Hands-On Workshop: Developing on Your Custom Hardware with MCUXpresso Config Tools

Brendon Slade

MCU Ecosystem

May 2019 | AMF-SOL-T3532

SECURE CONNECTIONS FOR A SMARTER WORLD

Agenda

- Overview of MCUXpresso SW and Tools
  - MCUXpresso IDE, SDK, Config Tools
- MCUXpresso SW and Tools Workflow
- MCUXpresso Config Tools
  - FAQs and How-To’s
- Hands-on Workshop
  - Creating a Config Tool configuration
  - Applying configuration to new and existing projects

MCUXpresso Software and Tools

MCUXpresso IDE

MCUXpresso SDK

MCUXpresso Config Tools

MCUXpresso Software and Tools

for LPC & Kinetis MCUs and i.MX RT crossover processors

MCUXpresso IDE

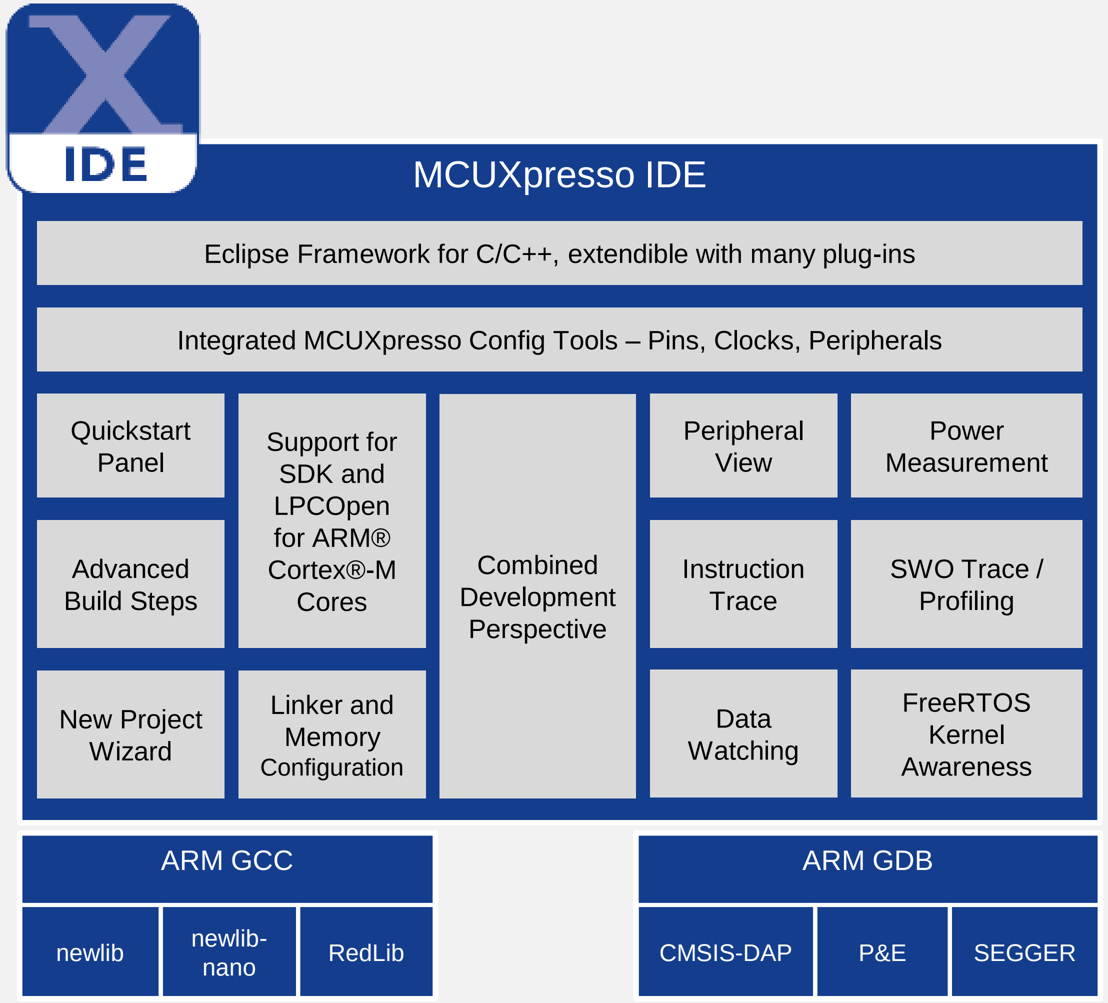
Edit, compile, debug and optimize in an intuitive and powerful IDE

MCUXpresso SDK

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

MCUXpresso Config Tools

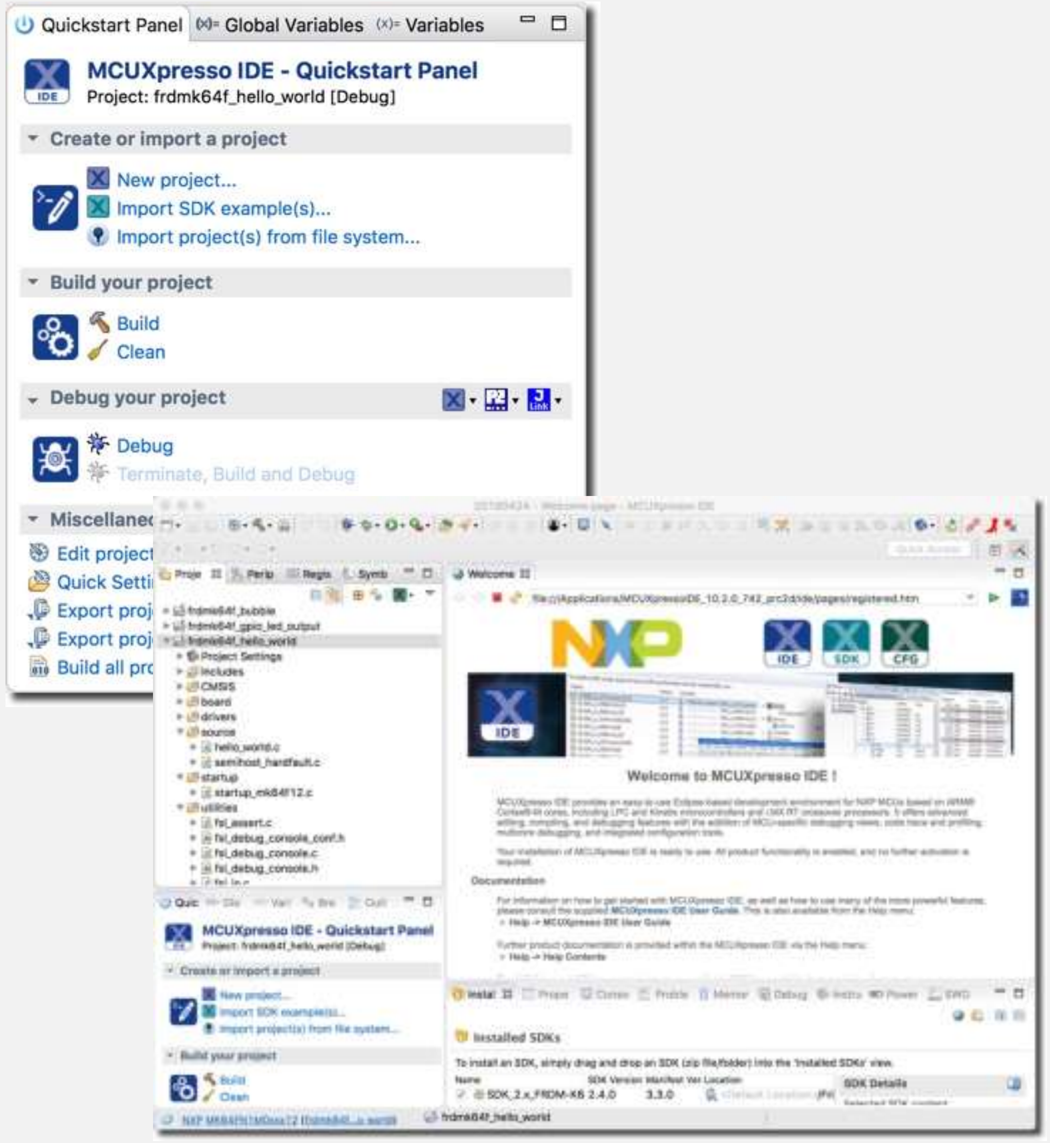
Online and desktop tool suite for system configuration and optimization


IDE
MCUXpresso IDE
Eclipse Framework for C/C++, extendible with many plug-ins
Integrated MCUXpresso Config Tools – Pins, Clocks, Peripherals
Quickstart Panel
Support for SDK and LPCOpen for ARM® Cortex®-M Cores
Combined Development Perspective
Peripheral View
Power Measurement
Advanced Build Steps
Instruction Trace
SWO Trace / Profiling
New Project Wizard
Linker and Memory Configuration
Data Watching
FreeRTOS Kernel Awareness
ARM GCC
newlib
newlib-nano
RedLib
ARM GDB
CMSIS-DAP
P&E
SEGGER

MCUXpresso IDE

Free Eclipse / GCC-based development

- **Feature-rich**, **unlimited code size**, optimized for ease-of-use, based on industry standard Eclipse framework for NXP's **Kinetis** and **LPC** MCUs and **i.MX RT** crossover processors
- 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:** Full Featured, unlimited Code Size, no special activation needed, community based support, advanced trace capabilities, MTB and ETB instruction trace
- Works in conjunction with **MCUXpresso Config Tools** and **MCUXpresso SDK** to provide complete development environment


Quickstart Panel | Global Variables | Variables

MCUXpresso IDE - Quickstart Panel
Project: frdmk64f_hello_world [Debug]

- Create or import a project
  - New project...
  - Import SDK example(s)...
  - Import project(s) from file system...
- Build your project
  - Build
  - Clean
- Debug your project
  - Debug
  - Terminate, Build and Debug
- Miscellaneo
  - Edit project
  - Quick Setti
  - Export proj
  - Export proj
  - Build all pro

Welcome to MCUXpresso IDE !

Installed SDKs

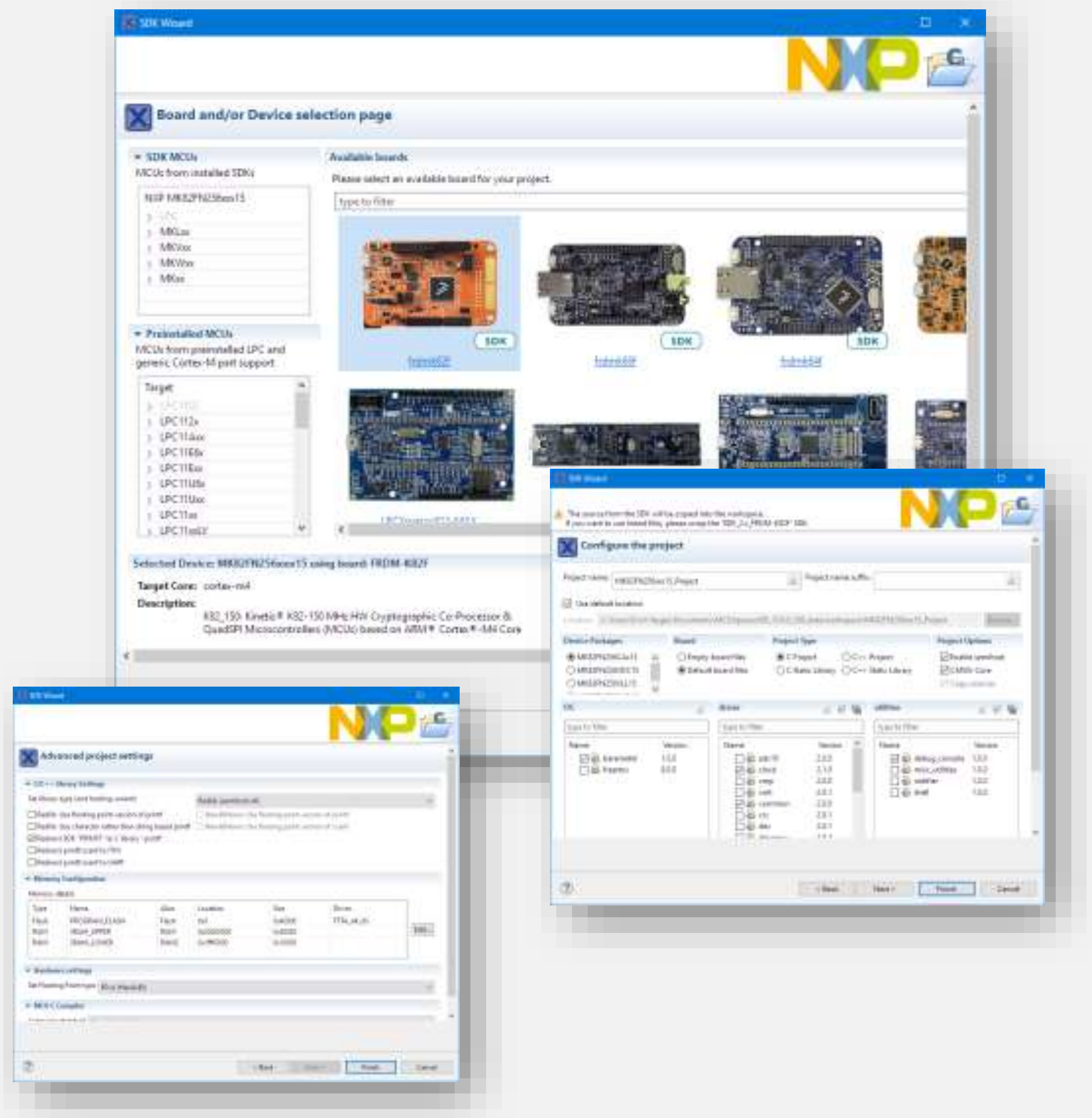
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view.

Name	SDK Version	Manifest Version	Location
SDK_2.x_FRDM-K6	2.4.0	3.3.0	

MCUXpresso IDE

Built for Ease of Use

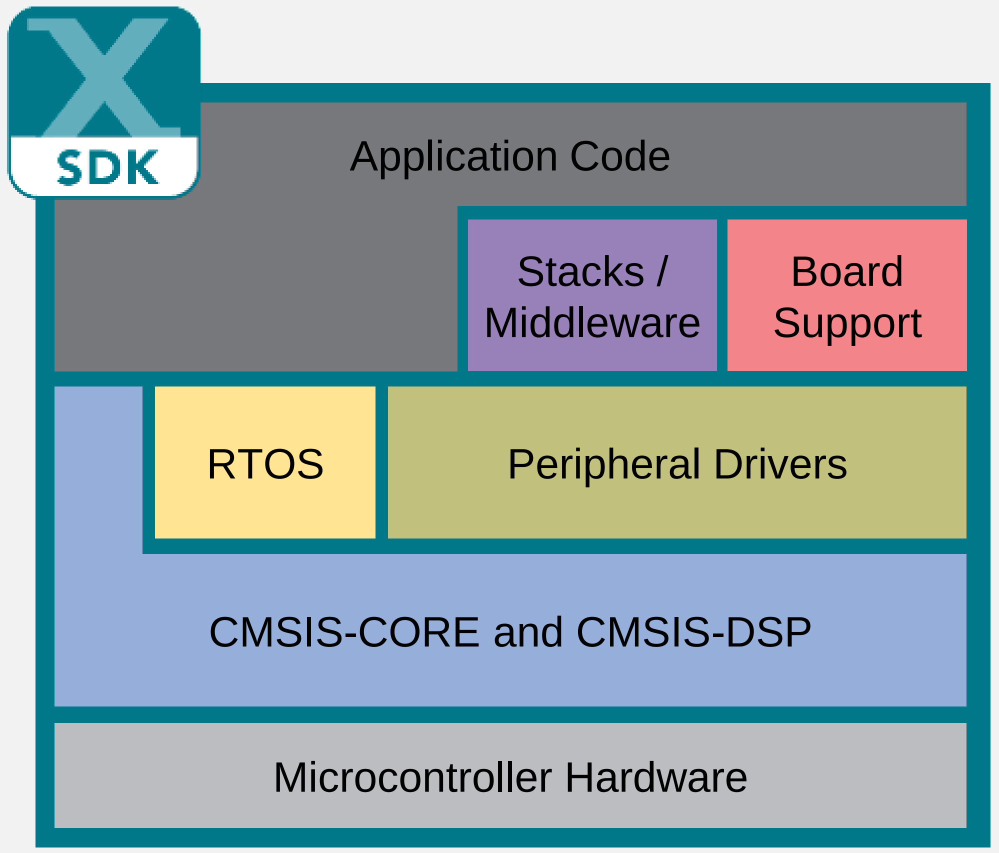
- **Quickstart Panel** guides users to most commonly used options
  - One-Click access to most used functions (Create, Build, Debug)
  - Direct access to standard debug functions (Debug, Attach, Program, Erase)
- **Develop Perspective** for both project editing and debugging
  - Simplifies Eclipse usage
- **GUI Flash Tool** for simplified programming with support for LinkServer, P&E, and Segger



MCUXpresso IDE

New Project Wizard

- SDK MCUs (LPC and Kinetis)
- Preinstalled LPC and generic Cortex®-M
- Installable device support through SDK packages (data driven)
- Selection of package, RTOS, drivers, utilities
- Standalone and linked projects
- Advanced project settings



MCUXpresso SDK

Software Framework and Drivers

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

Integrated RTOS:
- Amazon FreeRTOS
- RTOS-native driver wrappers

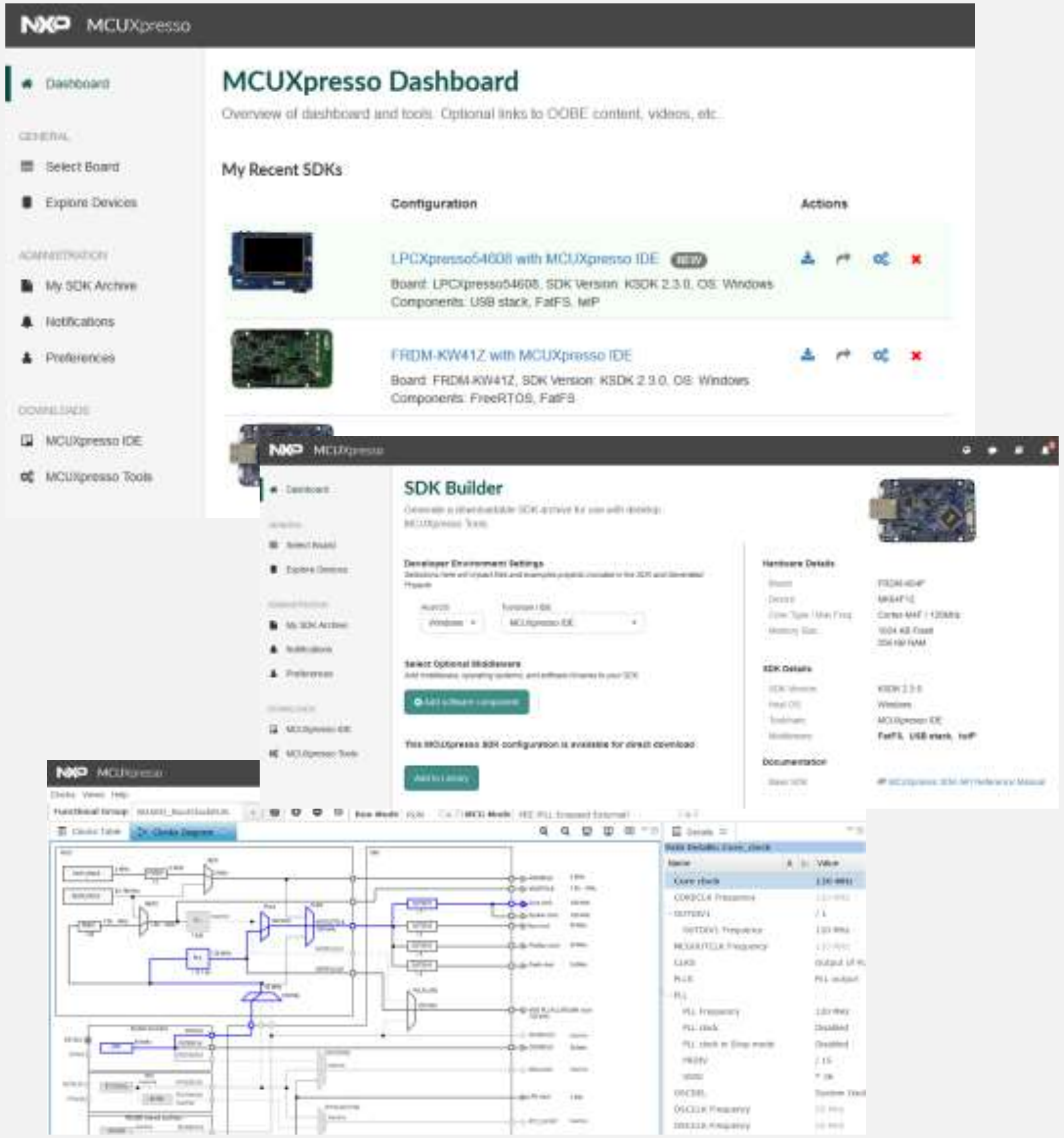
Integrated Stacks and Middleware:
- USB Host, Device and OTG
- lwIP, FatFS, LittleFS
- Crypto acceleration plus wolfSSL & mbedTLS
- AWS IoT and Azure IoT
- SD and eMMC card support

Reference Software:
- Peripheral driver usage examples
- Application demos
- FreeRTOS usage demos
- IoT connectivity examples

License:
- Clear 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 Dashboard

Overview of dashboard and tools. Optional links to OOBE content, videos, etc.

My Recent SDKs

	Configuration	Actions
	LPCXpresso54608 with MCUXpresso IDE NEW Board: LPCXpresso54608, SDK Version: KSDK 2.3.0, OS: Windows Components: USB stack, FatFS, lwIP	
	FRDM-KW41Z with MCUXpresso IDE Board: FRDM-KW41Z, SDK Version: KSDK 2.3.0, OS: Windows Components: FreeRTOS, FatFS	

MCUXpresso SDK

Online Builder and Dashboard

- MCUXpresso Dashboard
  - Access all previously downloaded SDKs
  - Notifications of update SDK content specific to your own SDK packages
  - Share SDK archive with other users
- MCUXpresso SDK Builder
  - Build customized SDKs for selected development board or device
  - Specify optional middleware, with dependency resolution
- Online MCUXpresso Config Tools
  - Run basic checks on Pin Muxing and Clock Configuration to aid in ideal device selection
  - Download Config Tool output for further development with desktop Config Tools

MCUXpresso SDK

Integrated Middleware

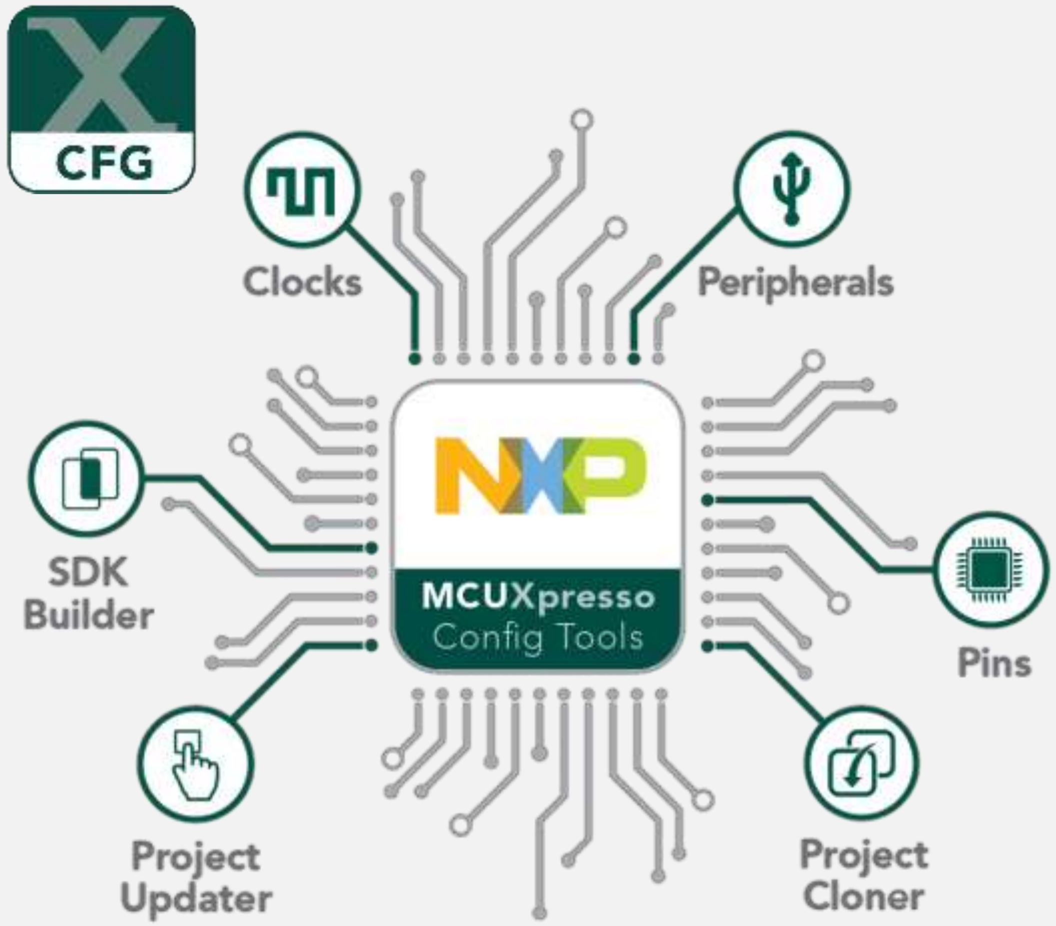
3rd Party Included Middleware

- Amazon FreeRTOS
  - FreeRTOS Kernel
  - AWS IoT Device SDK
- Filesystem:
  - FatFS, littleFS
- TCP/IP stack (lwIP)
- SSL/TLS:
  - mbedTLS, wolfSSL
- QCA Wifi stack
- SEGGER emWIN Graphics
- NAND Flash Management
- JPEG Encode / Decode

NXP Developed Middleware

- USB Stack (Host, Device, OTG)
- SDMMC
- Crypto Acceleration Software Libraries
- Real-Time Control Embedded Software Libraries
  - Motor Control, Math Functions, Digital Filtering
- BLDC / PMSM Motor Control Algorithms
- Companion Product Support
  - NTAG, IoT Sensing SDK, Touch Software
- Wireless Stacks:
  - Thread, BLE, 802.15.4 MAC,
    Zigbee, GenFSK, SMAC
- DMA Manager (DMAMUX)
- EMV Level 1 Contact Stack
- Multicore support
  - eRCP (embedded Remote Procedure Call)
  - RPMSG-lite (Remote Processor Messaging)
  - Multicore Manager Software Libraries

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



MCUXpresso Config Tools

Configuration and Code Generation

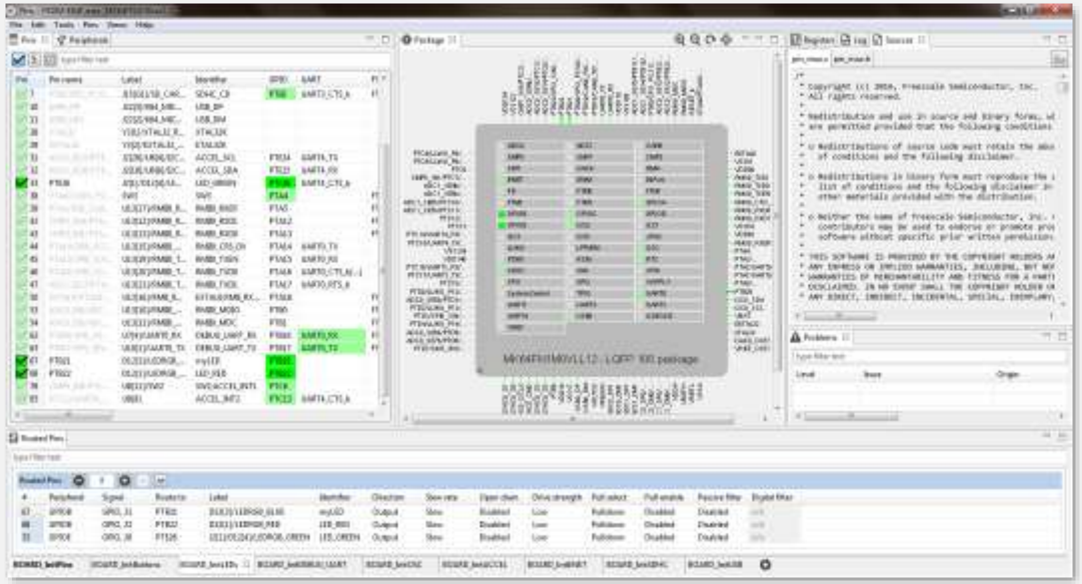
SDK Builder packages custom SDKs based on user selections of MCU, evaluation board, and optional software components.

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

Project Update works directly with existing SDK-based IDE projects with generated Pins, Clocks, and Peripheral source files.

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

Easy-to-use muxing and pin assignments

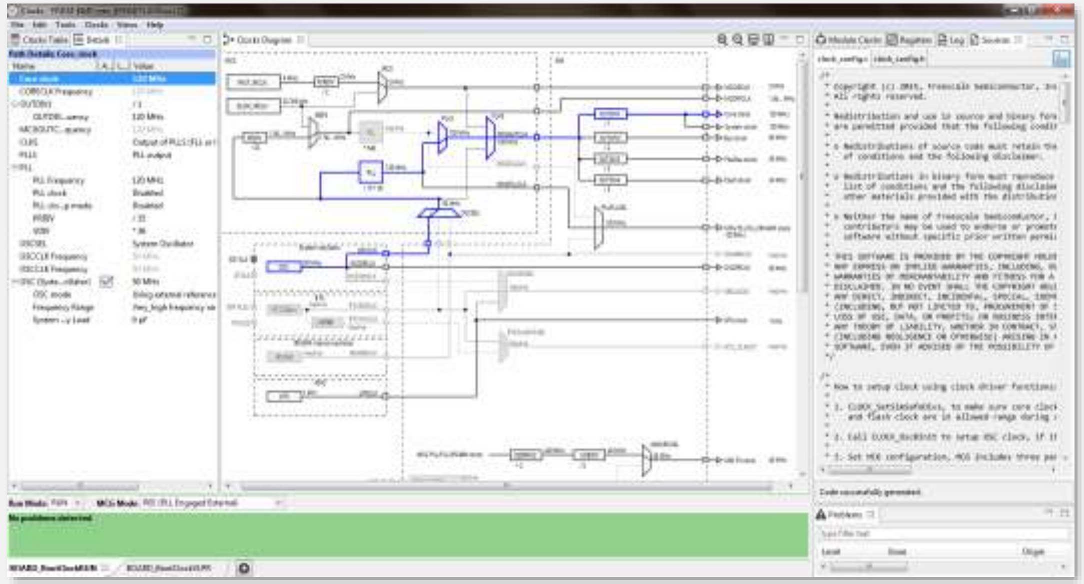


MCUXpresso Config Tools

Pins Configuration

- Muxing and pin configuration with consistency checking
- ANSI-C configuration code
- Graphical processor package view
- Wizard for optimized assignments of functionality to available pins
  - Selection of Pins and Peripherals
  - Routed pins with electrical characteristics
  - Registers with configured and reset values
  - Source code for C/C++ applications
  - GPIO Input / Output initialization
- Documented and easy to understand source code
- Report generation
- Integrates with any compiler and IDE

Clock configuration and diagram view

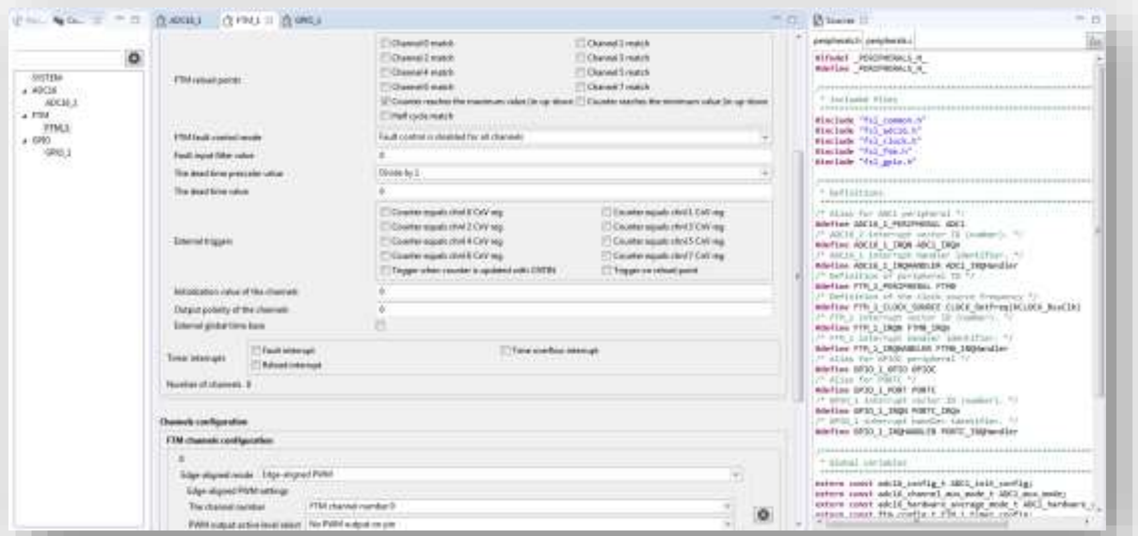


MCUXpresso Config Tools

Clock Configuration

- System clock configuration with consistency checking
- ANSI-C initialization code
- Graphical clock diagrams
- Easy-to-use guided graphical user interface
  - Selection of Clock Sources
  - Configuration of prescalers and clock outputs
  - Details and Full Diagram views with clock path
  - Registers with configured and reset values
  - Source code for C/C++ applications
- Documented and easy to understand source code
- Report generation

Clock configuration and diagram view



FTM reload points: Channel0 match, Channel1 match, Channel2 match, Channel3 match, Channel4 match, Channel5 match, Channel6 match, Channel7 match, [✓ Counter reaches the maximum value (in up-down] Counter reaches the minimum value (in up-down, Half cycle match

FTM fault control mode: Fault control is disabled for all channels

Fault input filter value: 0

The dead time prescaler value: Divide by 1

The dead time value: 0

External triggers: Counter equals chnl 0 CnV reg, Counter equals chnl 1 CnV reg, Counter equals chnl 2 CnV reg, Counter equals chnl 3 CnV reg, Counter equals chnl 4 CnV reg, Counter equals chnl 5 CnV reg, Counter equals chnl 6 CnV reg, Counter equals chnl 7 CnV reg, Trigger when counter is updated with CNTIN, Trigger on reload point

Initialization value of the channels: 0

Output polarity of the channels: 0

External global time base

Timer interrupts: Fault interrupt, Timer overflow interrupt, Reload interrupt

Number of channels: 8

Channels configuration

FTM channels configuration

0

Edge-aligned mode: Edge-aligned PWM

Edge-aligned PWM settings

The channel number: FTM channel number 0

PWM output active level select: No PWM output on pin

```
#ifndef _PERIPHERALS_H_
#define _PERIPHERALS_H_

/***********************************************************************
 * Included files
 **********************************************************************/
#include "fsl_common.h"
#include "fsl_adc16.h"
#include "fsl_clock.h"
#include "fsl_ftm.h"
#include "fsl_gpio.h"

/***********************************************************************
 * Definitions
 **********************************************************************/
/* Alias for ADC1 peripheral */
#define ADC16_1_PERIPHERAL ADC1
/* ADC16_1 interrupt vector ID (number). */
#define ADC16_1_IRQN ADC1_IRQn
/* ADC16_1 interrupt handler identifier. */
#define ADC16_1_IRQHANDLER ADC1_IRQHandler
/* Definition of peripheral ID */
#define FTM_1_PERIPHERAL FTM0
/* Definition of the clock source frequency */
#define FTM_1_CLOCK_SOURCE CLOCK_GetFreq(kCLOCK_BusClk)
/* FTM_1 interrupt vector ID (number). */
#define FTM_1_IRQN FTM0_IRQn
/* FTM_1 interrupt handler identifier. */
#define FTM_1_IRQHANDLER FTM0_IRQHandler
/* Alias for GPIOC peripheral */
#define GPIO_1_GPIO GPIOC
/* Alias for PORTC */
#define GPIO_1_PORT PORTC
/* GPIO_1 interrupt vector ID (number). */
#define GPIO_1_IRQN PORTC_IRQn
/* GPIO_1 interrupt handler identifier. */
#define GPIO_1_IRQHANDLER PORTC_IRQHandler

/***********************************************************************
 * Global variables
 **********************************************************************/
extern const adc16_config_t ADC1_init_config;
extern const adc16_channel_mux_mode_t ADC1_mux_mode;
extern const adc16_hardware_average_mode_t ADC1_hardware_
extern const ftm_config_t FTM_1_timer_config;
```

MCUXpresso Config Tools

Peripheral Configuration

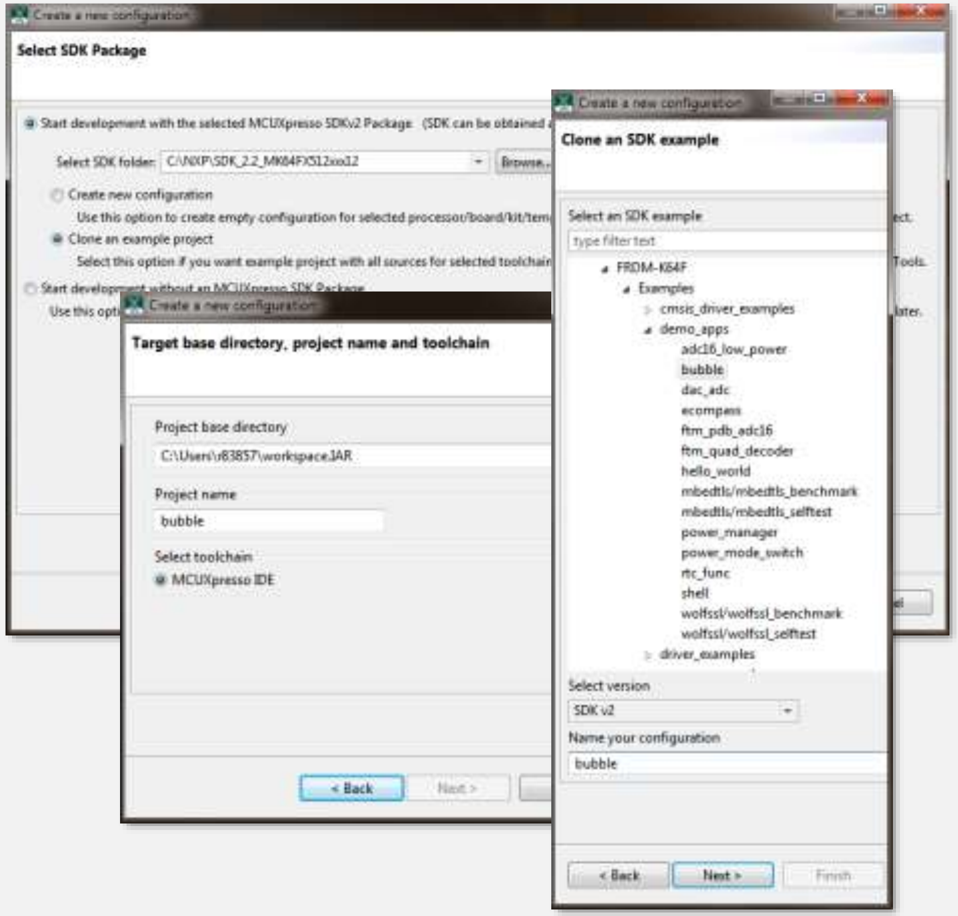
- SDK peripheral configuration
- Validation of user inputs / selection
- ANSI-C initialization code
- Generation of MCUXpresso SDK Initialization Structure
- Selection of common use case configurations
- Support for most common peripherals, with increasing part family and feature support
- Documented and easy to understand source code
- Report generation

MCUXpresso SDK Project Cloning

MCUXpresso Config Tools

Cloning Utility

- Ability to generate a fully standalone MCUXpresso project cloned from one of the many included examples
- Provide a native IDE project for any toolchain supported in your SDK configurations
- Available in the desktop version on the MCUXpresso Config Tool as part of the “New configuration dialog”
- Available in the online version of the MCUXpresso SDK Builder and webpage
- Clones example projects can be downloaded directly from the MCUXpresso webpage. Online cloned projects provide all project and SDK files required to quickly have an application running on a support NXP development board in a single download


Create a new configuration

Select SDK Package

- [✓ Start development with the selected MCUXpresso SDKv2 Package] (SDK can be obtained
  - Select SDK folder: C:\NXP\SDK_2.2_MK64FX512xxx12
  - Create new configuration — Use this option to create empty configuration for selected processor/board/kit/tem
  - [✓ Clone an example project] — Select this option if you want example project with all sources for selected toolchain
- Start development without an MCUXpresso SDK Package — Use this opti

Create a new configuration

Target base directory, project name and toolchain

Project base directory: C:\Users\r83857\workspace.IAR

Project name: bubble

Select toolchain: [✓ MCUXpresso IDE]

Create a new configuration

Clone an SDK example

Select an SDK example

```
FRDM-K64F
  Examples
    cmsis_driver_examples
    demo_apps
      adc16_low_power
      bubble
      dac_adc
      ecompass
      ftm_pdb_adc16
      ftm_quad_decoder
      hello_world
      mbedtls/mbedtls_benchmark
      mbedtls/mbedtls_selftest
      power_manager
      power_mode_switch
      rtc_func
      shell
      wolfssl/wolfssl_benchmark
      wolfssl/wolfssl_selftest
    driver_examples
```

Select version: SDK v2

Name your configuration: bubble

MCUXpresso SW and Tools Workflow

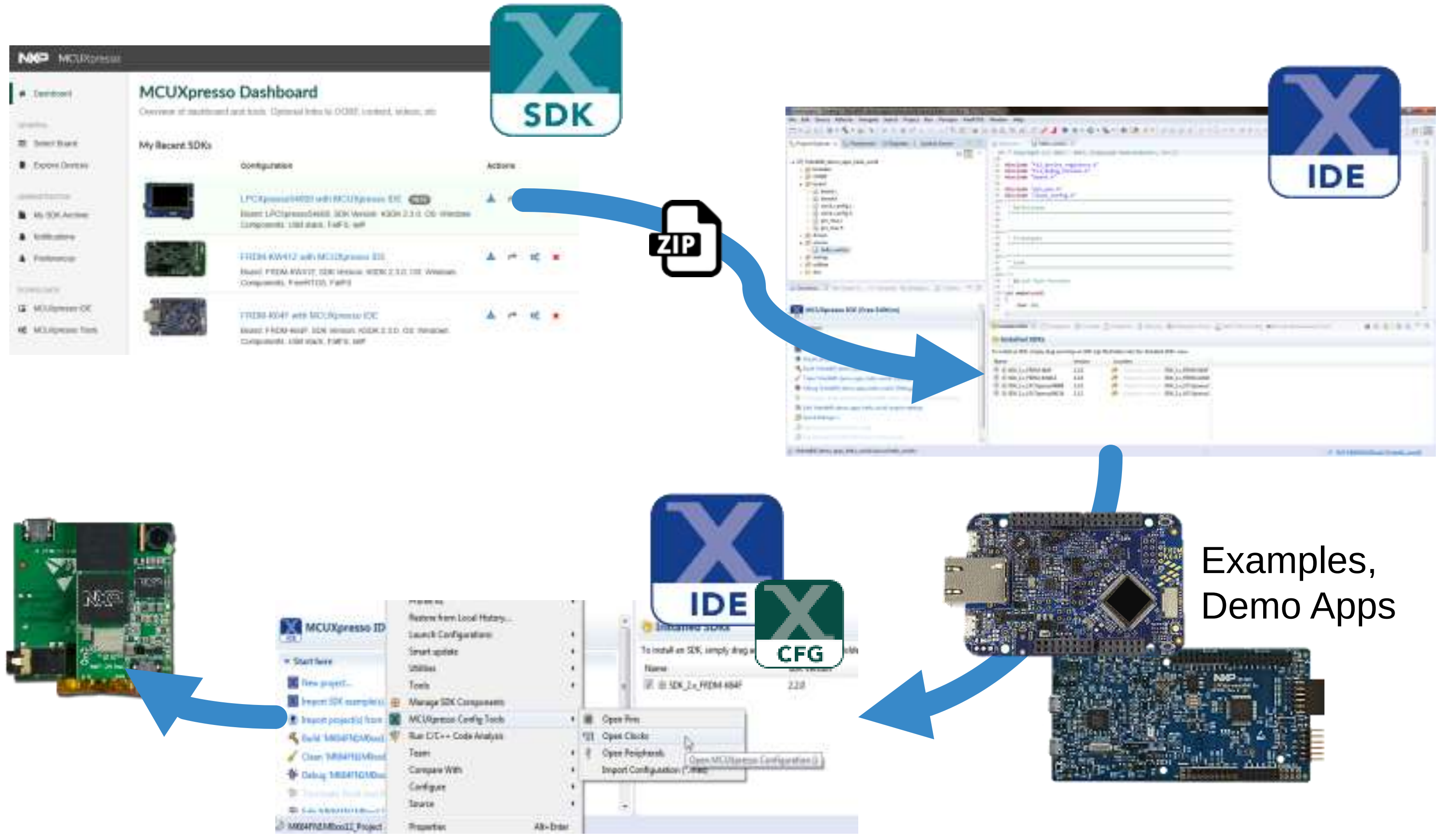
Efficient Development Flow

- SDK Configuration and Installation

- Integrated Config Tools

MCUXpresso SW and Tools

Efficient Development Flow



- Online Custom SDK Builder
- Drag-and-Drop installation of SDK into IDE
- SDK Project Importing / Cloning
- Demo applications, SDK driver examples, middleware use case projects
- Management of SDK drivers and middleware components
- Integrated Config Tools
- Pins and Clocks initialization for user defined boards

MCUXpresso Config Tools

FAQs and How-To’s

MCUXpresso Config Tool Basic Concepts

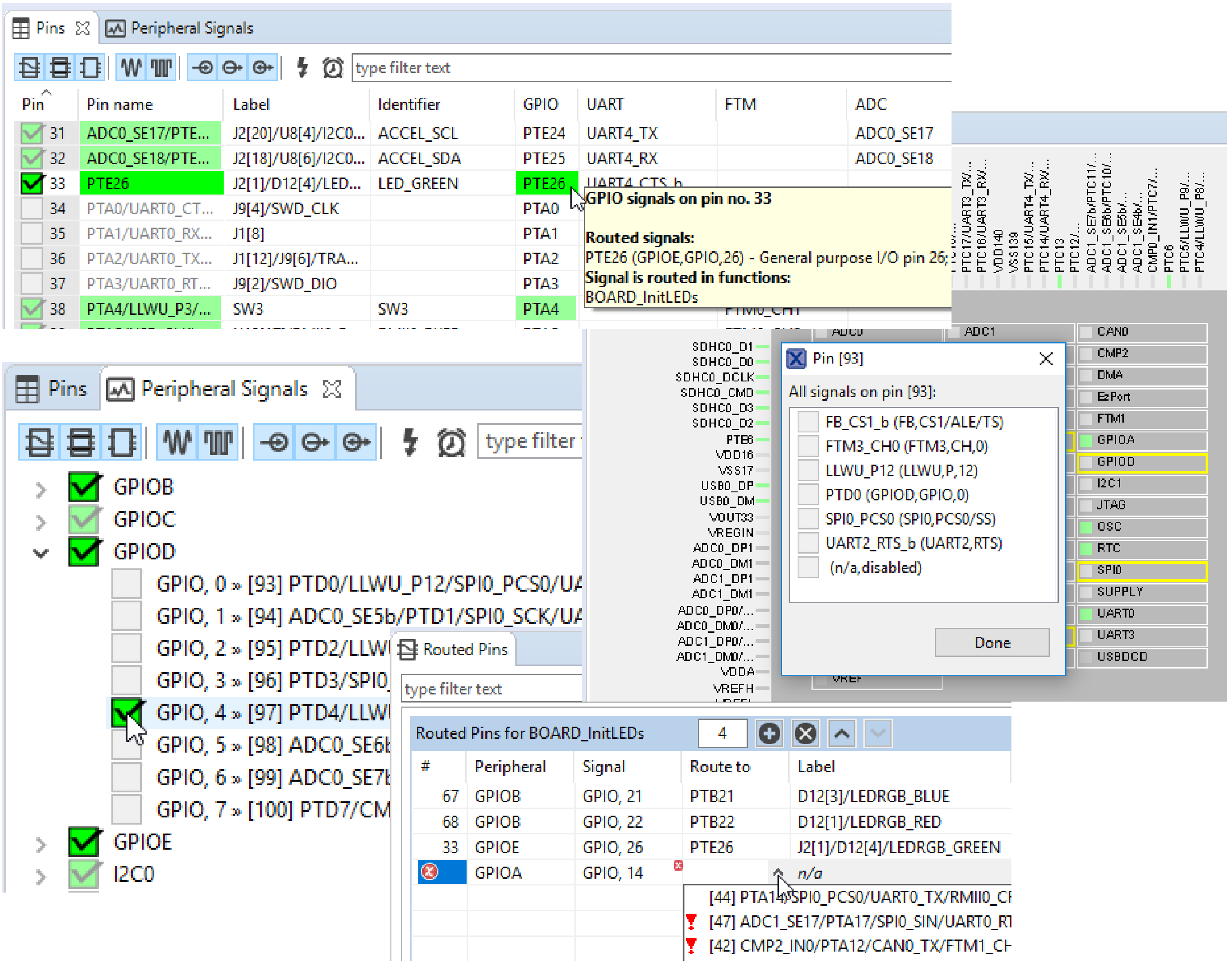
- Generated files include special comments that allow the configuration to be restored directly from source code
- The full configuration is save into a “.mex” configuration file that can be imported into new configurations to apply those settings
- MCUXpresso SDK Example projects include “special comments” that allow the examples to but used within the Config Tools
- MCUXpresso Config Tool does not generate projects, but is designed to work with existing projects, however there are instances where is can clone an existing SDK example as a starting reference
- Functional groups enable flexibility within application development to (re)configure pins and peripherals at runtime
- Functional groups are available for clock configuration, but are not intended for switch clock settings during runtime
- The peripheral tool is designed to supplement the MCUXpresso SDK by providing a graphical configuration utility for the <peripheral>_Init() function. The peripheral tool generated the input structure to this function call.
- Config Tool data is downloaded independent of the IDE and SDK, when accessing the Config Tools, the tool will check online for the latest available data (which can include support for new features)

Online vs. Integrated vs. Standalone

- Online (http://mcuxpresso.nxp.com)
  - Useful during device selection process
  - Quick sanity check that device pin muxing and clock scheme will meet product requirements
  - Output can be exported for “real” work later with desktop versions of the config tools
- Integrated (with MCUXpresso IDE)
  - Seamless workflow, implemented as additional eclipse perspectives
  - Works with new project generation and SDK component manager, allowing missing drivers to added to the IDE project folder
- Standalone (Windows, Mac, Linux)
  - Designed to be used with other IDEs (IAR Embedded Workbench, Keil uVision)
  - Can clone MCUXpresso SDK examples directly for use with supported IDEs
  - Supports direct updating of generated files detected within an existing toolchain project
  - Useful for hardware engineer that does not work directly with IDE

How to Configure a Pin?

- Pins Table – Select cell at the cross section of the Pin and Mux setting
- Graphical Chip view – popup dialog will guide thru possible selections
- Peripheral Signals view – similar to graphical chip view, with signals categorized as peripherals
- Add new entry to “Routed Pins Table”



Pin	Pin name	Label	Identifier	GPIO	UART	FTM	ADC
✓ 31	ADC0_SE17/PTE...	J2[20]/U8[4]/I2C0...	ACCEL_SCL	PTE24	UART4_TX		ADC0_SE17
✓ 32	ADC0_SE18/PTE...	J2[18]/U8[6]/I2C0...	ACCEL_SDA	PTE25	UART4_RX		ADC0_SE18
✓ 33	PTE26	J2[1]/D12[4]/LED...	LED_GREEN	PTE26	UART4_CTS_b		
34	PTA0/UART0_CT...	J9[4]/SWD_CLK		PTA0			
35	PTA1/UART0_RX...	J1[8]		PTA1			
36	PTA2/UART0_TX...	J1[12]/J9[6]/TRA...		PTA2			
37	PTA3/UART0_RT...	J9[2]/SWD_DIO		PTA3			
✓ 38	PTA4/LLWU_P3/...	SW3	SW3	PTA4		FTM0_CH1	

GPIO signals on pin no. 33

Routed signals:
PTE26 (GPIOE,GPIO,26) - General purpose I/O pin 26;
Signal is routed in functions:
BOARD_InitLEDs

Pin [93]

All signals on pin [93]:

- FB_CS1_b (FB,CS1/ALE/TS)
- FTM3_CH0 (FTM3,CH,0)
- LLWU_P12 (LLWU,P,12)
- PTD0 (GPIOD,GPIO,0)
- SPI0_PCS0 (SPI0,PCS0/SS)
- UART2_RTS_b (UART2,RTS)
- (n/a,disabled)

Done

- ✓ GPIOB
- ✓ GPIOC
- ✓ GPIOD
  - GPIO, 0 » [93] PTD0/LLWU_P12/SPI0_PCS0/UA
  - GPIO, 1 » [94] ADC0_SE5b/PTD1/SPI0_SCK/UA
  - GPIO, 2 » [95] PTD2/LLWU
  - GPIO, 3 » [96] PTD3/SPI0
  - ✓ GPIO, 4 » [97] PTD4/LLWU
  - GPIO, 5 » [98] ADC0_SE6b
  - GPIO, 6 » [99] ADC0_SE7b
  - GPIO, 7 » [100] PTD7/CM
- ✓ GPIOE
- ✓ I2C0

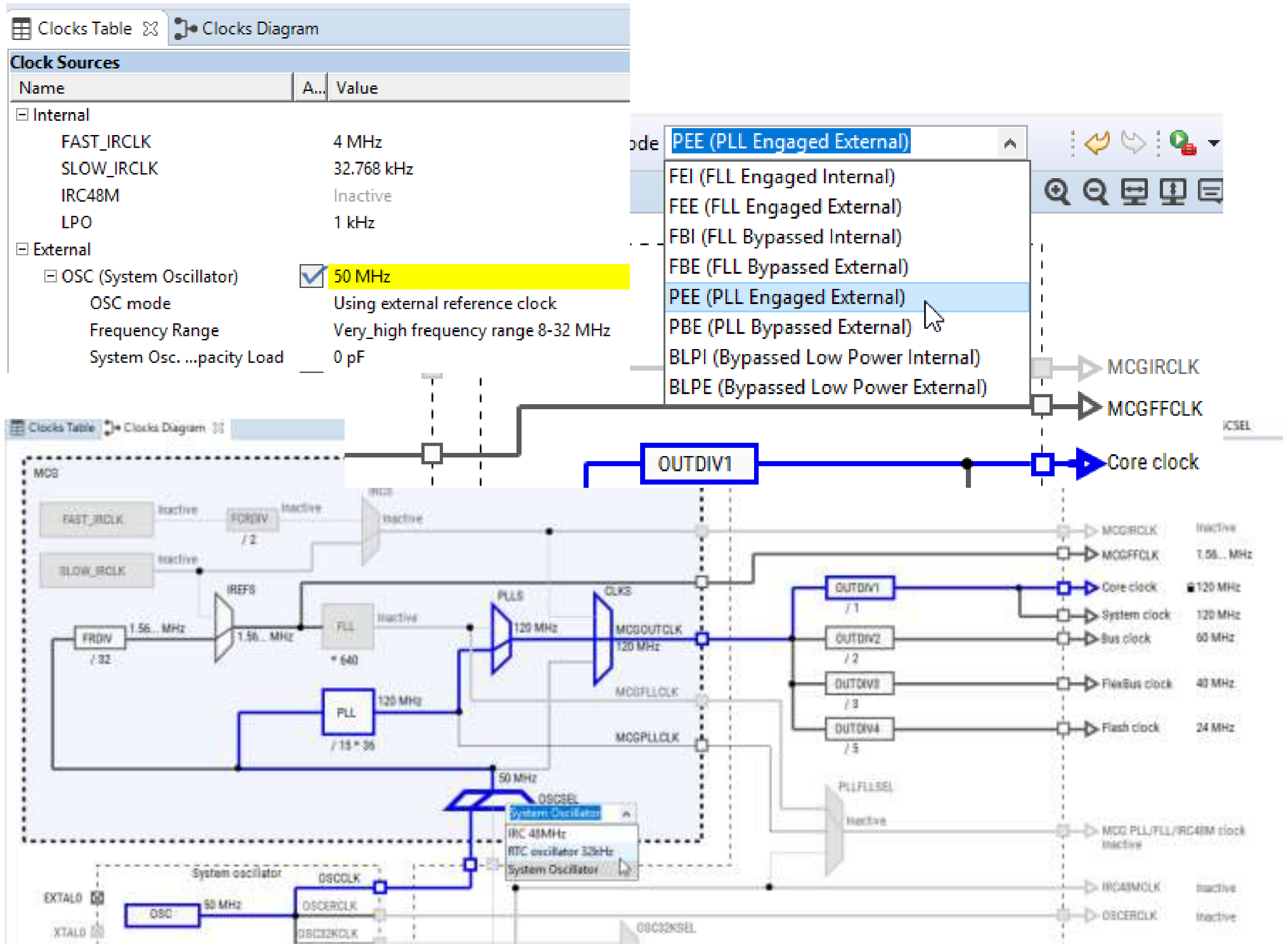
Routed Pins for BOARD_InitLEDs 4

#	Peripheral	Signal	Route to	Label
67	GPIOB	GPIO, 21	PTB21	D12[3]/LEDRGB_BLUE
68	GPIOB	GPIO, 22	PTB22	D12[1]/LEDRGB_RED
33	GPIOE	GPIO, 26	PTE26	J2[1]/D12[4]/LEDRGB_GREEN
	GPIOA	GPIO, 14		n/a

[44] PTA14/SPI0_PCS0/UART0_TX/RMII0_CF
[47] ADC1_SE17/PTA17/SPI0_SIN/UART0_RT
[42] CMP2_IN0/PTA12/CAN0_TX/FTM1_CH

How to Configure Clock Settings?

- Recommended to setup clock sources first
- Select functional group's clock mode (MGC / PLL) to adjust several settings in sync
- Select desired output clock within Clocks Diagram view to see full clock path and limit details panel to relevant parameters
- Selecting on an object within the graphical view will allow many setting to be selected directly



Clocks Table | Clocks Diagram

Clock Sources

Name	A...	Value
⊟ Internal		
FAST_IRCLK		4 MHz
SLOW_IRCLK		32.768 kHz
IRC48M		Inactive
LPO		1 kHz
⊟ External		
⊟ OSC (System Oscillator)	✓	50 MHz
OSC mode		Using external reference clock
Frequency Range		Very_high frequency range 8-32 MHz
System Osc. ...pacity Load		0 pF

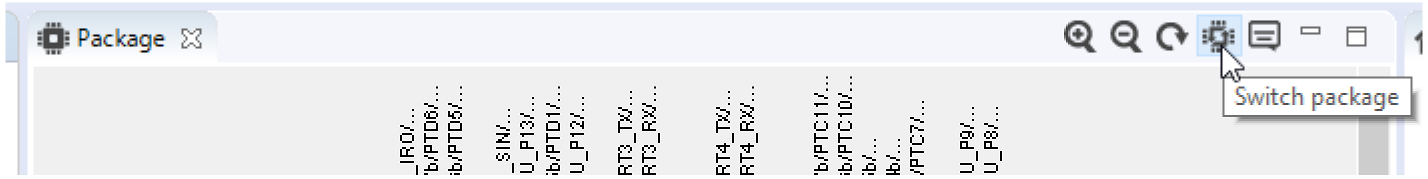
Mode: PEE (PLL Engaged External)

- FEI (FLL Engaged Internal)
- FEE (FLL Engaged External)
- FBI (FLL Bypassed Internal)
- FBE (FLL Bypassed External)
- PEE (PLL Engaged External)
- PBE (PLL Bypassed External)
- BLPI (Bypassed Low Power Internal)
- BLPE (Bypassed Low Power External)

How to Change Device Package?

Requires edits to IDE, SDK, and Config Tools:

- MCUXpresso IDE:
  - Not explicitly required, but can be changed to prevent conflict messaging in Config Tools
  - Requires manually editing .cproject file, editable in IDE 10.3 onwards
- MCUXpresso SDK:
  - Package variants use exact same SDK driver files.
  - SDK uses preprocessor defines to properly compile SDK drivers for different packages
  - Preprocessor defines can be edited within the IDE project settings
- MCUXpresso Config Tools:
  - Use the “switch package” icon from within the Pins Tool to select a different package
  - Config Tools will attempt to transfer all compatible pin muxing settings to the new package



How to Change Device Part Number?

Option 1 – Create new project: Recommended / Cleaner approach

- MCUXpresso IDE:
  - Create a new blank SDK project for the desired device, this will ensure the IDE and SDK are setup correctly and include the correct SDK driver files.
  - Include all required drivers during the new project creation wizard (these can also be added later)
  - Copy application specific source files from previous project to new project (sources subfolder)
  - Update device header includes in application specific source files
- MCUXpresso SDK:
  - New IDE project should correctly setup needed preprocessor defines and include the correct SDK drivers that are unique to each device
- MCUXpresso Config Tools:
  - Open Config Tools to enable them within the new IDE project
  - Import the previously configure .mex file from the previous project to apply as best possible the pin and clock settings

How to Change Device Part Number?

Option 2 – Edit existing project: Harder, but might be necessary…

- MCUXpresso IDE:
  - Change the target MCU in the IDE project settings
  - Update device header includes in application specific source files
- MCUXpresso SDK:
  - Change the preprocessor defines within IDE project settings to match the device and package
  - Replace all SDK drivers with equivalent ones from the correct SDK download.
  - Replace CMSIS header files, startup files, utilizes with equivalent ones from the correct SDK.
- MCUXpresso Config Tools:
  - The Config Tools will present error in detect project setting and can auto correct to the new device.
  - As needed use the “switch package” icon from within the Pins Tool to select a different package
  - Config Tools will attempt to transfer all compatible pin muxing settings to the new package

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

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

Supported Devices

- Supported Devices Table (Community Doc)

Hands-On Lab:

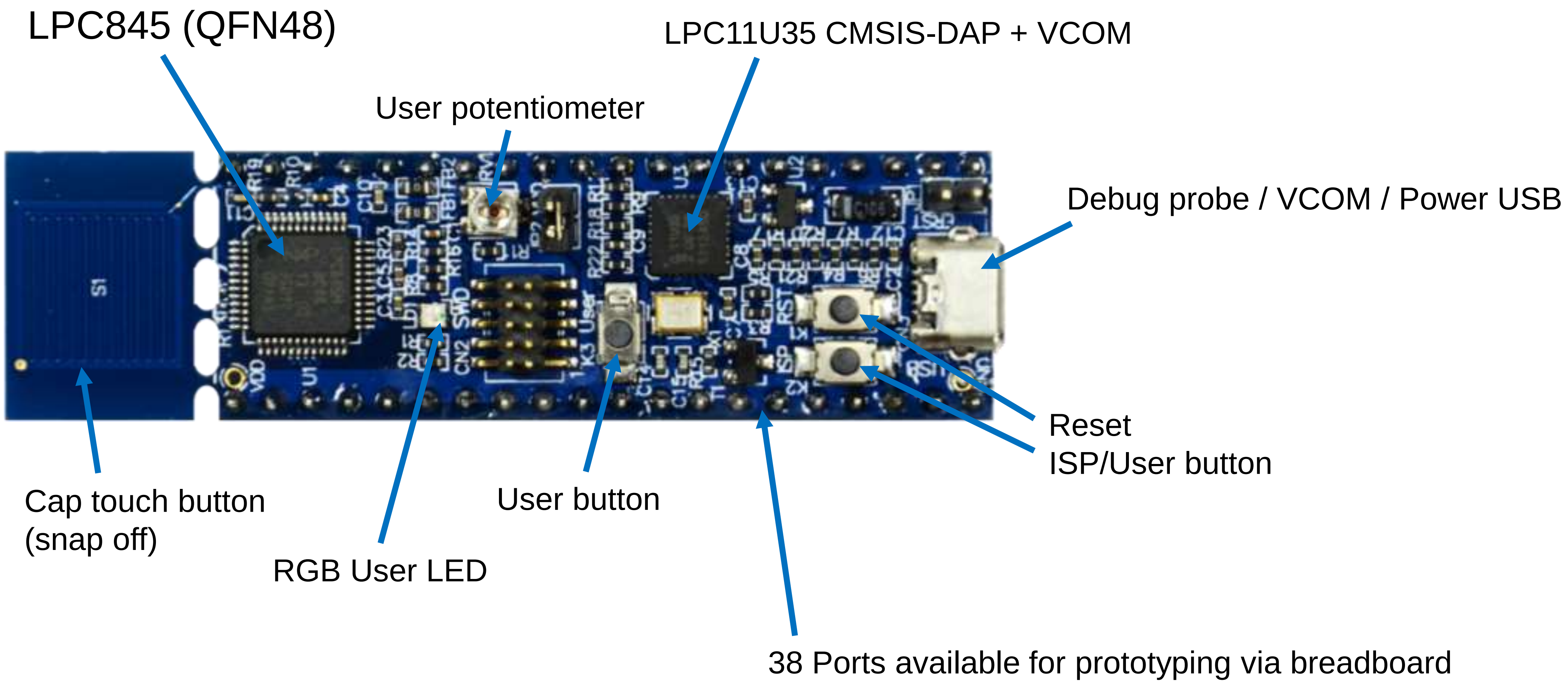
Developing SDK Support for Your Hardware

Creating a MCUXpresso Config Tool configuration file (.mex)

Applying configuration to a new MCUXpresso IDE project

Porting existing MCUXpresso SDK Example project to your configuration

LPC845 Breakout Board – Overview


LPC845 (QFN48)
LPC11U35 CMSIS-DAP + VCOM
User potentiometer
Debug probe / VCOM / Power USB
Reset
ISP/User button
Cap touch button
(snap off)
User button
RGB User LED
38 Ports available for prototyping via breadboard

Schematic included in back of lab handout

SECURE CONNECTIONS
FOR A SMARTER WORLD