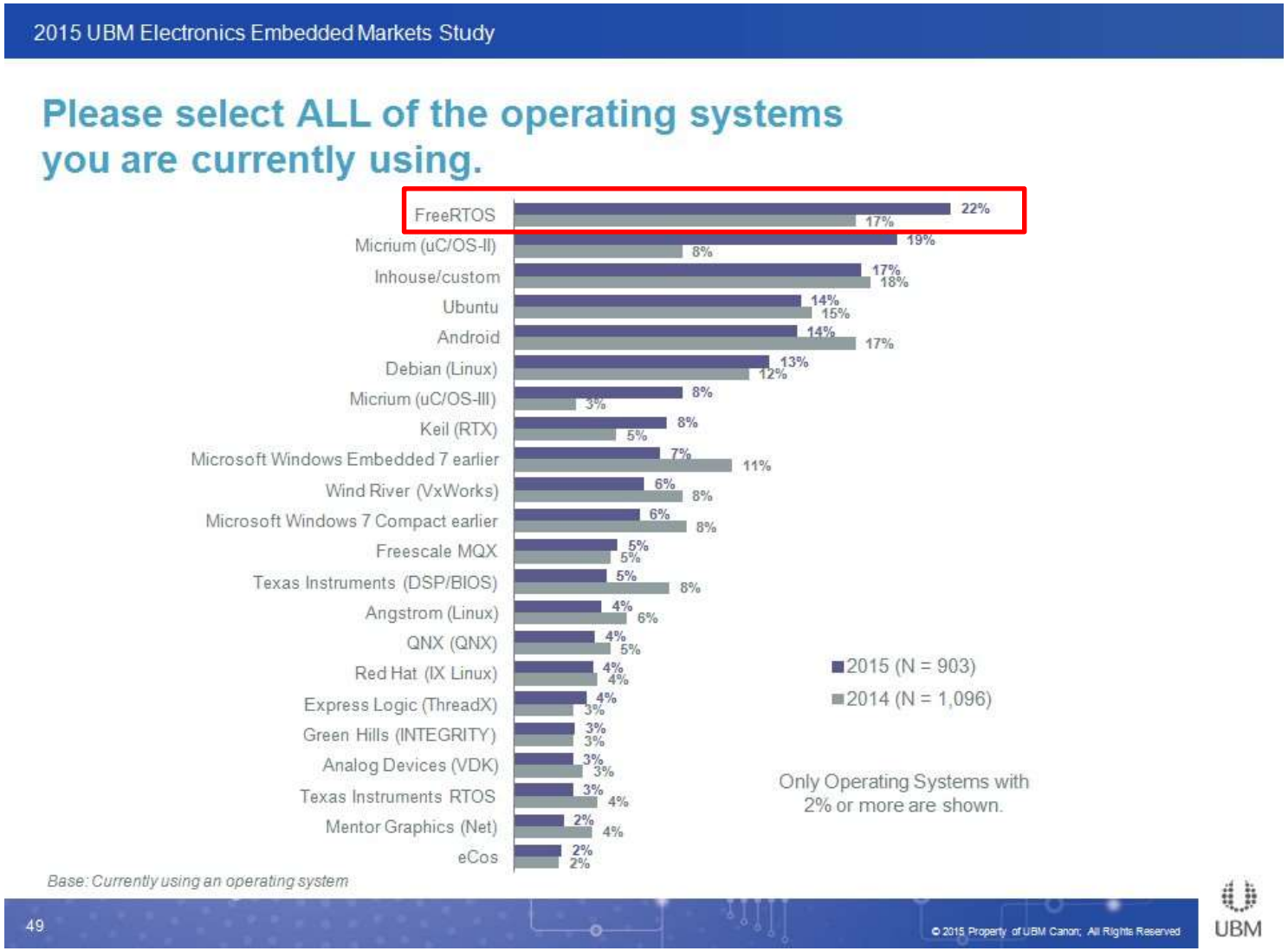
- **Introduction**
  - Licensing, distributions
  - Software and tools, MCUXpresso SDK
- **Architecture and Features**:
  - Sources, kernel, interrupts
  - Low Power, Tickless Idle Mode
- **Visualization**
  - Kernel awareness and trace

FreeRTOS

- http://www.freertos.org
- Maintained by Real Time Engineers Ltd., London (Richard Barry)
- Open Source, free-of-charge, royalty free
- >35 architectures, >113’000 downloads in 2015
- Portable, simple to learn and use
- Ecosystem and commercial supported ports available
  - **OpenRTOS**: commercial supported version
  - **SafeRTOS**: special version dedicated to safety critical systems
  - Sold Reference Manual/Tutorial Book

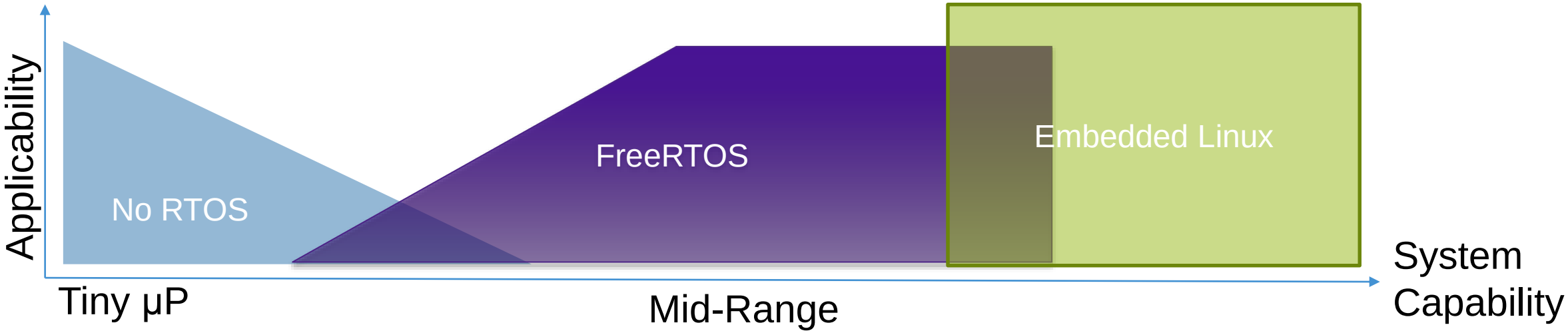


UBM Embedded Market Study



Positioning

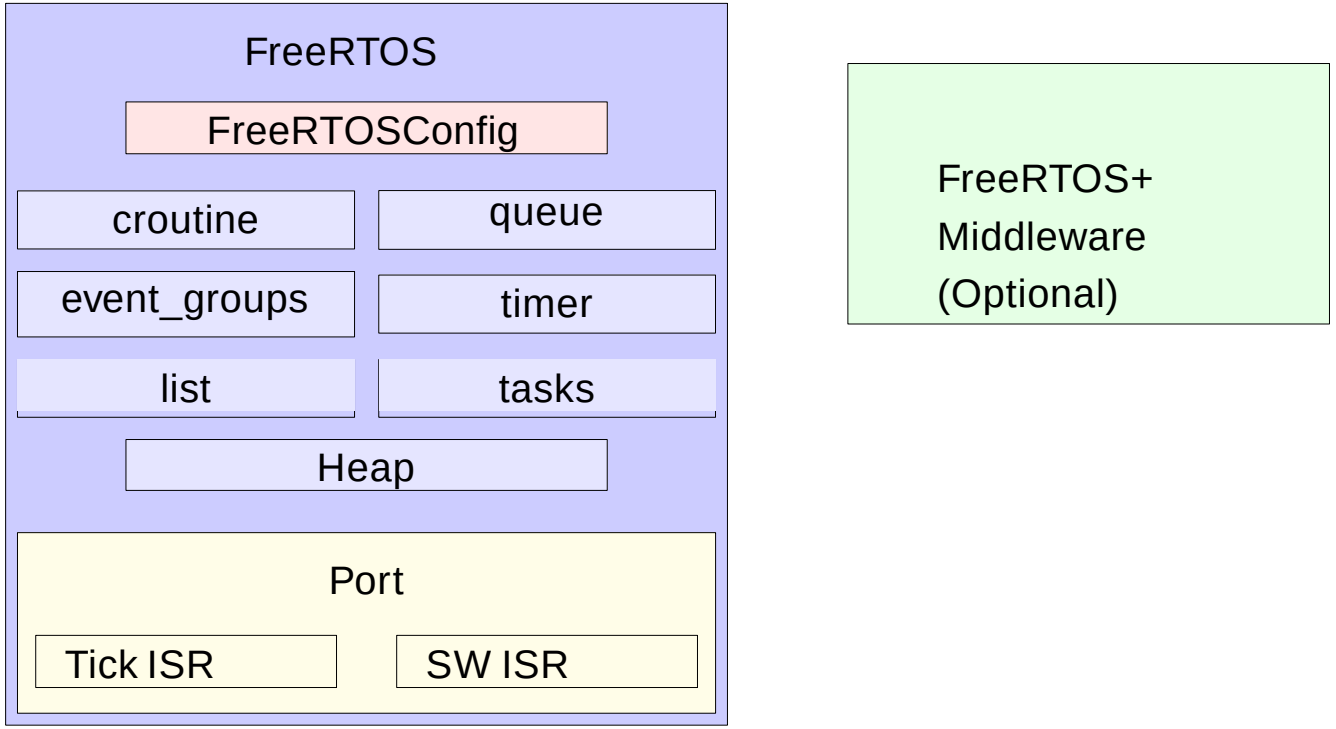
- Moderated Open Source
- True Realtime behavior
- Portable and easy to understand, very few source files
- Scalable: from 64bit down to 8bit microcontrollers, > 32 ports available
- Small: 5-15 kByte FLASH, 250 Bytes RAM (plus space for stacks)



Philosophy: Kernel

- Small Kernel, **implemented in** C, compiled and linked with application
- Single port M0+ and M4(F) each (differences compiler/assembly)
- Kernel configuration with `#define` in **`FreeRTOSConfig.h`**
- Preemptive or cooperative scheduler mode (at compile time)
- Kernel only needs **tick interrupt** and **software interrupt**
- Timing based on tick interrupt (timing resolution), max 1 kHz
- RTOS always creates and runs in IDLE task
- Scheduler variables and task stack in dynamic memory (**heap**)
- Different selection of heap allocation (**schemes**)
- Multiple tasks with same priority
- ’Official’ and ’Contributed’ Examples and ports

Block Diagram



- Only 10 source plus header files for co-routine, queue, event, timer, list and tasks
- FreeRTOS Configuration Header File
- 5 different heap implementations
- Port depending on architecture and tool chain and compiler
- FreeRTOS+: different license

FreeRTOS+

Internet of Things New cloud managed service for peer to peer communication over the Internet of Things (IoT)	**File Systems** A choice of file systems, from a tiny footprint FAT file system, to a fault tolerant transactional file system	**TCP/IP** Thread aware, low footprint, sockets based, efficient TCP/IP stack and related protocols
Trace Get 20+ graphically interlinked views of the trace, providing an unprecedented level of insight	**UDP/IP** Thread aware, sockets based, efficient UDP/IP stack –ideal for communication between small embedded devices	**CLI** Enable your application to efficiently process command line input
Safety & Certification A safety certified kernel for microcontrollers, with a similar usage model to FreeRTOS	**SSL and TLS** State of the art networking security for embedded systems	**Complete BSP Solutions** Driver and middleware packages provided by third parties to the FreeRTOS community
RTOS Training Expert instructor led RTOS training to maximize productivity – delivered online or on site	**I/O** Add an open(), read(), write(), ioctl() peripheral interface to your application	**FreeRTOS Labs** The place where future FreeRTOS products come to incubate.

Architecture

- Micro Realtime Kernel
  - OS runs as lowest priority task
- Scheduling policies:
  - **Preemptive**
  - **Cooperative**
- Configured at compile time using `FreeRTOSConfig.h`
- Uses different heap schemes for memory allocation (task stacks, internal data, queues, etc.)
- Implemented in C language
- Target architectural dependencies are in ’port’ files
  - C and Assembly
  - Interrupt and tick handling

Services and Features

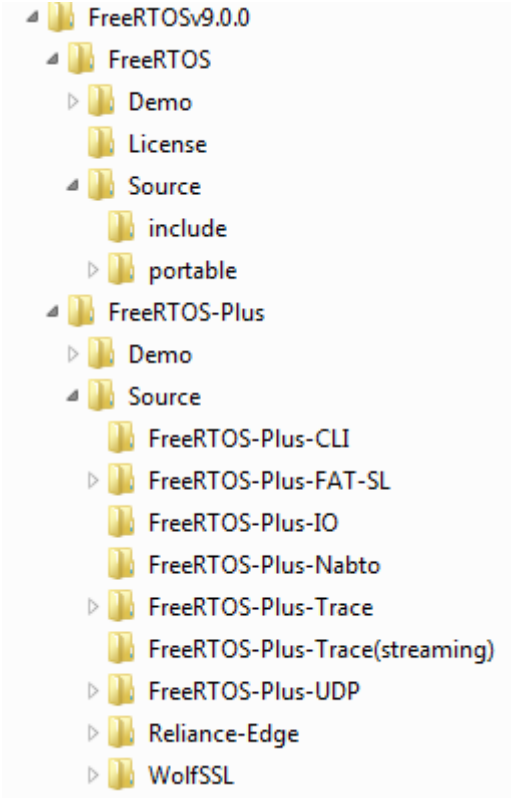
- Scheduler
- Tasks with multiple priority lists
- Dynamic memory (heap)
- Coroutines
- Message queue
- Software timer
- Semaphore and Mutex
- Event set
- Direct task notification
- Thread Local Storage pointers
- Low Power and Tickless idle mode
- Trace

Licensing

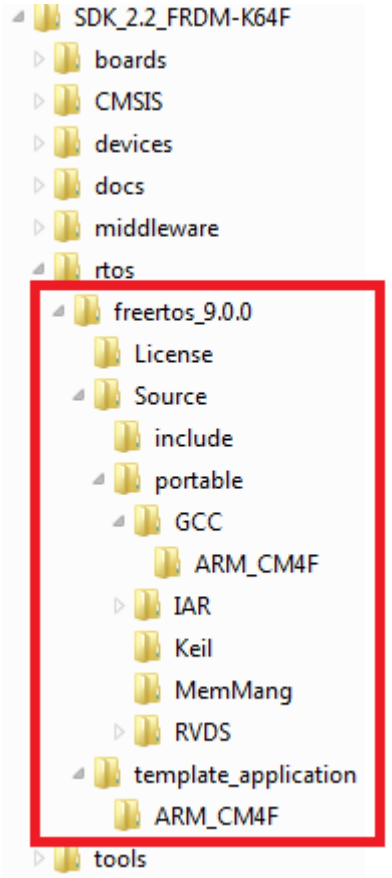
- Terms on www.freertos.org/a00114.html
- Open Source, Free of charge, no royalties
- *The modification to the GPL is included to allow you to* ***distribute a combined work*** *that includes FreeRTOS* ***without being obliged to provide the source code*** *for proprietary components*
- Community support: sourceforge.net/projects/freertos
- Can be used in commercial applications, need to contribute back changes in the **kernel**
- If source code is published, FreeRTOS needs to be published too
- Not allowed to publish benchmarks results without permission
- **Different** licensing terms for OpenRTOS, SafeRTOS and FreeRTOS+ parts

Distribution

FreeRTOS.org

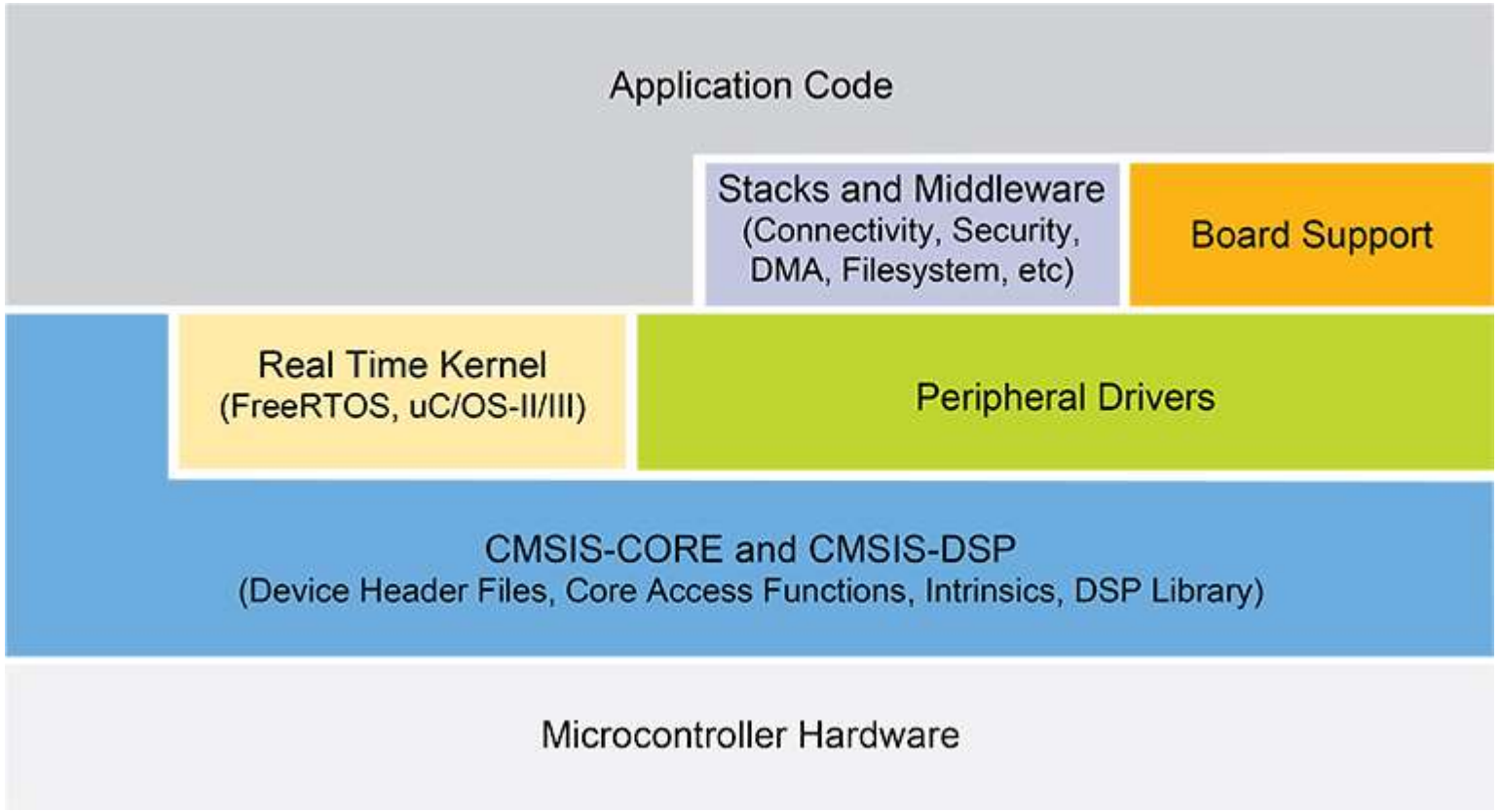


MCUXpresso SDK



1. Default distribution: http://www.freertos.org
2. Integrated into NXP MCUXpresso SDK

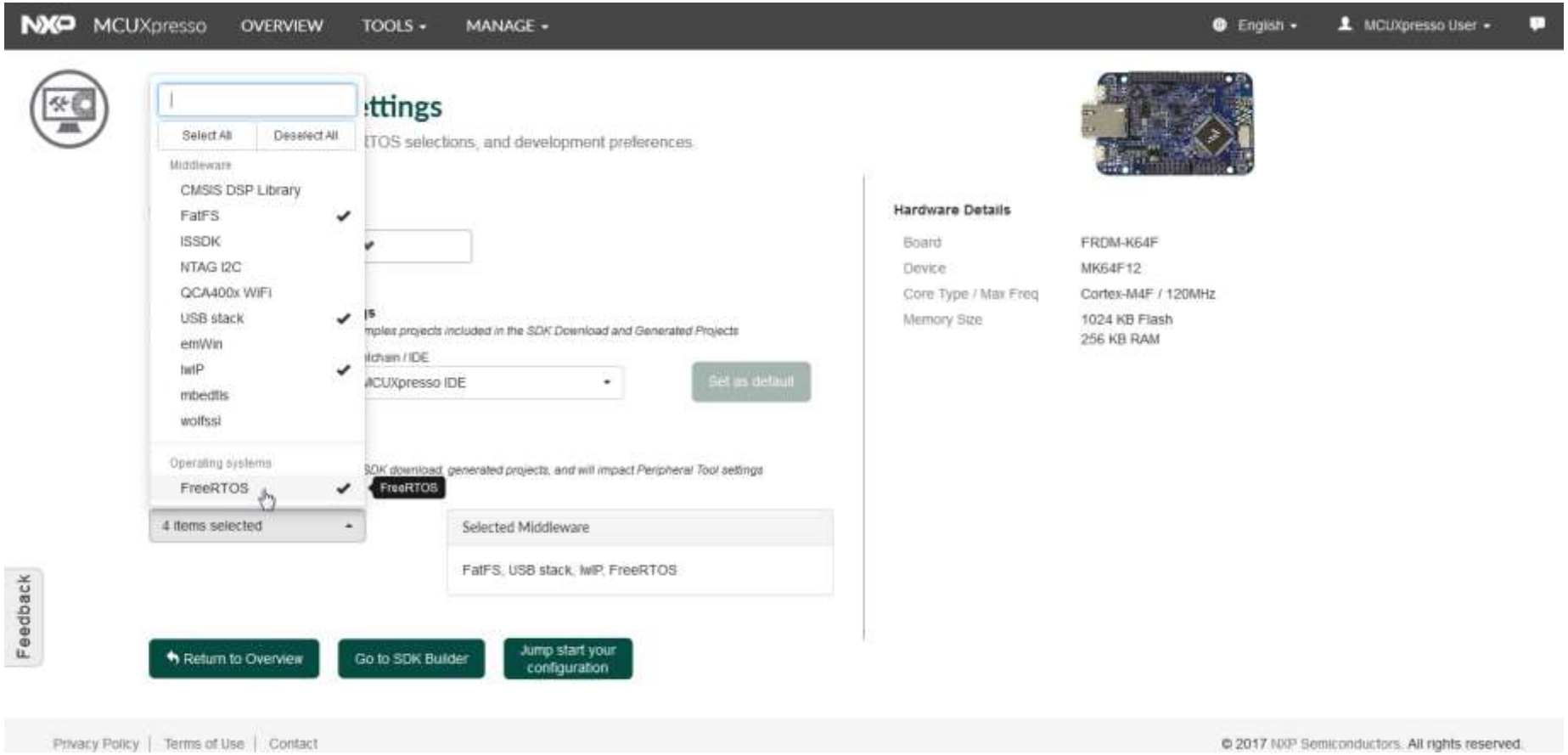
MCUXpresso SDK Block Diagram



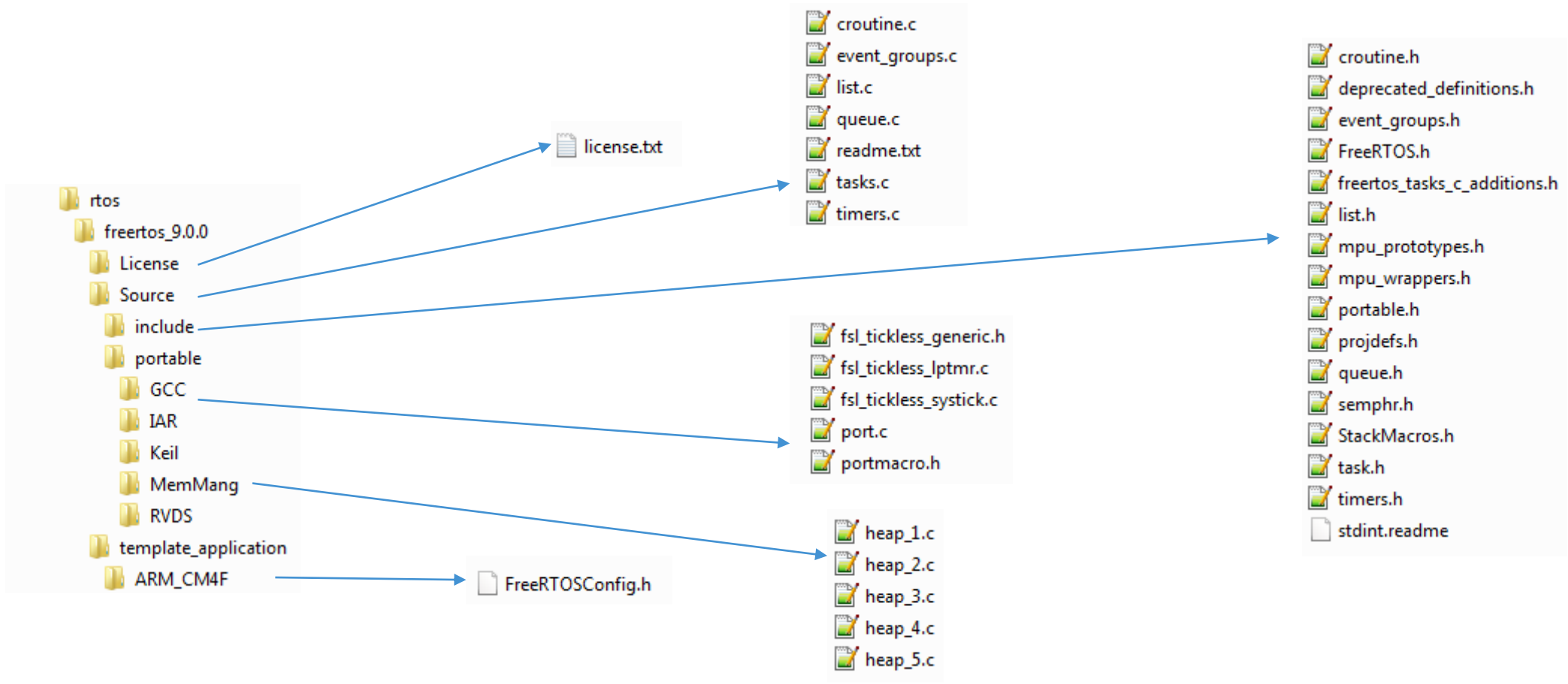
- No need for dedicated RTOS AL and HAL
- FreeRTOS directory structure matching freertos.org

NXP MCUXpresso SDK Builder

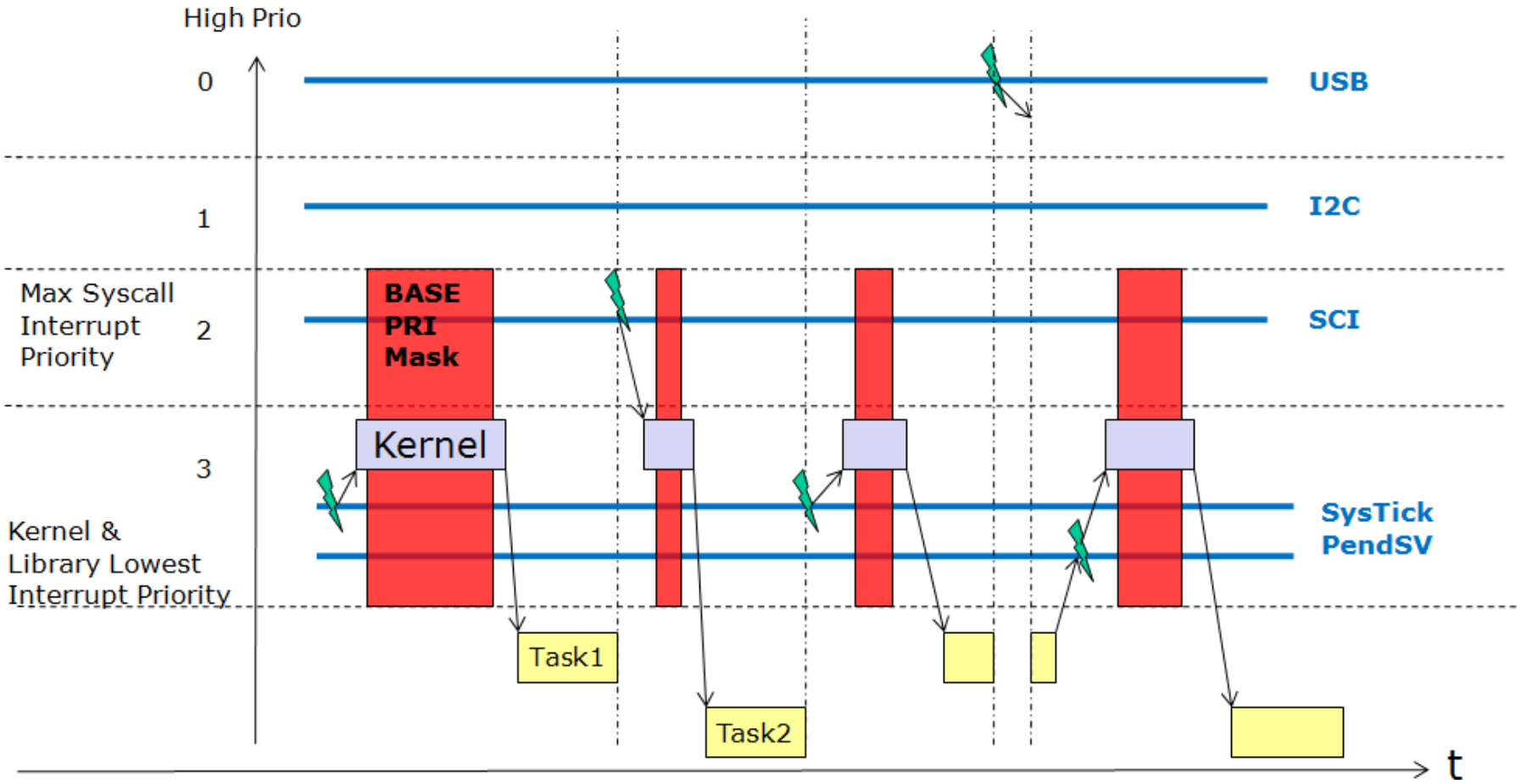
- Cloud based creation of SDK Package on demand with FreeRTOS enabled



FreeRTOS Files in MCUXpresso SDK



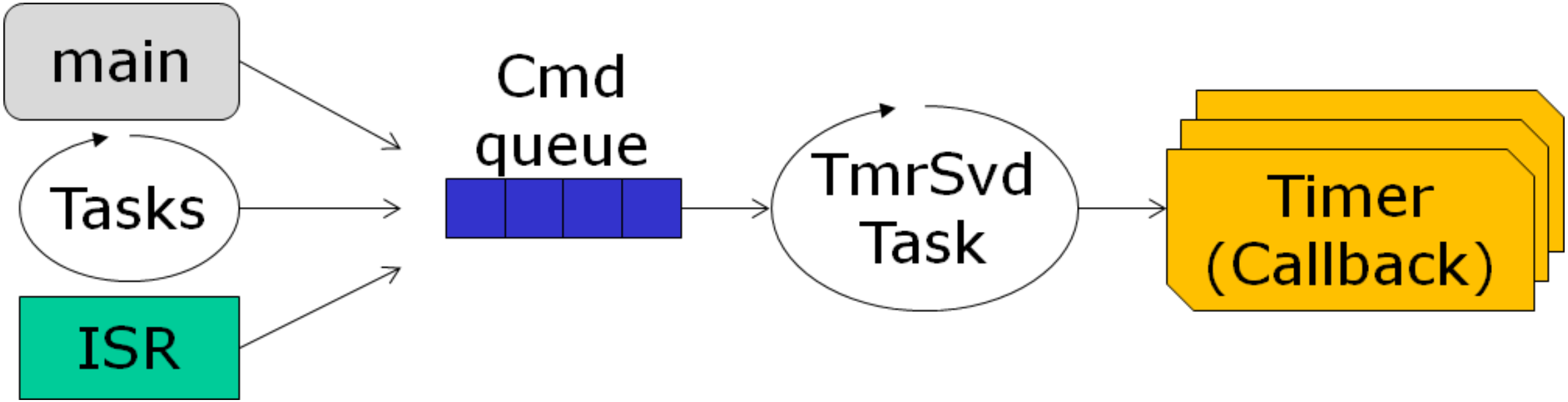
FreeRTOS and ARM Cortex-M Interrupts



- Cortex-M0+: Kernel disables global interrupts
- Cortex-M4: using `PRIMASK` to mask interrupts
  - `configMAX_SYSCALL_INTERRUPT_PRIORITY`

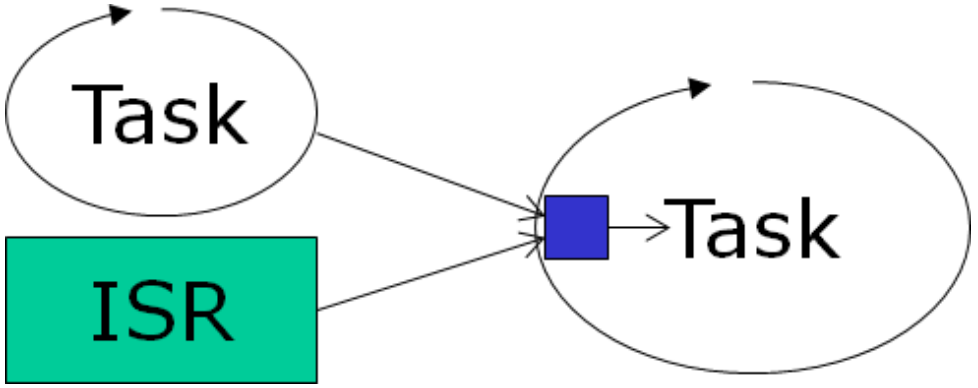
FreeRTOS Features: Software Timers

- Special Timer Daemon task: `TmrSvd`
- Daemon task uses queue for timer commands
- Timers are application callbacks
- Periodic (auto-re-install) and ’one-shot’ timers
- Timer is called from Timer Daemon task context: **do not block**
- Concept works well for low power and tickless idle mode: Scheduler is aware of future events



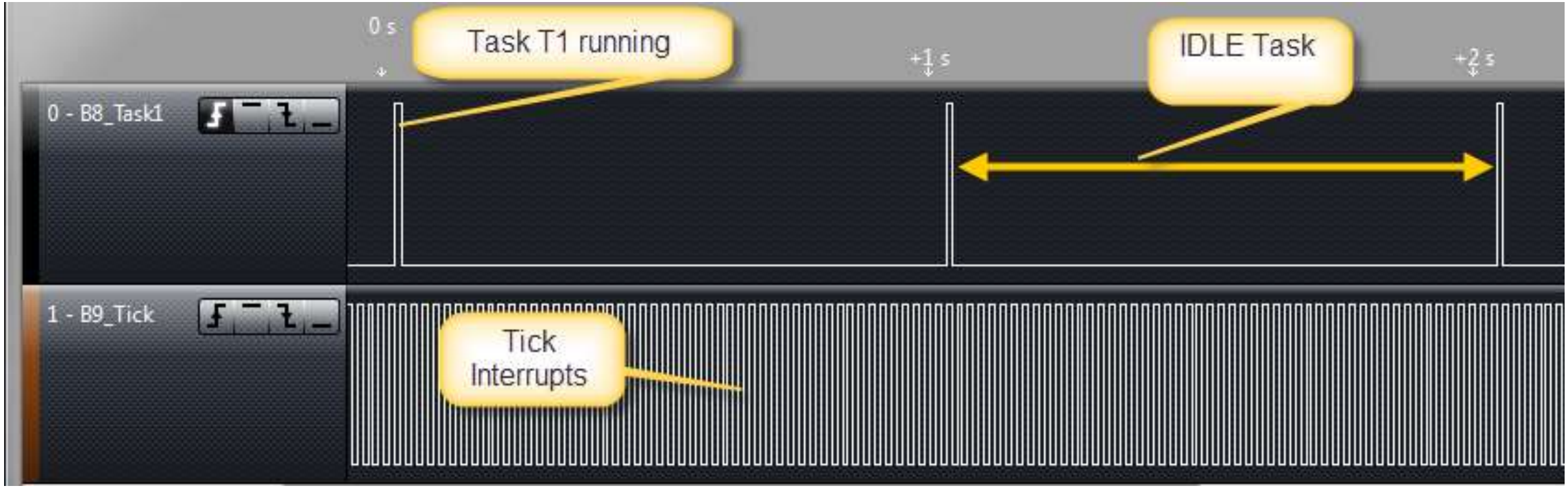
FreeRTOS Feature: Direct Task Notification

- 'Direct' notification to task without intermediate object
  - Enabled with `configUSE_TASK_NOTIFICATION` in `FreeRTOSConfig.h`
  - Faster, less overhead, does not have to be explicitly initialized
  - Each task extended with unsigned 32-bit notification value plus a state enum
    - *eWaitingNotification:* task has been notified, value has been read
    - *eNotWaitingNotification:* task has been notified, value not read yet
    - *eNotified:* the notification value
  - API provides functions to set/clear/increment notification values and to clear state



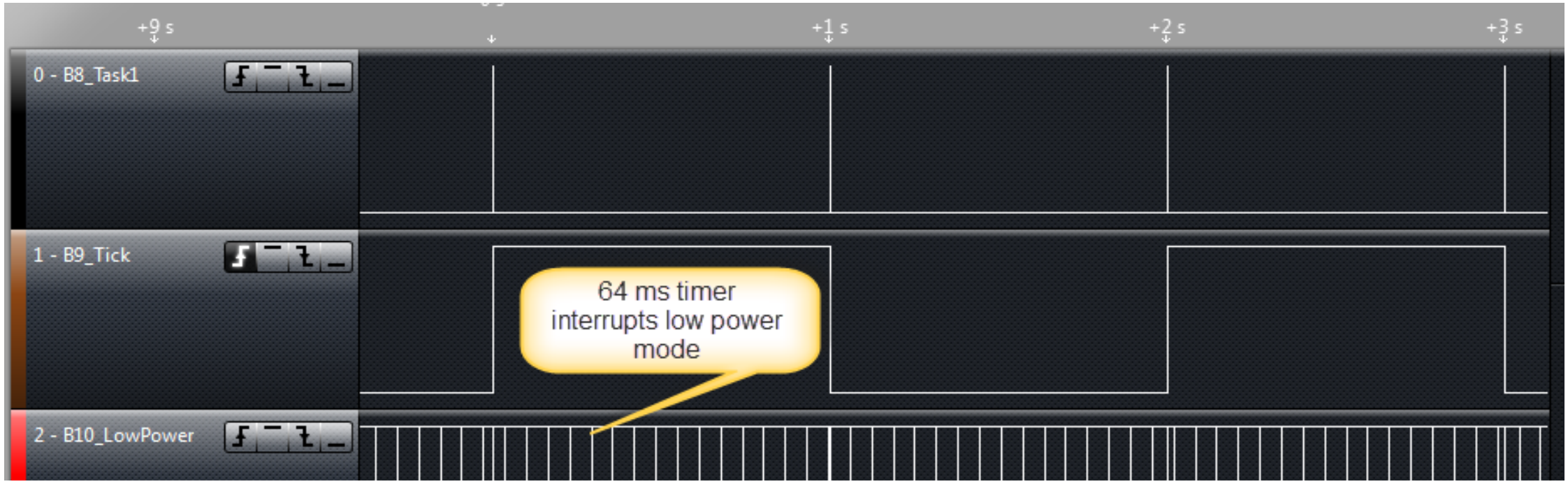
Low-Power: Tick Interrupts

- Tick interrupt wakes CPU up even if no task needs to run
- Wakes CPU up, scheduler re-enters low power mode
- Unnecessary leaving low power mode
- Keep CPU in low-power mode as long as possible



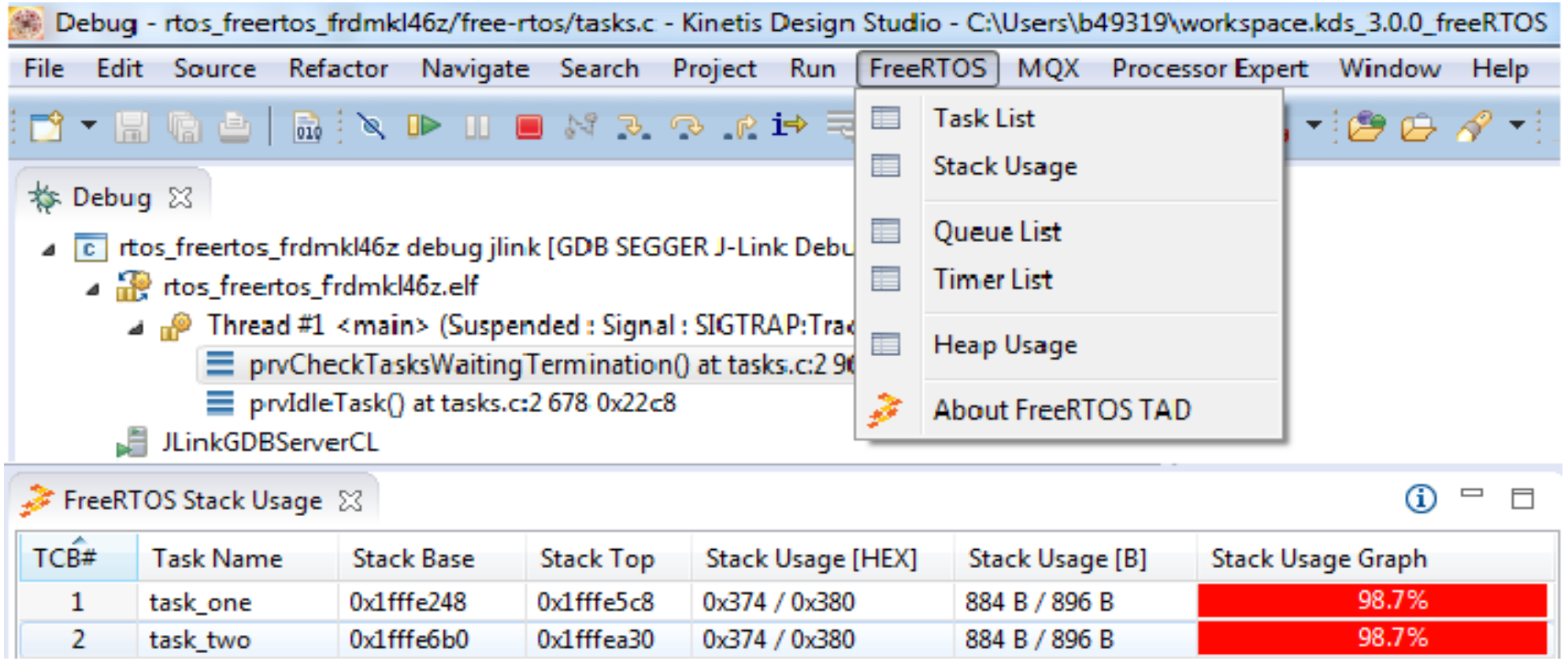
Low-Power: Tickless Idle Mode and Interrupts

- Interrupts (KBI, ...) wake-up CPU, Scheduler needs to check wake-up reason
- Checked by `vPortSuppressTicksAndSleep()` in `port.c`
- Scheduler needs to suspend tick interrupt: tiny drift of tick interrupt
  - `configSTOPPED_TIMER_COMPENSATION`
  - `configEXPECTED_IDLE_TIME_BEFORE_SLEEP`



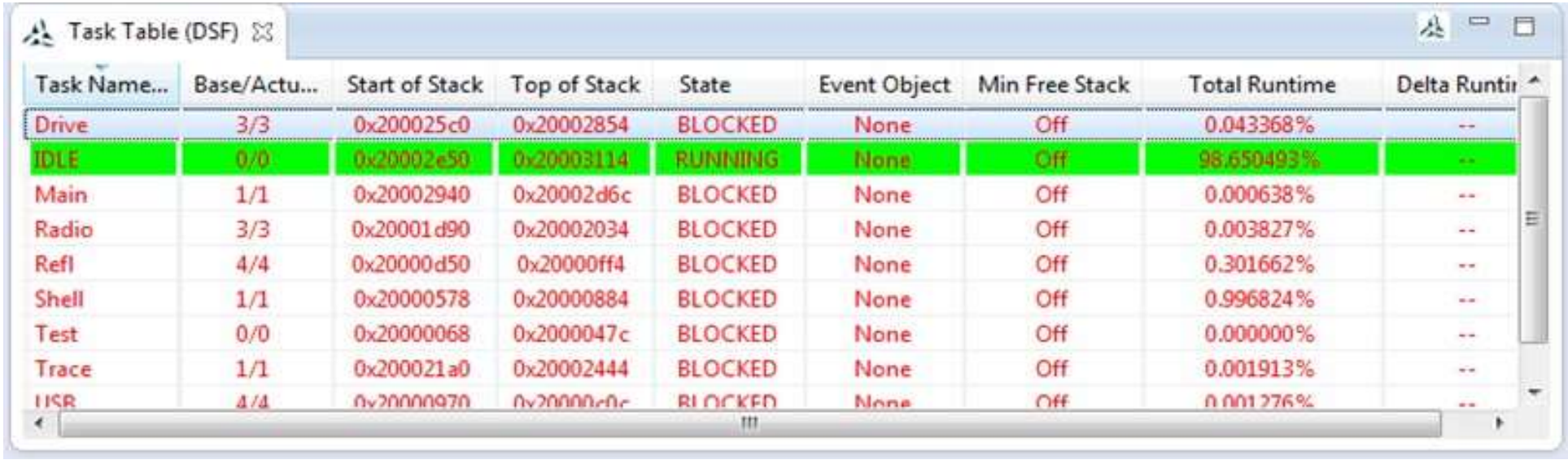
NXP Eclipse Plugins

- Preview: NXP plugins for Eclipse
- Tasks, stacks, queues, timers, heap views
- Multi-thread debug views



Wittenstein Eclipse Plugins

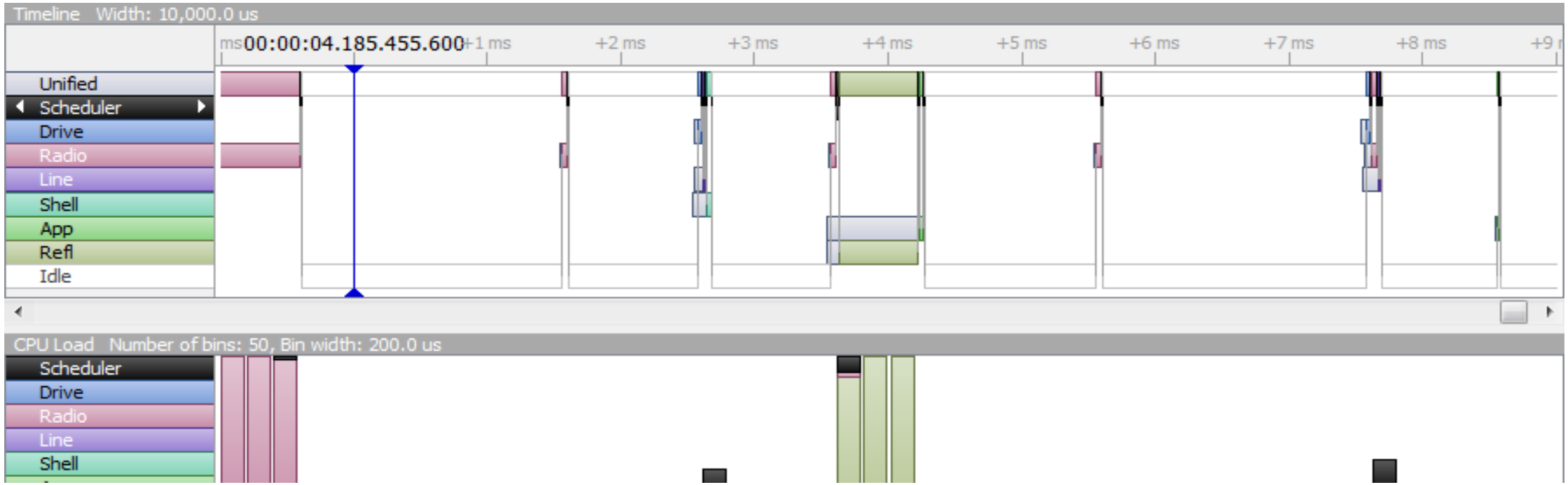
- Set of free Eclipse Debug views from Wittenstein
- http://www.highintegritysystems.com/
- 'Stop-Mode' Views
- Views for Tasks, Queues and Timers



Task Name...	Base/Actu...	Start of Stack	Top of Stack	State	Event Object	Min Free Stack	Total Runtime	Delta Runti
Drive	3/3	0x200025c0	0x20002854	BLOCKED	None	Off	0.043368%	--
IDLE	0/0	0x20002e50	0x20003114	RUNNING	None	Off	98.650493%	--
Main	1/1	0x20002940	0x20002d6c	BLOCKED	None	Off	0.000638%	--
Radio	3/3	0x20001d90	0x20002034	BLOCKED	None	Off	0.003827%	--
Refl	4/4	0x20000d50	0x20000ff4	BLOCKED	None	Off	0.301662%	--
Shell	1/1	0x20000578	0x20000884	BLOCKED	None	Off	0.996824%	--
Test	0/0	0x20000068	0x2000047c	BLOCKED	None	Off	0.000000%	--
Trace	1/1	0x200021a0	0x20002444	BLOCKED	None	Off	0.001913%	--
USB	4/4	0x20000970	0x20000c0c	BLOCKED	None	Off	0.001276%	--

Segger SystemViewer

- Free-of-Charge, requires Segger debug interface
- Based on Segger RTT (Real Time Transfer)
- Realtime data recording and time measurement
- Uses Cortex M4 Cycle count register, SysTick on M0+
- http://mcuoneclipse.com/2015/11/16/segger-systemview



Segger SystemViewer- Timeline

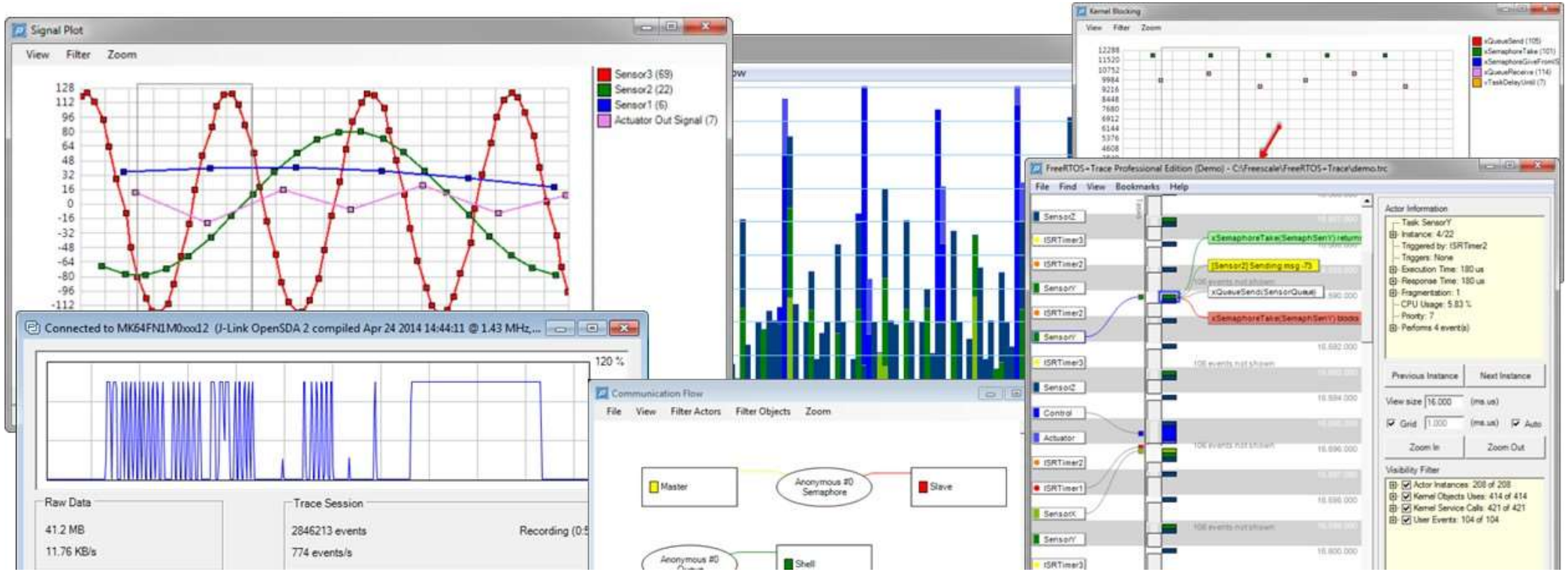
- Up to 1 Mio events
- Scheduler and Idle handled separately
- CPU load with ’bins’: how many items to stack/show in a bar

Events

#	Timestamp	Context	Event	Detail
19650	00:00:04.186.991.333	Idle	Task Ready	Radio, runs after 14.0 us (105 cycles)
19651	00:00:04.187.005.333	Radio	Task Run	Runs for 42.4 us (318 cycles)
19652	00:00:04.187.016.933	Radio	xQueueGeneric…	xQueue=0x00000b06 pvBuffer=0x00000e9f xTicksToWait=…
19653	00:00:04.187.028.933	Radio	xQueueGeneric…	xQueue=0x00000aba pvBuffer=0x000001c0 xTicksToWait=…
19654	00:00:04.187.036.400	Radio	vTaskDelay	xTicksToDelay=2
19655	00:00:04.187.047.733	Radio	Task Block	Radio, Wait for timeout
19656	00:00:04.187.060.133	Sched…	System Idle	Idle for 959.8 us (7 198 cycles)
19657	00:00:04.187.991.333	Idle	Task Ready	Shell, runs after 95.6 us (717 cycles)
19658	00:00:04.187.998.533	Idle	Task Ready	Drive, runs after 21.4 us (160 cycles)
19659	00:00:04.188.005.466	Idle	Task Ready	Line, runs after 56.8 us (426 cycles)
19660	00:00:04.188.019.866	Drive	Task Run	Runs for 30.6 us (230 cycles)
19661	00:00:04.188.034.933	Drive	xQueueGeneric…	xQueue=0x00000806 pvBuffer=0x00000948 xTicksToWait=…

Percepio FreeRTOS+Trace

- Free-of-charge (Task information), (Percepio.com)
- Memory buffer dump/read, unlimited tracing with Segger RTT
- Graph plotting, CPU load, communication flow, ...
- http://mcuoneclipse.com/2015/07/10/trace-streaming



Summary: FreeRTOS

- FreeRTOS is open source, portable, permissible open source license
- Commercial version: **OpenRTOS** and **SafeRTOS**
- Feature Rich: Tasks, Semaphore, Mutex, …, Low Power
- FreeRTOS & NXP MCUXpresso SDK
  - Simplified Architecture
  - On Demand SDK creation with RTOS
- Visualization
  - Eclipse Kernel Awareness
  - Segger SystemViewer
  - Percepio FreeRTOS+Trace

AGENDA

- MCUXpresso Software And Tools Overview
- MCUXpresso SDK Introduction
  - Web Builder
  - File Structure
  - Examples
- MCUXpresso SDK Deep Dive
  - Drivers
  - Interrupts
  - Porting
- FreeRTOS
- Conclusion

LEARN MORE

Web: www.nxp.com/mcuxpresso
Training: www.nxp.com/onlineacademy
Community: https://community.nxp.com/community/mcuxpresso

SECURE CONNECTIONS
FOR A SMARTER WORLD