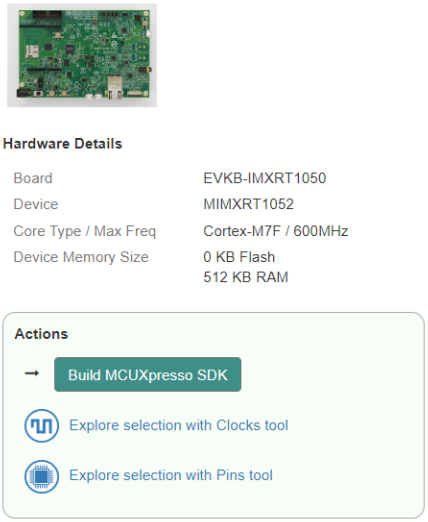
Lab 1 – Overview of Tools

- Online MCUXpresso SDK Builder
- Importing MCUXpresso SDK into MCUXpresso IDE
- Basic functionality of MCUXpresso IDE
- Enabling / Using MCUXpresso Config Tools

MCUXpresso SDK Configuration and Download

*This section will cover use of the Online MCUXpresso SDK Builder (http://mcuxpresso.nxp.com). This section is **optional** as the desired SDK has already been downloaded and installed to the training laptop.*

1. Configure and download an MCUXpresso SDK build



   1.1. Visit https://mcuxpresso.nxp.com

   1.2. For a new build click the "**Select Development Board**" action (login required)

      1.2.1. Use **Search by Name** to locate the specific board used in this lab "EVKB-IMXRT1050"

      1.2.2. Select "**EVKB-IMXRT1050**" from the Boards category, a panel of overview information will appear with additional details related to this board.

      1.2.3. Select "**Build MCUXpresso SDK**" from the panel on the right of the webpage

   1.3. Configure your SDK

      1.3.1. Explore available middleware for this development board, by selecting the "**Add software component**" action button
      Note: For this lab "Amazon FreeRTOS" is required and should be added to the build configuration.

      1.3.2. Select the **Download** or **Request Build** Action button.
      Note: The Online MCUXpresso SDK Builder features a dynamic build engine that caches SDK configurations are they are added to the system. NXP pre-caches several standard configurations when a new SDK or device is released.

   1.4. SDK Download will start after accepting the NXP End User License Agreement resulting in a .zip saved to browsers default download directory

   1.5. The MCUXpresso SDK Dashboard will list all SDK downloads associated to your account.
   Note: From this Dashboard you can redownload previous SDKs, be alerted to updates for specific configurations, modify and rebuild SDK as additional entries, and share specific configurations with colleagues or support teams.

MCUXpresso SDK Installation

This section will cover the installation of the MCUXpresso SDK into the MCUXpresso IDE. The SDK can be left as a .zip archive or extracted within the IDE UI to allow for linked projects and other benefits.
*This section is **optional** as the desired SDK has already been downloaded and installed to the training laptop.*

2. Installation of SDK

   2.1. Launch the **MCUXpresso IDE** using the short-cut located on the desktop

      2.1.1. Ensure that the "**Installed SDKs**" panel is visible
      Note: By default, the panel in located in the bottom dock position

   2.2. Locate the downloaded SDK archive (pre-downloaded on training laptops)

      2.2.1. Double-click the "**MCUXpresso SDK – Shortcut**" link located on the desktop

   2.3. Using the mouse drag the file "**SDK_2.5.0_EVKB-IMXRT1050**" from the file explorer window into the "**Installed SDK**" panel of the IDE.

      2.3.1. A confirmation dialog should appear in the IDE (you may need to minimize the File Explorer Window)

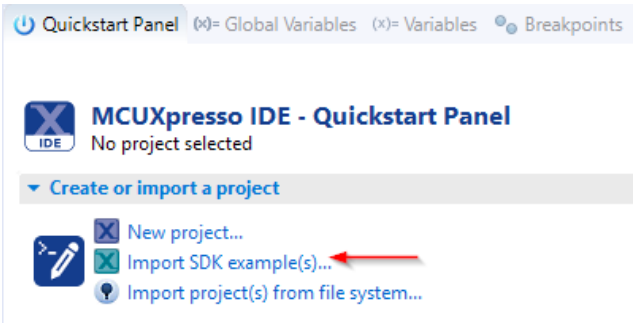
      2.3.2. Accept the SDK installation

   2.4. *Optional*: Unzip the SDK archive

      2.4.1. Right click on the newly installed SDK from the "**Installed SDKs**" panel, select "**Unzip archive**"

Cloning an example SDK Application

This section will cover importing an SDK example project from the installed SDK and running the application on a development board. Details will include project options available during the cloning process include a comparison of Semihosting vs UART as a debug console.



3. Clone the "**freertos_generic**" example application

   3.1. From the **Quickstart Panel** (lower left dock of the IDE) select "**Import SDK example(s)…**"

   3.2. Select the evkbimxrt1050 board.

      3.2.1. If needed to help filter the selection of SDK, select "MIMXRT1050" from the list of "SDK MCUs"

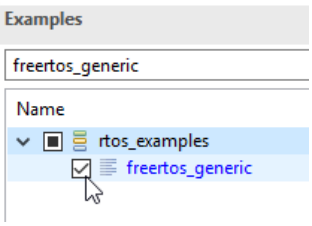
      3.2.2. Locate and select the picture of the board (the text below the picture is a hyperlink to learn more about the board on http://nxp.com

      3.2.3. Select the board selected (highlighted) select "**Next**"

   3.3. Locate and place a checkmark next to the "**freertos_generic**" example

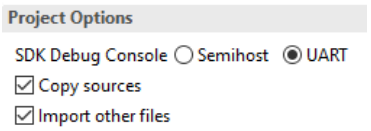
      3.3.1. Option 1) use the text filter to filter on **generic**, expand the results to find the correct example

      3.3.2. Option 2) browse the list of examples expanding the following path: **rtos_examples → freertos_ generic**



3.4. Review default selections

3.4.1. For this example we will initially be using **UART** as the Debug Console (*selected by default*)



3.5. Select **Finish**

Basic debugging with MCUXpresso IDE

This section will cover the basic steps to compile, flash, and debug an application within the IDE. Details will cover Debug Probe Discovery, Runtime Controls, and FreeRTOS Task Aware Debugging.

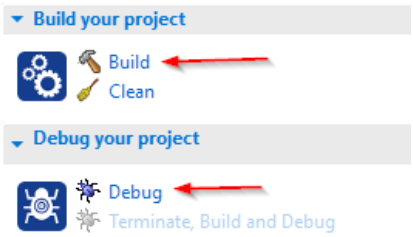
Note, there is an important step covered in this lab to properly allow Task Aware Debugging with the CMSIS-DAP debugger.

4. Build and Debug

4.1. Connect the **microUSB** cable between the laptop and the **OpenSDA** connector on the laptop (*USB connector in the upper right corner – J28)*

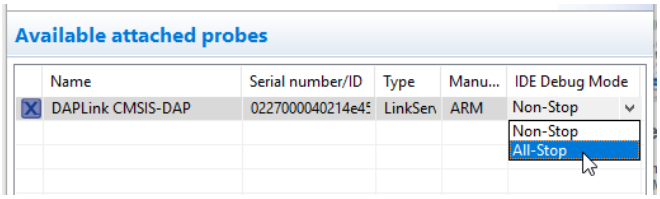
4.2. Build "**freertos_generic**" application using the "**Build**" option in the **Quickstart Panel**

4.3. Launch the debug session using the "**Debug**" option in the **Quickstart Panel**



4.3.1. Launch the debug session from the Quickstart Panel will initially open a probe discovery dialog.

4.3.2. For Task Aware Debugging you MUST change the IDE Debug Mode to "**All-Stop**"



4.3.3. Select the **CMSIS-DAP** probe, change IDE Debug Mode to **All-Stop,** and select **OK**

5. Runtime Controls

5.1. Refer to the table below for common runtime controls that are located along the top menu of the IDE

Button	Description	Keyboard Shortcut
	Restart program execution (from reset)	
	Run/Resume the program	F8
	Pause Execution of the running program	
	Terminate the debug Session	Ctrl + F2
	Clean up debug	
	Run, Pause, Terminate all debug sessions	
	Step over a C/C++ line	F6
	Step into a function	F5
	Return from a function	F7
	Step in, over, out all debug sessions	
	Show disassembled instructions	

5.2. Use the Run/Resume (play) button to start execution of the application.

With UART used for debug, you will need to open a terminal application (Tera Term / puTTY / IDE Console)

- Open Tera Term (shortcut in windows Task bar)
- Select "Serial" and locate the "mbed Serial Port" from the list (COM port may vary), Select **OK**
- Select **Setup → Serial Port** from the Tera Term menu and configure the Baud Rate to **115200**
  Note: Output message will now appears in Tera Term window

5.3. Use the Pause Execute button to pause the target application

5.4. Use the Step Over / Step In / Step Out function to explore moving in and out of the application functions.

6. FreeRTOS Task Aware Debugging

6.1. Open **Task List** and **Timer List** from the **FreeRTOS** menu (top of the IDE) and review resulting information

6.1.1. In the Task List window we see the RX and TX tasks along with other generic internal tasks like Sem, IDLE, and Tmr Svc

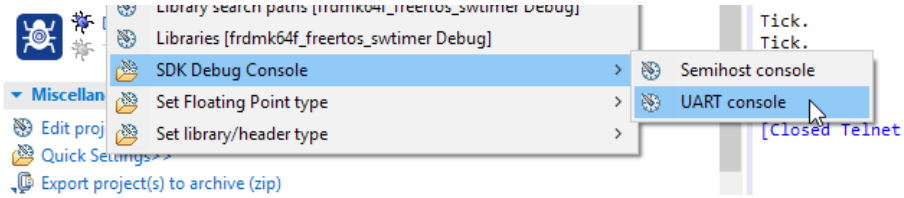
6.1.2. In the Timer List window we see one LEDTimer with a period of 200 ticks
(for this example FreeRTOS is configure with a configTICK_RATE of 200Hz, which implies a 1s timer)

7. End the debug session using the Terminate (red stop) button

***Note**: If further in this lab you ever you encounter a "There is already a running debug session…" error message it is likely that you did not properly end / terminate the previous session. Use the red stop button to end the debug session and to try launch the new debug session again.*

8. Switch the example application from UART to Semihosting Debug Console

8.1. From the Quickstart Panel, under Miscellaneous select
Quick Settings → SDK Debug Console → Semihosting console



8.2. From the Quickstart Panel, select **Debug** (which will cause a clean and build to occur)

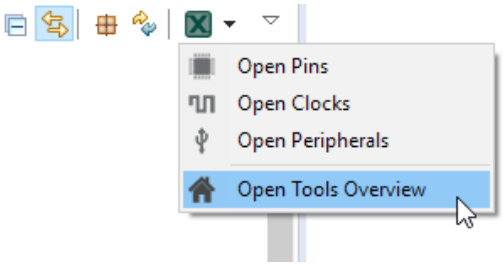
8.3. Start the application with the **Run/Resume** button
Note: Now the output now appears in the IDE Console panel

9. End the debug session using the Terminate (red stop) button.

Launching integrated MCUXpresso Config Tools

This section will cover launching the integrated MCUXpresso Config Tools for a specific IDE project. Details will cover importing of YAML within example source code and review of the Config Tool Overview screen.

10. Repeat the steps from **Sections 3 and 4** to **Import** and **Build** the "**demo_apps -> iled_blinky**" example application.
11. Launching / Enabling the MCUXpresso Config Tools
    - 11.1. Ensure that the **evkbimxrt1050_iled_blinky** application is highlighted in the Project Explorer
    - 11.2. Locate the green Config Tool dropdown icon in the Project Explorer toolbar, select Open Tools Overview

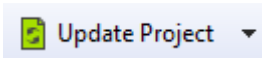


      - 11.2.1. If needed, updated Config Tool data will be downloaded from the internet
      - 11.2.2. The config tool related source code in the example application will be processed and YAML comments in the code will be used to populate the config tools with the applicable settings
      - 11.2.3. An overview tool will display a current summary of the Config Tools

Note: Alternatively you can launch directly into a specific tool (Pin, Clock, Peripheral) using the associated drop-down menu or selecting one of the corresponding perspectives icons in the upper right.

12. Explore Pins and Clock perspectives.

    Note: In this example application only a single GPIO pin is muxed, which is used for the blinking LED. In the next lab we will use the Pin and Peripheral tool to fully configure a GPIO for Input and Output.

13. Exiting the Config Tool perspective and return to the Development perspective.
    There are two primary ways to exit the Config Tools and return to the typical IDE development perspective.
    - 13.1. Apply the updated configuration, by selecting the "Update Project" tool bar item 

      This option will write the generated Config Tool source files directly to the IDE project
    - 13.2. Use the Develop Perspective icon to return to the default IDE view. 
      This option will not apply any Config Tool settings to the source files, but the settings will be preserved when switching between the perspectives.