

# CREATING A CUSTOM BOARD MCUXPRESSO SDK

AMF-SOL-T3986

BRENDON SLADE  
MCU ECOSYSTEM TEAM  
MARCH 2020



PUBLIC



SECURE CONNECTIONS  
FOR A SMARTER WORLD

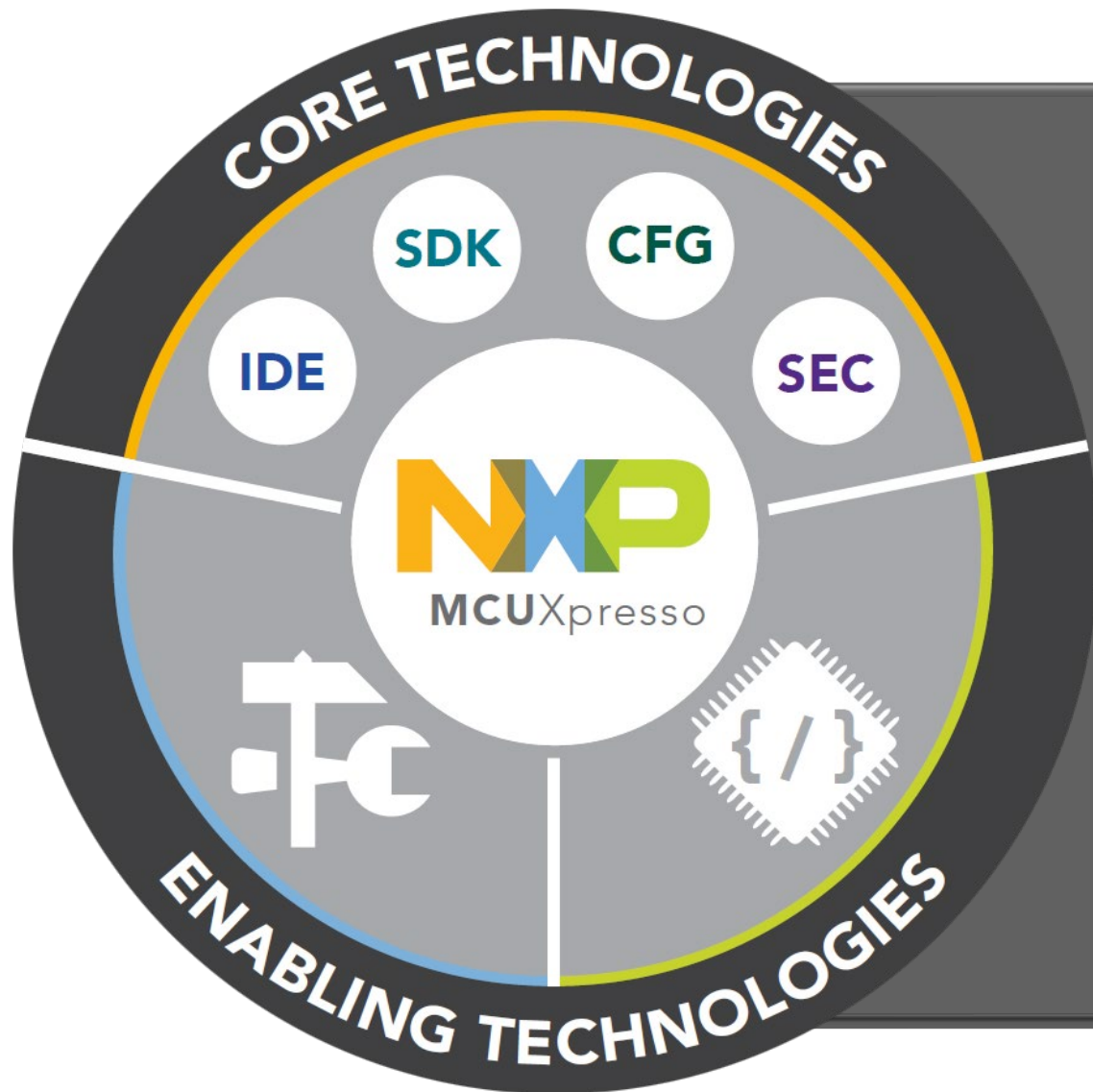
# Course pre-requisites/notes

- This course is an update of a 2019 Tech Day course on creating a custom SDK (AMF-SOL-T3532)
  - Updates include new functionality provided for custom SDK creation, first introduced in MCUXpresso IDE 11.0
- The first section of this course for Tech Day hands-on sessions would include introductions to the MCUXpresso suite of tools
  - These introductory notes are also covered in AMF-SOL-T3991 “Understanding MCUXpresso IDE and Configuration tools”
  - These are not included in this training slide set

# Content

- Recap of MCUXpresso SW and Tools
- MCUXpresso Config Tools overview
- Changing package and part types
- Overview of the Custom SDK wizard
- Lab/demo sessions:
  - Creating a board configuration
  - Creating a Custom Board SDK
  - Using a Custom Board SDK with the New Project Wizard
  - Applying a board configuration to new and existing projects

# The MCUXpresso Ecosystem



## Core Technologies from NXP:

- MCUXpresso IDE
- MCUXpresso SDK
- MCUXpresso Config Tools
- MCUXpresso Secure Provisioning Tool

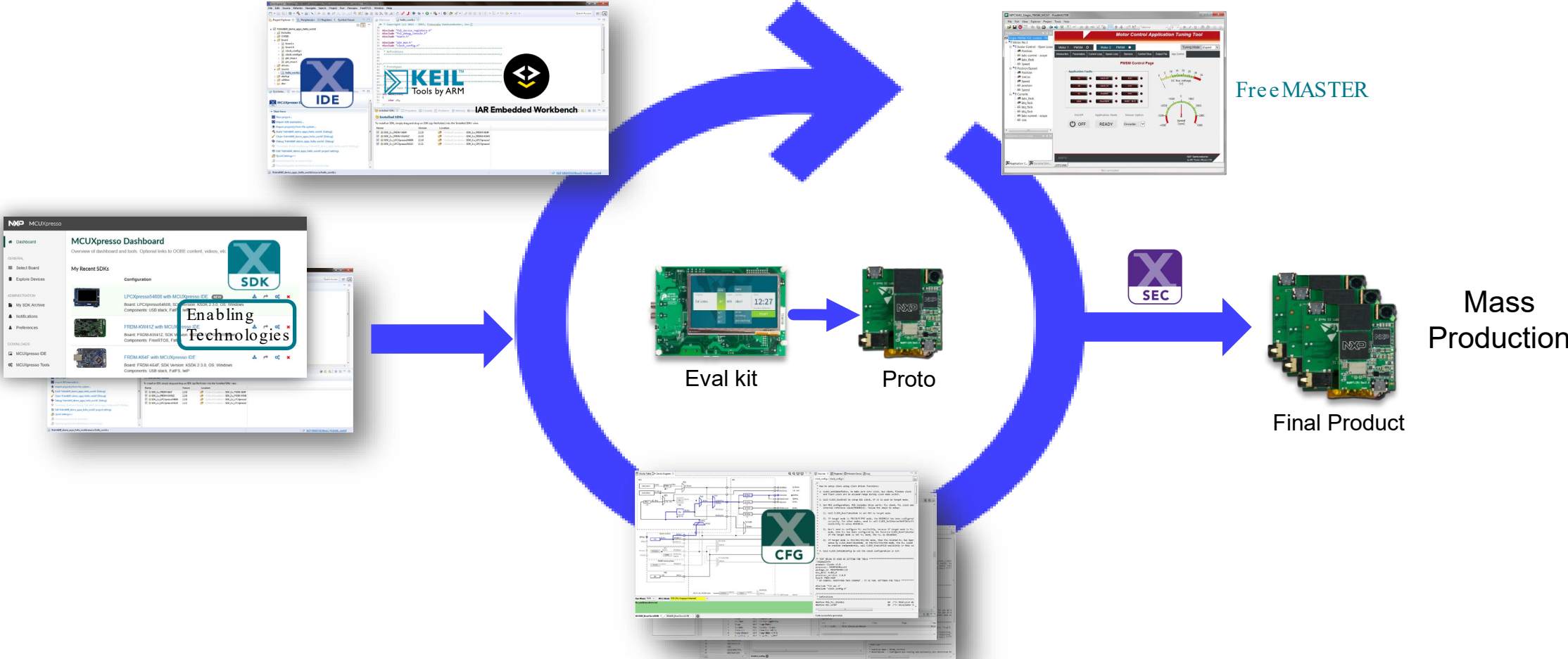
## Enabling Software Technologies:

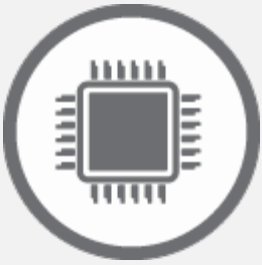
- Run time software libraries and middleware
- Enable customers to focus on differentiation
- From NXP and partners

## Enabling Tools Technologies:

- Partner IDEs
- Debug Probes
- Development Boards
- From NXP and partners

# Streamlined MCUXpresso development flow





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

The screenshot displays the MCUXpresso Pins Configuration tool interface. It features a main pin assignment table, a graphical package view, and a routed pins table.

Pin	Pin name	Label	Identifier	GPIO	UART	FT
7	PT16/SPI_PCS3...	I31Q1/SD_CAR...	SDHC_CD	PT16	UART3_CTS_b	FT
10	USBD_DP	J212/KM4_MIC...	USER_DP			
11	USBD_DM	J212/KM4_MIC...	USER_DM			
28	XTAL32	Y311/XTAL32_R...	XTAL32K			
29	EXTAL32	Y312/EXTAL32...	EXTAL32K			
31	ADDC_SE18/PTE...	J218/AB16/J2C...	ACCEL_SCL	PT24	UART4_TX	
32	ADDC_SE18/PTE...	J218/AB16/J2C...	ACCEL_SDA	PT25	UART4_RX	
33	PT26	J211/D124/L1E...	LED_GREEN	PT26	UART4_CTS_b	
38	PTA14/LWU_P3...	SW3	SW3	PTA4		
39	PTA5/USE_C1E...	U1317/RMID_R...	RMID_RXER	PTA5		
42	CMR2_IN0/PTA...	U1312/RMID_R...	RMID_RXD0	PTA2		
43	CMR2_IN0/PTA...	U1313/RMID_R...	RMID_RXD0	PTA13		
44	PTA14/SPI_PCS...	U1315/RMID_...	RMID_CRS_DV	PTA14	UART0_TX	
45	PTA15/SPI_SCK...	U1319/RMID_T...	RMID_TXEN	PTA15	UART0_RX	
46	PTA16/SPI_SO...	U1320/RMID_T...	RMID_TXD0	PTA16	UART0_CTS_b...	
47	ADDC_SE18/PTE...	U1313/RMID_T...	RMID_TXD0	PTA17	UART0_RTS_b	
50	EXTAL0/PTA18...	U1316/RMID_R...	EXTAL0/RMID_RX...	PTA18		
53	ADDC_SE18/ADC...	U1310/RMID_...	RMID_MQD0	PTB0		
54	ADDC_SE18/ADC...	U1311/RMID_...	RMID_MQD0	PTB1		
62	PTB16/SPI_SO...	U714/JUART0_RX...	DEBUG_UART_RX	PTB16	UART0_RX	
63	PTB17/SPI_SBL...	U1311/JUART0_TX...	DEBUG_UART_TX	PTB17	UART0_TX	
67	PTB21	D121/LEDRGB...	myLED	PTB21		
68	PTB22	D121/LEDRGB...	LED_RED	PTB22		
78	CMR2_IN0/PTA...	U811/SW2	SW2ACCEL_INT1	PTC6		
85	PTC13/JUART...	U819	ACCEL_INT2	PTC13	UART4_CTS_b	

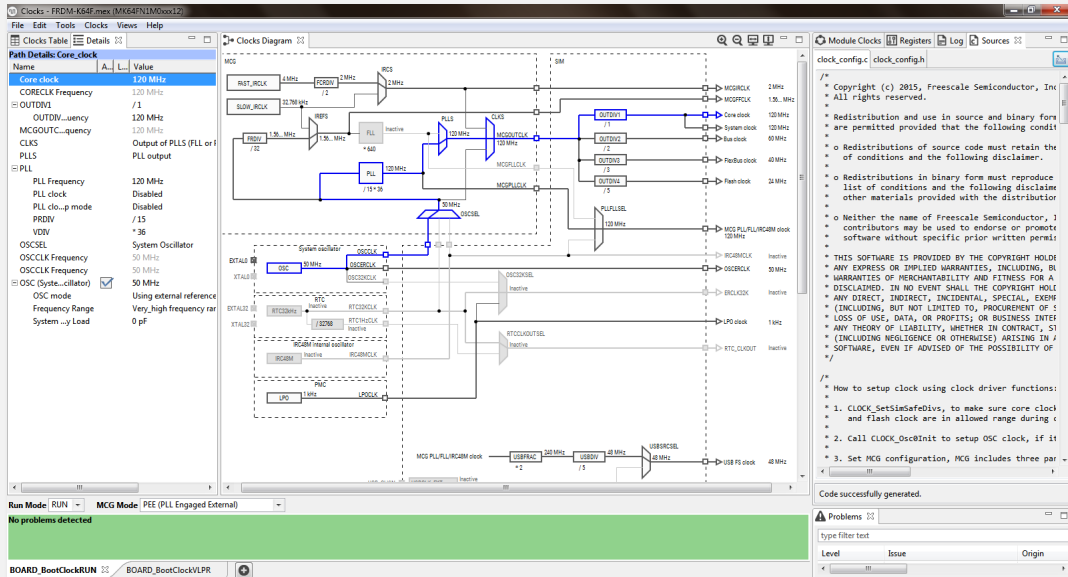
#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull select	Pull enable	Passive filter	Digital filter
67	GPIOB	GPIO_21	PTB21	D121/LEDRGB_BLUE	myLED	Output	Slow	Disabled	Low	Pulldown	Disabled	Disabled	n/a
68	GPIOB	GPIO_22	PTB22	D121/LEDRGB_RED	LED_RED	Output	Slow	Disabled	Low	Pulldown	Disabled	Disabled	n/a
33	GPIOE	GPIO_26	PT26	J211/D124/LEDRGB_GREEN	LED_GREEN	Output	Slow	Disabled	Low	Pulldown	Disabled	Disabled	n/a







## Clock configuration and diagram view



Name	Value
Core_clock	120 MHz
CORECLK Frequency	120 MHz
OUTDIV1	/1
OUTDIV1_uency	120 MHz
MCGOUTC_uency	120 MHz
CLKS	Output of PLLS (PLL or PLLS)
PLLS	PLL output
PLL	120 MHz
PLL Frequency	120 MHz
PLL clock	Disabled
PLL div_p mode	Disabled
PREV	/15
VDDV	*36
OSCSEL	System Oscillator
OSCCLK Frequency	50 MHz
OSCCLK Frequency	50 MHz
OSC (System_oscillator)	<input checked="" type="checkbox"/>
OSC mode	Using external reference
Frequency Range	Very_high frequency rat
System Load	0 pf

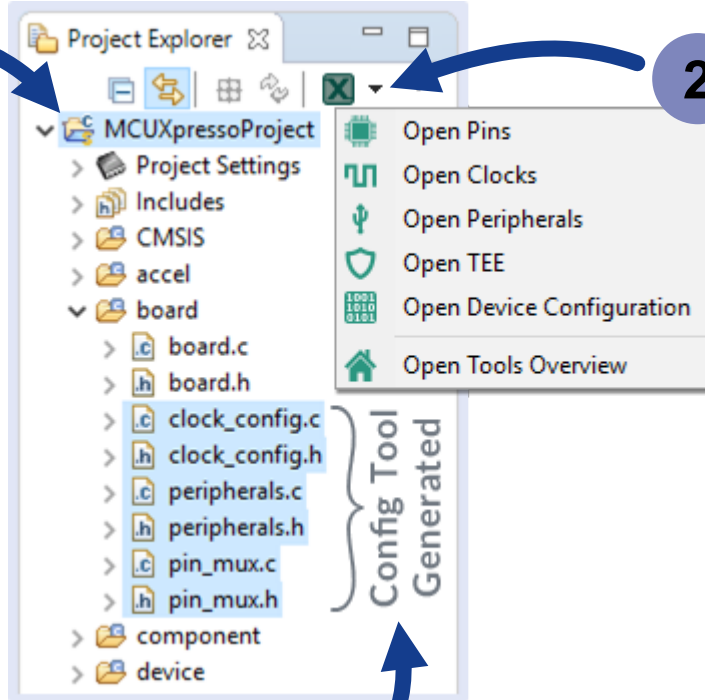
# MCUXpresso Config Tools Clock Configuration

- 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

# MCUXpresso IDE with integrated Config Tools

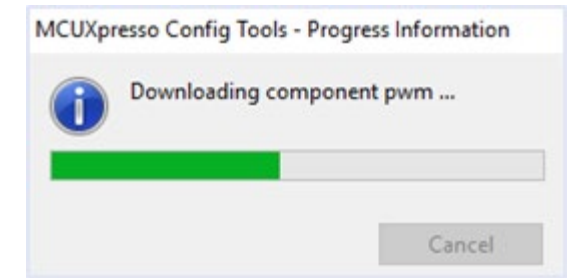


1 Use the MCUXpresso IDE Quick Start Panel to **import** an existing SDK Project or **create** a new one



2 Use the IDE project menu bar to switch to a Config Tool **perspective**

*Config Tool data will download on demand*



3 Use the Config Tools perspective to configure **Pins**, **Clocks**, and **Peripherals**



*The perspective icons can also be used to switch between config tools or return to the Development perspective*



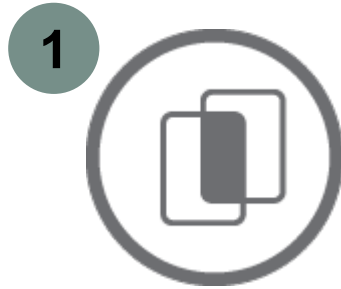
4 **Update Code** from Config Tools to write the generated files directly to the project folder and return to the Development perspective



# MCUXpresso Config Tool – Standalone Tool Workflow



The standalone **MCUXpresso Config Tools** are developed specifically for working with non-MCUXpresso IDEs



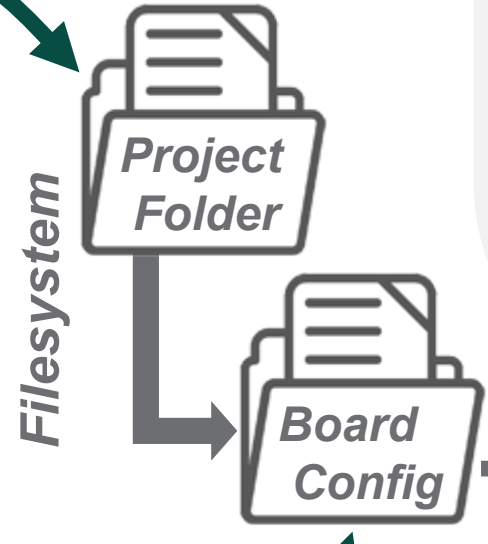
1 Use the MCUXpresso Config Tools to **clone** an existing SDK Project

Alternatively the Config Tools can be pointed to an existing project, and will create the connections

2 Use the MCUXpresso Config Tools to configure **Pins, Clocks, and Peripherals**



3 **Update Code** from Config Tools to write the generated files directly to the project folder



Generated **.MEX** file can be exported / imported to other projects to share board settings

Supporting:



# Config Tools – 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 Config Tool **does not generate projects**, but is designed to work with existing projects, however there are instances where you 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).

# 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

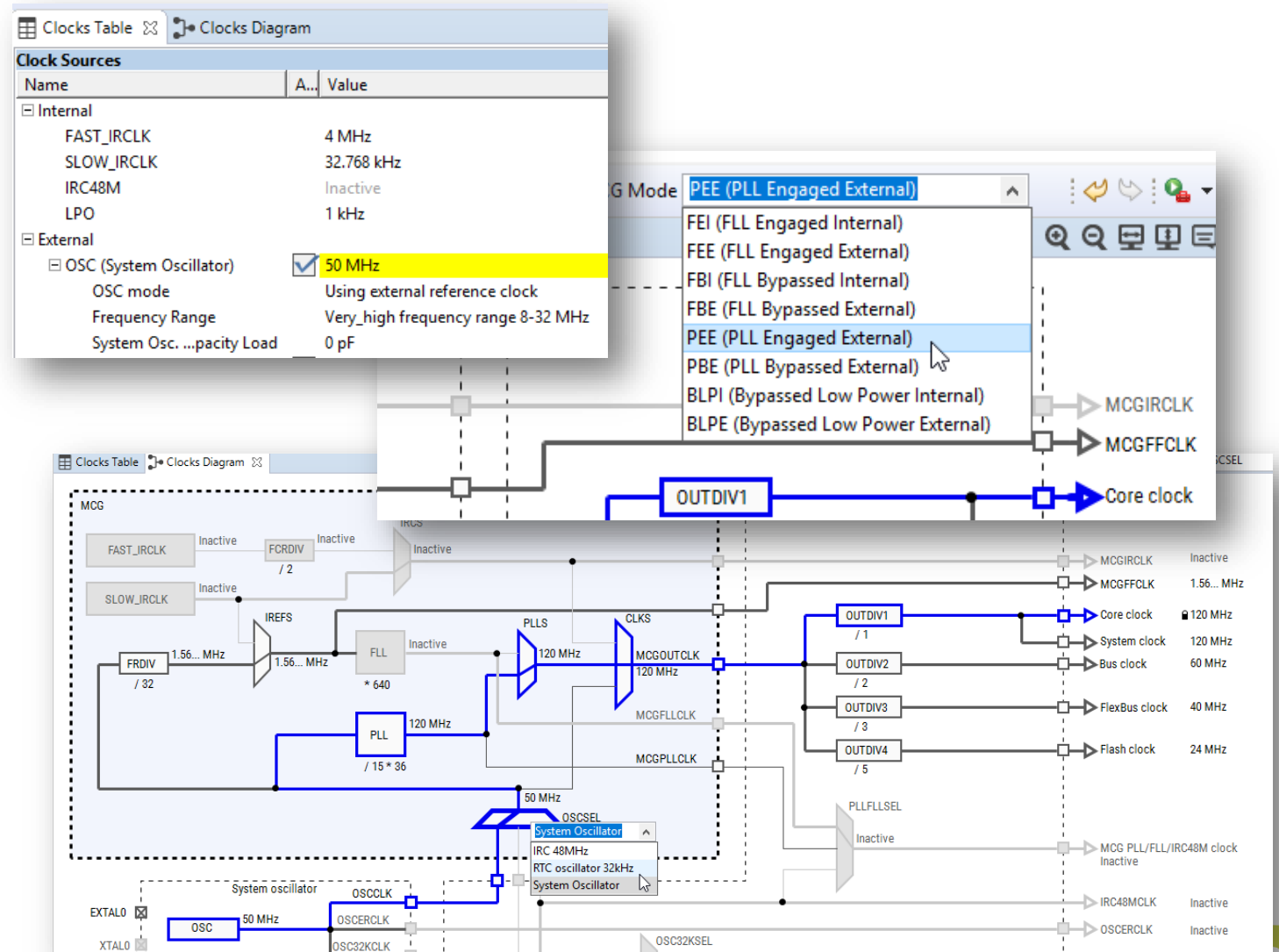
- Add new entry to “**Routed Pins Table**”

The screenshots illustrate the following steps:

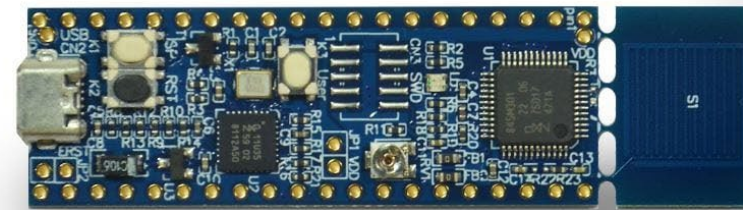
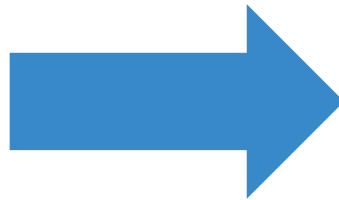
- Pins Table:** A table listing pins and their associated peripheral signals. Row 33 is highlighted, showing Pin 33 (PTE26) is configured for UART4\_CTS\_b and GPIO signals.
- Peripheral Signals:** A tree view showing peripheral categories like GPIOB, GPIOC, and GPIOD. GPIOD is expanded to show specific pin assignments.
- Pin [93] Dialog:** A dialog box titled "All signals on pin [93]:" listing various peripheral signals such as FB\_CS1\_b, FTM3\_CH0, LLWU\_P12, PTD0, SPI0\_PCS0, UART2\_RTS\_b, and (n/a, disabled).
- Routed Pins Table:** A table showing the routing of pins. A new entry is being added for GPIOA/GPIO 14, which is routed to PTE26 (J2[1]/D12[4]/LED\_RGB\_GREEN).

# 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 settings to be selected directly



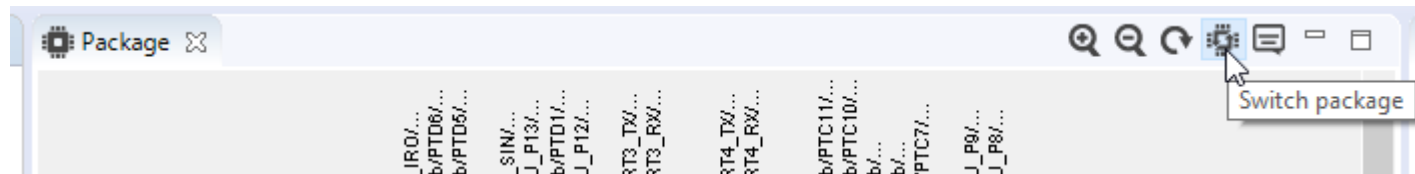
# Migrating from a standard SDK to a custom SDK



# 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
- **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 configured .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 the 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 errors in detect project setting and can auto corrected 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

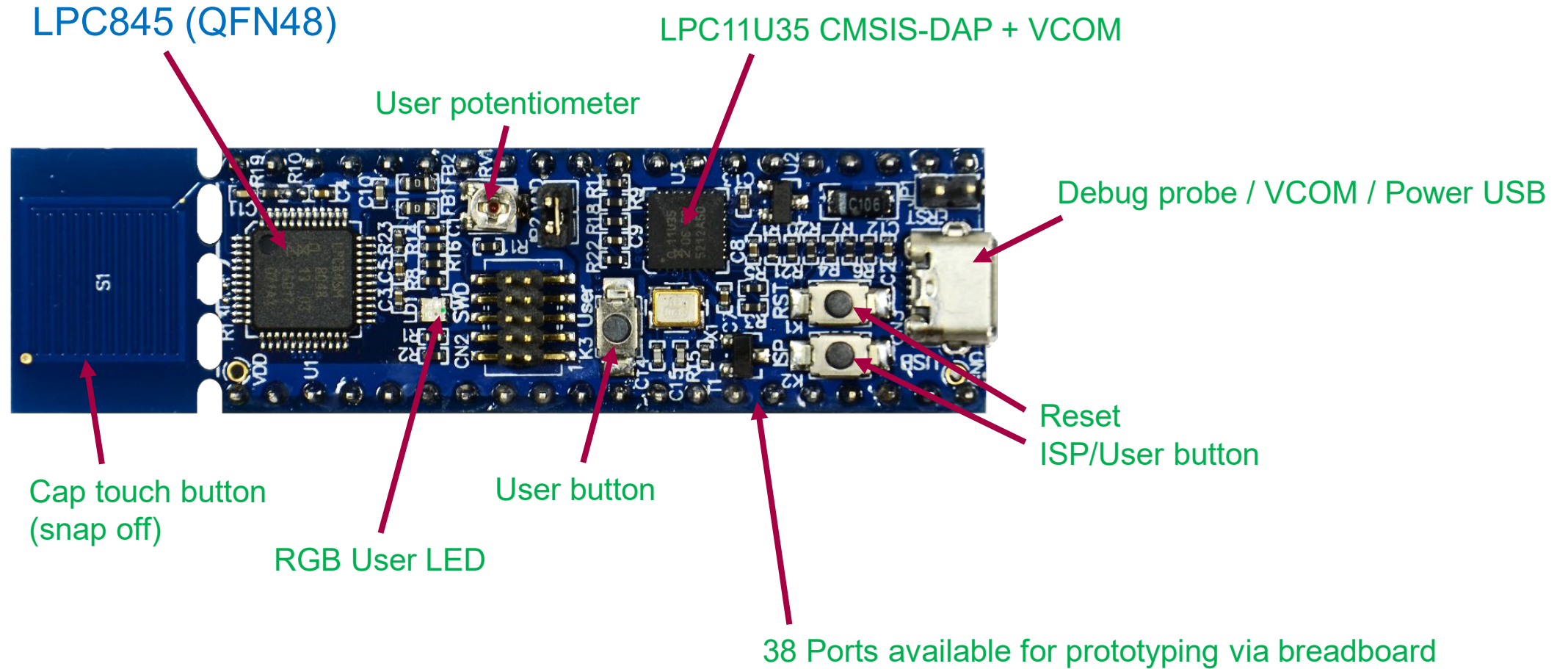
# Board SDK Wizard

- Board SDK Wizard was introduced in MCUXpresso IDE 11.1
- Starting from an existing SDK and a project customized for your board, the Wizard enables:
  - Creation of an exportable .zip SDK package that others can easily import and use
  - SDK attributes, photo and #defines to be added to improve usability
- Primary purpose:
  - Enabling creation of applications using MCUXpresso IDE New Project Wizard with pin and clock configurations included

# Example Custom SDK Creation

- Step 1: create a board configuration that includes the custom pin and clock configuration of the (custom) board being used
- Step 2: use the Custom SDK Wizard in MCUXpresso IDE to generate a custom SDK that can be imported and used by other developers in your team
- Step 3: create such a new project with the custom SDK from Step 2
- Step 4: use board configurations from Step 1 to modify a standard demo board example for use on a custom board
- These steps are described in the lab handout, and walked through in the video tutorial
- The tutorial uses the LPC845-BRK board (see next slide)
  - Although this board now has its own SDK, for the tutorial we will start from the LPC845 part SDK, as a representative flow from a standard NXP EVK

# LPC845 Breakout Board - Overview



Schematics located at: [https://www.nxp.com/downloads/en/schematics/SCH\\_LPC845\\_BRK.zip](https://www.nxp.com/downloads/en/schematics/SCH_LPC845_BRK.zip)



**SECURE CONNECTIONS  
FOR A SMARTER WORLD**