University of Colorado Boulder

PRINCIPLES OF EMBEDDED SYSTEMS

Getting Started with KL25Z on MCUXpresso IDE

-By Shreya Chakraborty 9/10/19

INDEX

INTRODUCTION TO IDE AND KL25Z

What is an IDE?

An integrated development environment (IDE) is a software suite that consolidates all tools required to write and test software. As developers we use numerous tools throughout the software development process. An IDE includes all these tools such as text editors, compilers, debugger, code libraries such as vendor specific APIS, test platforms and so on. Without IDE we would have had to manage all these tools ourselves. Hence an IDE can improve productivity as we do not spend time deciding which tools to use. It also standardizes the development process. An IDE can help us in the following ways:

1. **Syntax Highlighting** : IDE knows the syntax of your language and prevents the syntactical errors by highlighting it, saving you some debugging time.
2. **Autocomplete :** It can anticipate what you type next, you do not need to write the entire name of the function , it automatically pops up. This saves keystrokes so the programmer can focus on logic in their code.
3. **Builds executables** : You do not need to make a Makefile. It automatically builds the code for you saves it in relevant directories in your project.
4. **Debugging** : You do not need external debuggers, logic analyzers or oscilloscopes, to see what's exactly happening with the code. The IDE provides you with a detailed debugging capability allowing you to use breakpoints, watchpoints, view variables, register values, disassembly (more on this later) and so on.
5. **Version Control, Multiple language support, multiple board support, refactoring etc**..

A little about MCUXpresso..

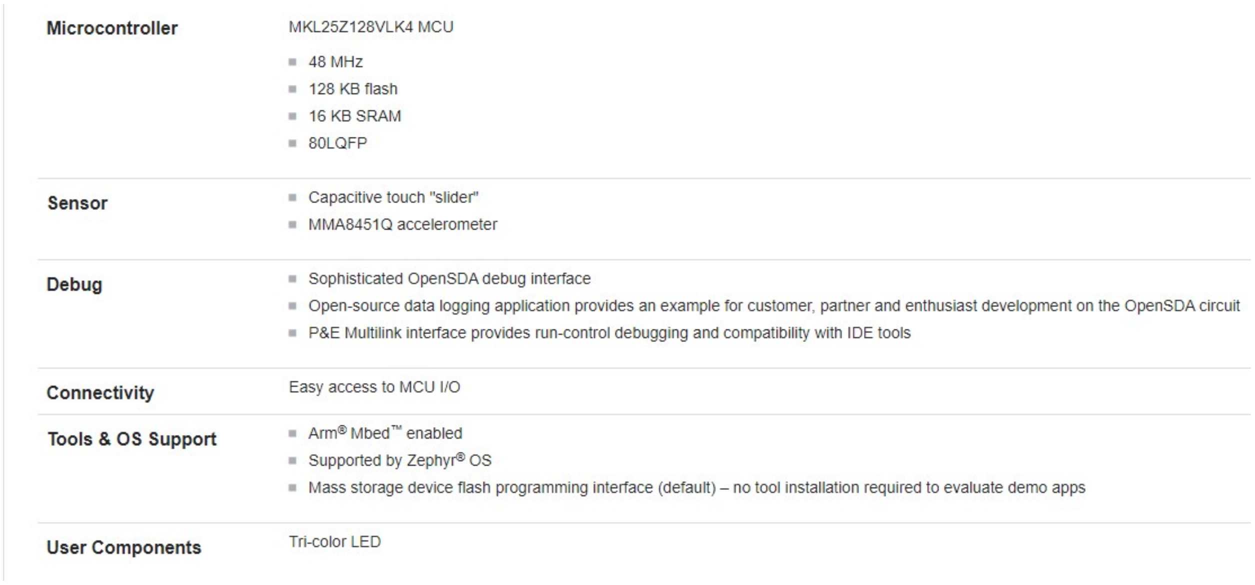
It is an Eclipse based IDE for NXP MCUs based on Arm Cortex-M cores, including LPC and Kinetis microcontrollers and so on. Provides advanced editing, compiling and editing with MCU-specific debugging views, code trace, and profiling, includes pin, clock and peripheral tools, support for FreeRTOS etc. More info in this link ->(https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE)

For a detailed guide of MCUXpresso IDE visit the link -> https://www.nxp.com/docs/en/user-guide/MCUXpresso_IDE_User_Guide.pdf

A little about FRDM KL25Z

Originally developed by Freescale for their Kinetis L series, was acquired by NXP semiconductors in 2015. It's built on built on 32 bit ARM® Cortex™-M0+ processor running at 48MHz.

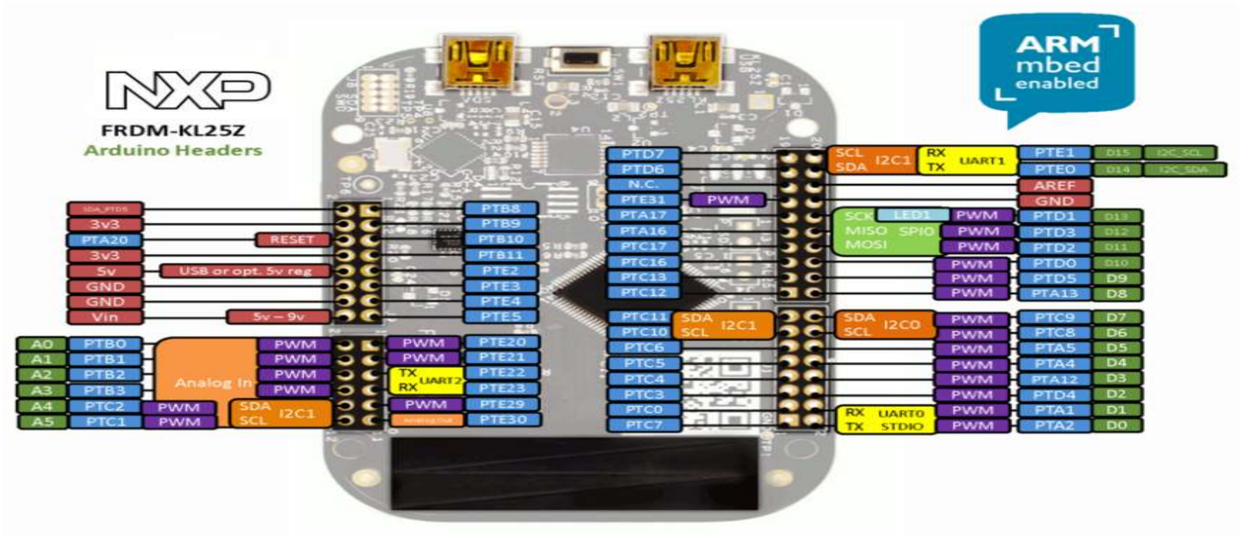
Specification:

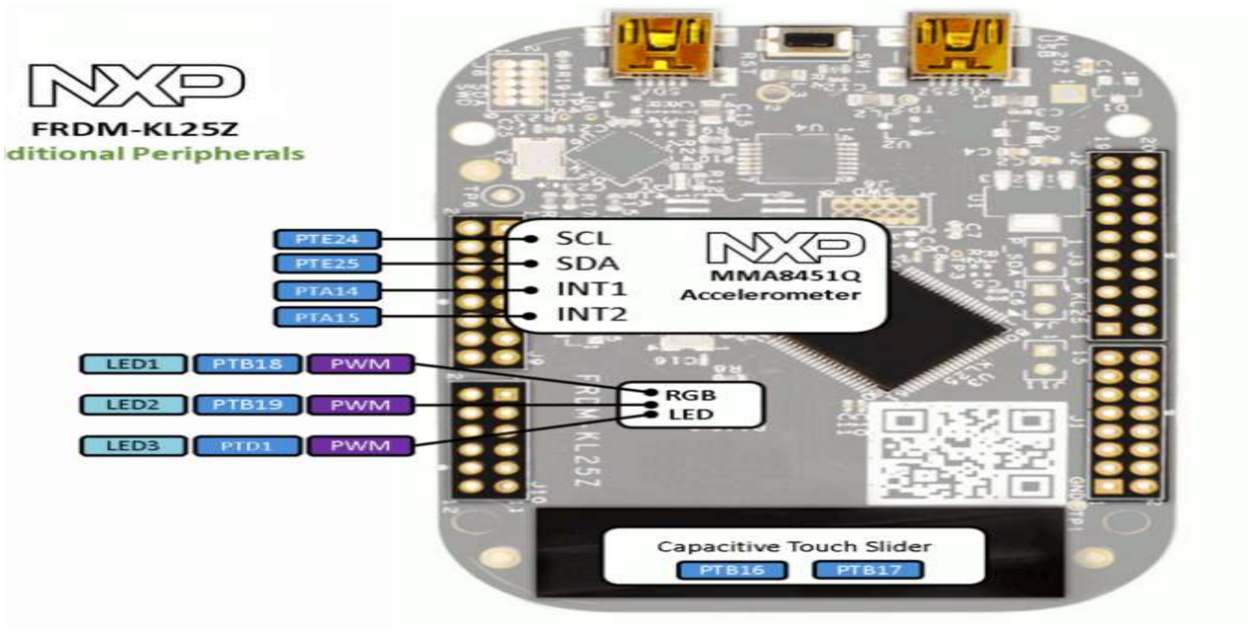


Microcontroller	MKL25Z128VLK4 MCU ▪ 48 MHz ▪ 128 KB flash ▪ 16 KB SRAM ▪ 80LQFP
Sensor	▪ Capacitive touch "slider" ▪ MMA8451Q accelerometer
Debug	▪ Sophisticated OpenSDA debug interface ▪ Open-source data logging application provides an example for customer, partner and enthusiast development on the OpenSDA circuit ▪ P&E Multilink interface provides run-control debugging and compatibility with IDE tools
Connectivity	Easy access to MCU I/O
Tools & OS Support	▪ Arm® Mbed™ enabled ▪ Supported by Zephyr® OS ▪ Mass storage device flash programming interface (default) – no tool installation required to evaluate demo apps
User Components	Tri-color LED

and lots of interfaces including USB Host, USB Device, SPI, I2C, ADC, DAC, PWM, Touch Sensor and other I/O interfaces.

Pinout :





Notice the pinouts given. You can see that majority of the pins are multiplexed! They have 2 or more functionality. We need to choose which functionality to use.

Link to the User Manual -> https://www.seeedstudio.com/document/pdf/FRMD-KL25Z.pdf

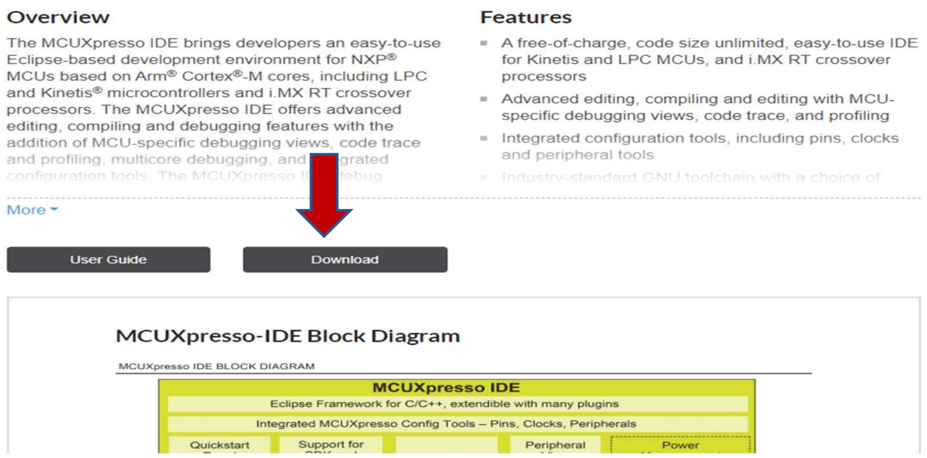
Link for the reference manual/data sheets -> https://www.nxp.com/docs/en/reference-manual/KL25P80M48SF0RM.pdf

Let's get started!

MCUXpresso Installation instructions for KL25Z

Go to this link and click download

https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE



Overview

The MCUXpresso IDE brings developers an easy-to-use Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores, including LPC and Kinetis® microcontrollers and i.MX RT crossover processors. The MCUXpresso IDE offers advanced editing, compiling and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools. The MCUXpresso IDE debug

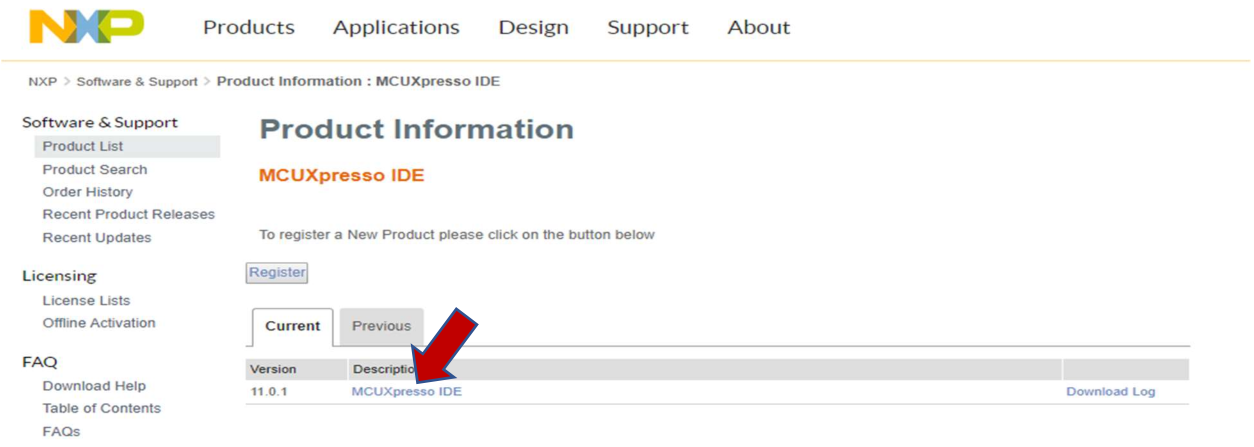
Features

- A free-of-charge, code size unlimited, easy-to-use IDE for Kinetis and LPC MCUs, and i.MX RT crossover processors
- Advanced editing, compiling and editing with MCU-specific debugging views, code trace, and profiling
- Integrated configuration tools, including pins, clocks and peripheral tools
- Industry-standard GNU toolchain with a choice of

More
User Guide
Download
MCUXpresso-IDE Block Diagram
MCUXpresso IDE BLOCK DIAGRAM
MCUXpresso IDE
Eclipse Framework for C/C++, extendible with many plugins
Integrated MCUXpresso Config Tools – Pins, Clocks, Peripherals
Quickstart
Support for
Peripheral
Power

This will prompt you to register if you are a first time users, sign in if you already have an account

Next Click on MCUXpresso IDE under current .



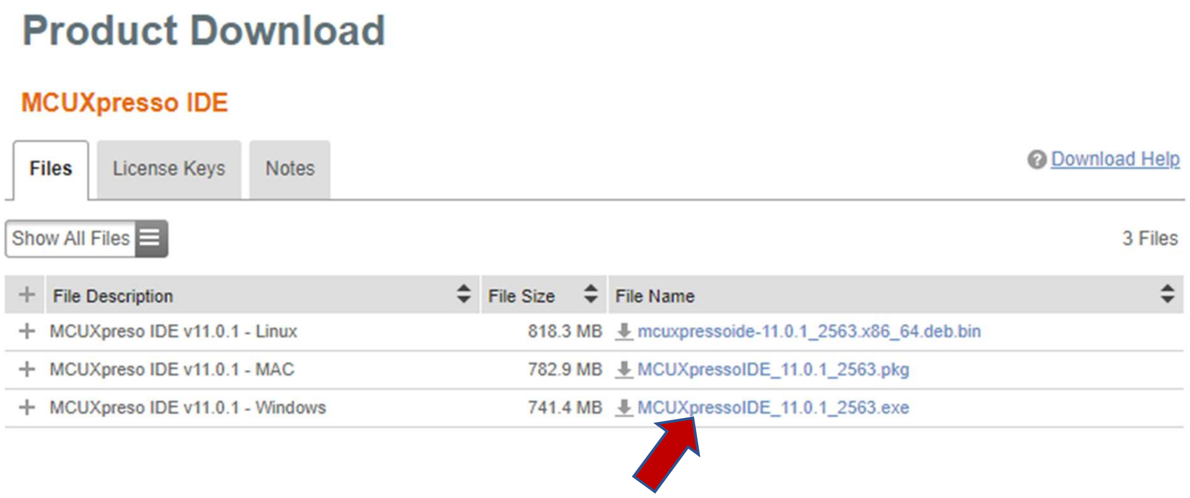
Product Information

MCUXpresso IDE

To register a New Product please click on the button below

Version	Description	
11.0.1	MCUXpresso IDE	Download Log

Agree to the software terms and conditions, and then select the package based on your operating system. For our purposes we will download for windows.

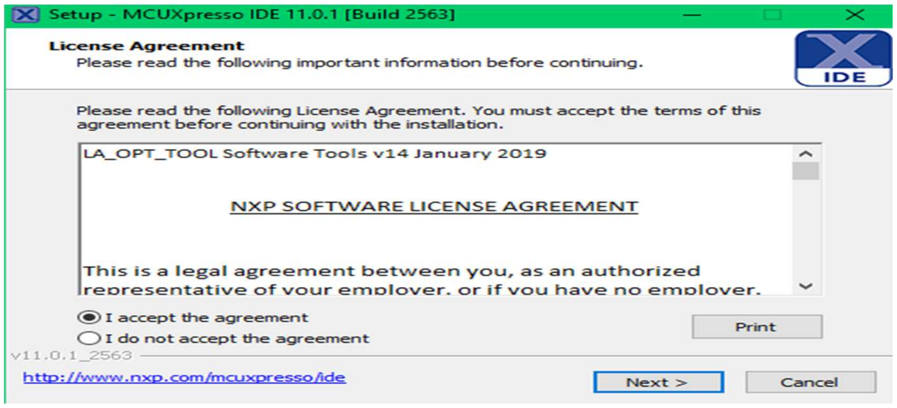


Product Download

MCUXpresso IDE

File Description	File Size	File Name
MCUXpreso IDE v11.0.1 - Linux	818.3 MB	mcuxpressoide-11.0.1_2563.x86_64.deb.bin
MCUXpreso IDE v11.0.1 - MAC	782.9 MB	MCUXpressoIDE_11.0.1_2563.pkg
MCUXpreso IDE v11.0.1 - Windows	741.4 MB	MCUXpressoIDE_11.0.1_2563.exe

It takes around 5 mins to download the exe file. When you click on the exe, it will take you to IDE setup.



Setup - MCUXpresso IDE 11.0.1 [Build 2563]

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

LA_OPT_TOOL Software Tools v14 January 2019

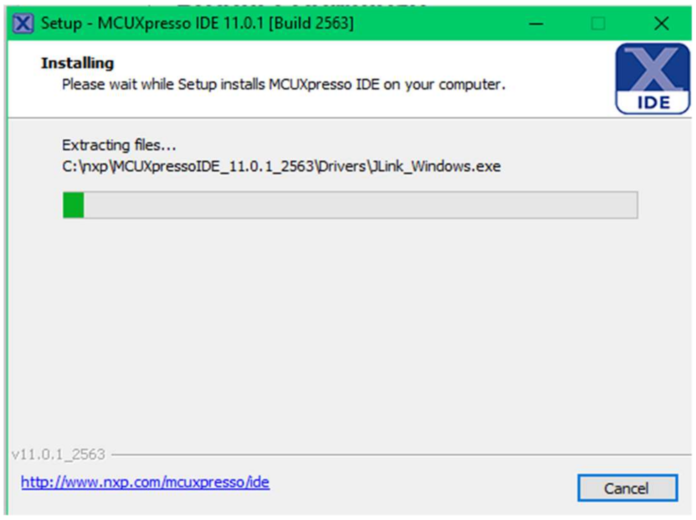
NXP SOFTWARE LICENSE AGREEMENT

This is a legal agreement between you, as an authorized representative of your employer, or if you have no employer,

[✓ I accept the agreement]
I do not accept the agreement

v11.0.1_2563
http://www.nxp.com/mcuxpresso/ide

Keep on clicking next without changing anything and then click install and finally finish!!

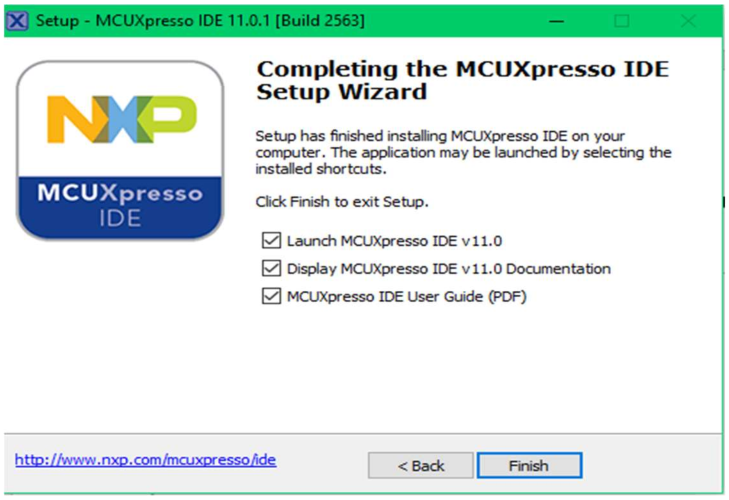


Setup - MCUXpresso IDE 11.0.1 [Build 2563]

Installing
Please wait while Setup installs MCUXpresso IDE on your computer.

Extracting files...
C:\nxp\MCUXpressoIDE_11.0.1_2563\Drivers\JLink_Windows.exe

v11.0.1_2563
http://www.nxp.com/mcuxpresso/ide



Setup - MCUXpresso IDE 11.0.1 [Build 2563]

Completing the MCUXpresso IDE Setup Wizard

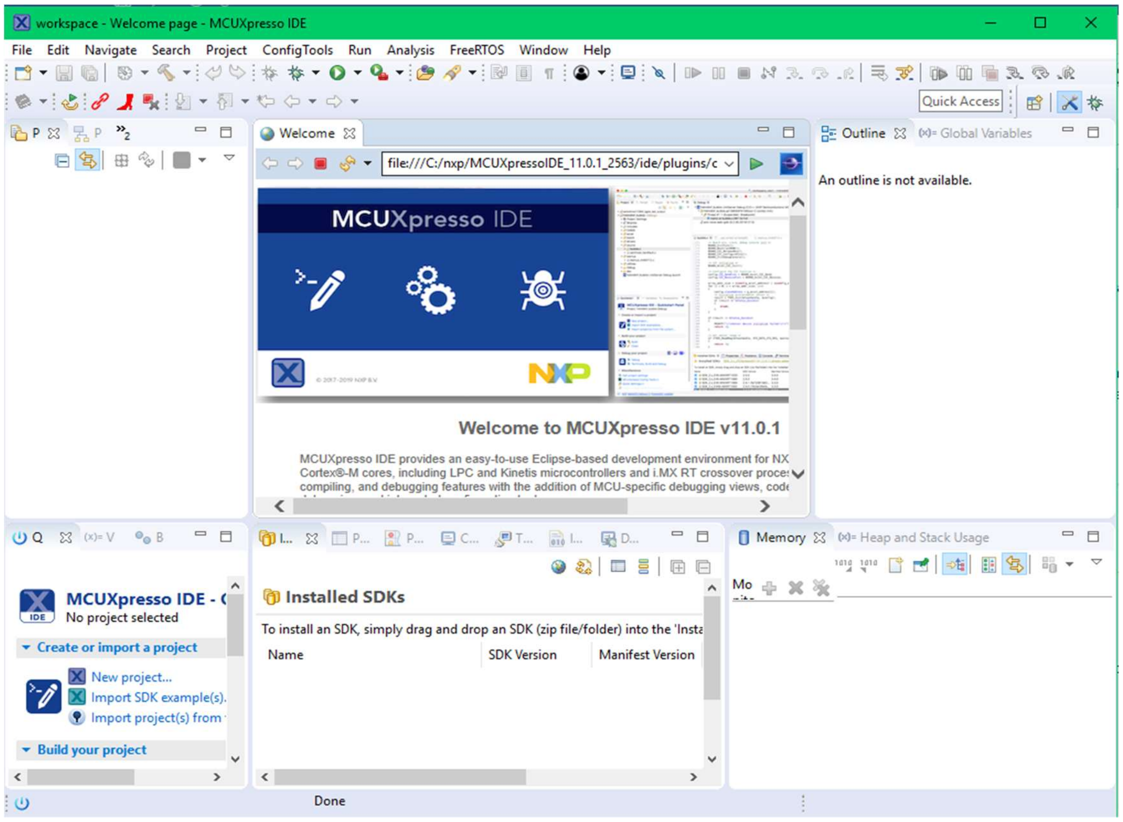
Setup has finished installing MCUXpresso IDE on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

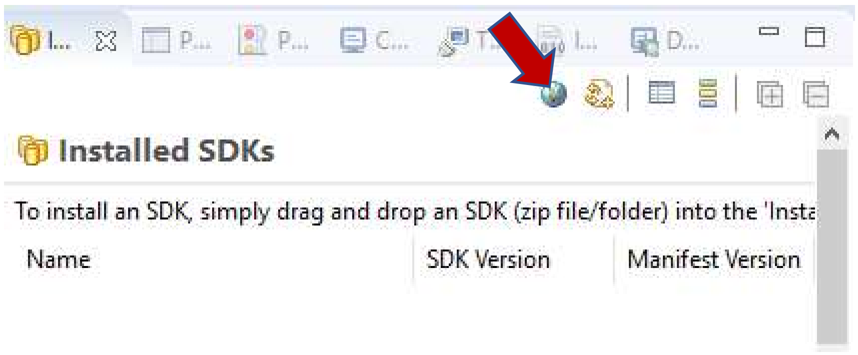
[✓ Launch MCUXpresso IDE v11.0]
[✓ Display MCUXpresso IDE v11.0 Documentation]
[✓ MCUXpresso IDE User Guide (PDF)]

http://www.nxp.com/mcuxpresso/ide

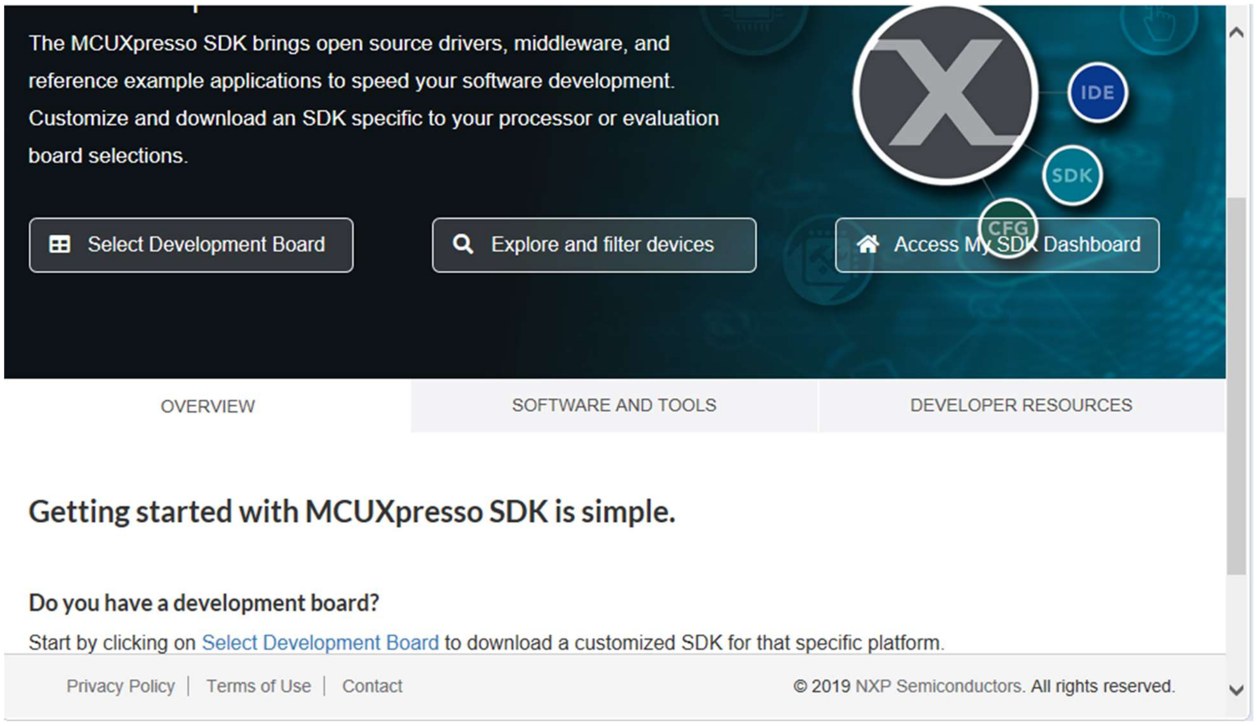
On successful installation, you see the MCU user guide automatically open up. For any security alert that you may come across during the installation process, allow access. And at this point we are done installing.


workspace - Welcome page - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Quick Access
Welcome
file:///C:/nxp/MCUXpressoIDE_11.0.1_2563/ide/plugins/c
Outline
Global Variables
An outline is not available.
MCUXpresso IDE
Welcome to MCUXpresso IDE v11.0.1
MCUXpresso IDE provides an easy-to-use Eclipse-based development environment for NX Cortex®-M cores, including LPC and Kinetis microcontrollers and i.MX RT crossover proces compiling, and debugging features with the addition of MCU-specific debugging views, code
MCUXpresso IDE -
No project selected
Create or import a project
New project...
Import SDK example(s).
Import project(s) from
Build your project
Installed SDKs
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Insta
Name
SDK Version
Manifest Version
Memory
Heap and Stack Usage
Done

We have the IDE, now its time to get the SDK. We can observe that there are no SDKs under installed SDKs in the above example. To download it, go to the tiny blue globe icon on the right of installed sdks


Installed SDKs
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Insta
Name
SDK Version
Manifest Version

It opens the following tab, alternately you can also go to this link : https://mcuxpresso.nxp.com/en/welcome


The MCUXpresso SDK brings open source drivers, middleware, and reference example applications to speed your software development. Customize and download an SDK specific to your processor or evaluation board selections.

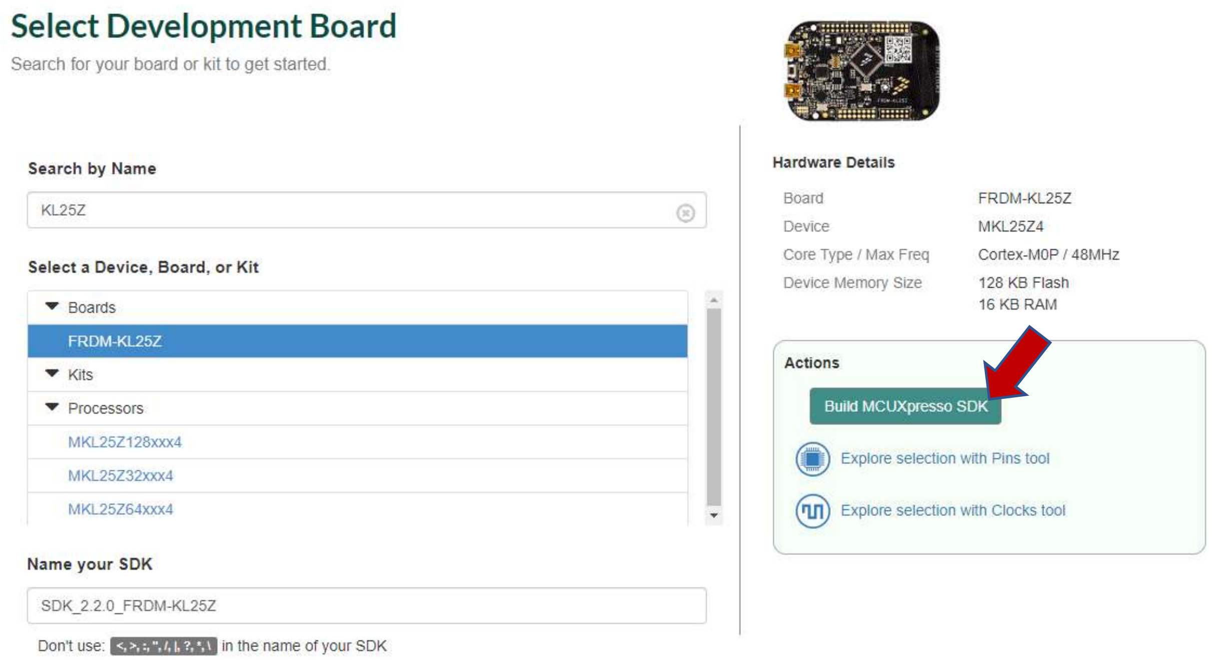
Getting started with MCUXpresso SDK is simple.

Do you have a development board?

Start by clicking on Select Development Board to download a customized SDK for that specific platform.

© 2019 NXP Semiconductors. All rights reserved.

Click on ‘select development board’, after signing in with you account, you see the following display:


Select Development Board

Search for your board or kit to get started.

Search by Name: KL25Z

Select a Device, Board, or Kit

- Boards
  - FRDM-KL25Z
- Kits
- Processors
  - MKL25Z128xxx4
  - MKL25Z32xxx4
  - MKL25Z64xxx4

Name your SDK: SDK_2.2.0_FRDM-KL25Z

Don't use: <>:;",/,|,?,*,\ in the name of your SDK

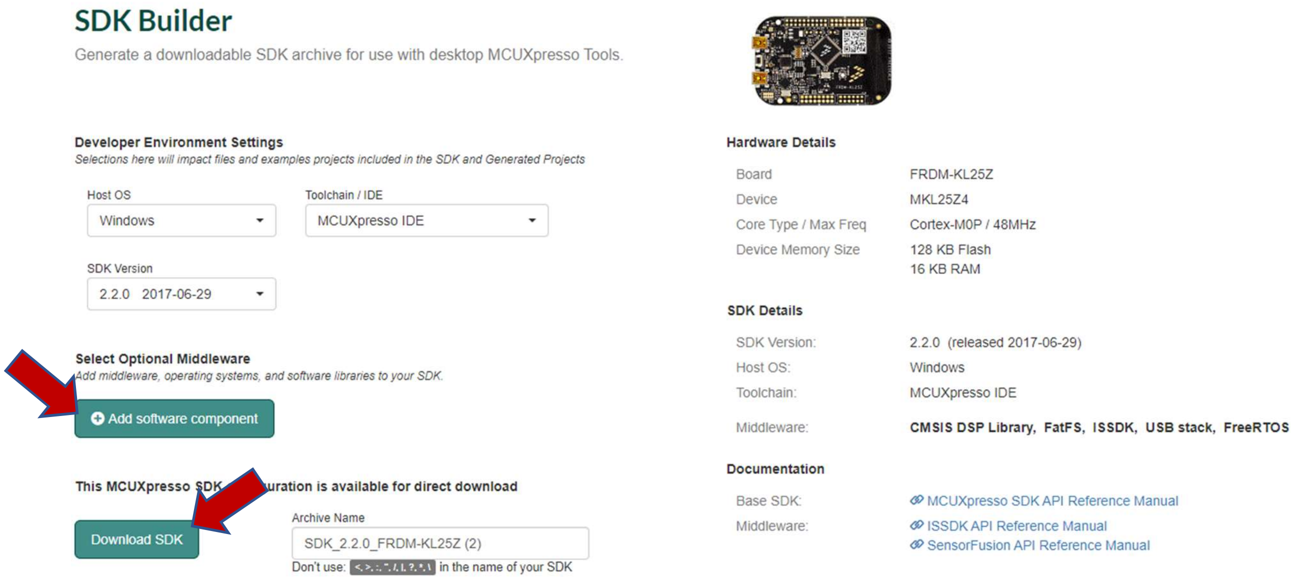
Hardware Details

Board	FRDM-KL25Z
Device	MKL25Z4
Core Type / Max Freq	Cortex-M0P / 48MHz
Device Memory Size	128 KB Flash 16 KB RAM

Actions
Build MCUXpresso SDK
Explore selection with Pins tool
Explore selection with Clocks tool

Click on MCUXpresso SDK and it you the following page.

.



SDK Builder

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here will impact files and examples projects included in the SDK and Generated Projects

Host OS	Toolchain / IDE
Windows	MCUXpresso IDE

SDK Version: 2.2.0 2017-06-29

Select Optional Middleware

Add middleware, operating systems, and software libraries to your SDK.

This MCUXpresso SDK configuration is available for direct download

Archive Name: SDK_2.2.0_FRDM-KL25Z (2)

Don't use: < > : " / \ | ? * \ in the name of your SDK

Hardware Details

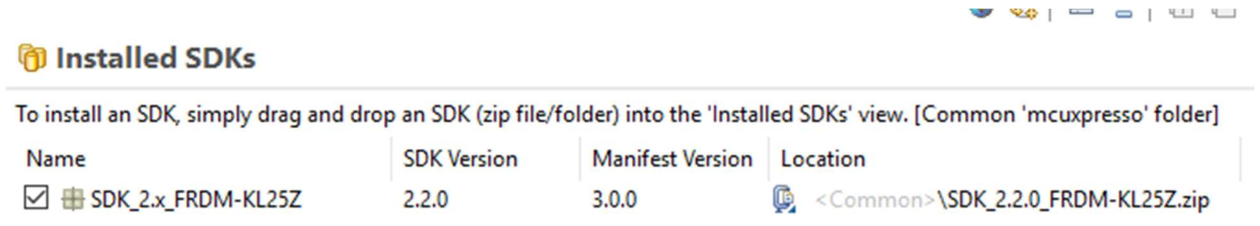
Board	FRDM-KL25Z
Device	MKL25Z4
Core Type / Max Freq	Cortex-M0P / 48MHz
Device Memory Size	128 KB Flash 16 KB RAM

SDK Details

SDK Version:	2.2.0 (released 2017-06-29)
Host OS:	Windows
Toolchain:	MCUXpresso IDE
Middleware:	**CMSIS DSP Library, FatFS, ISSDK, USB stack, FreeRTOS**

Documentation

Base SDK:	MCUXpresso SDK API Reference Manual
Middleware:	ISSDK API Reference Manual SensorFusion API Reference Manual



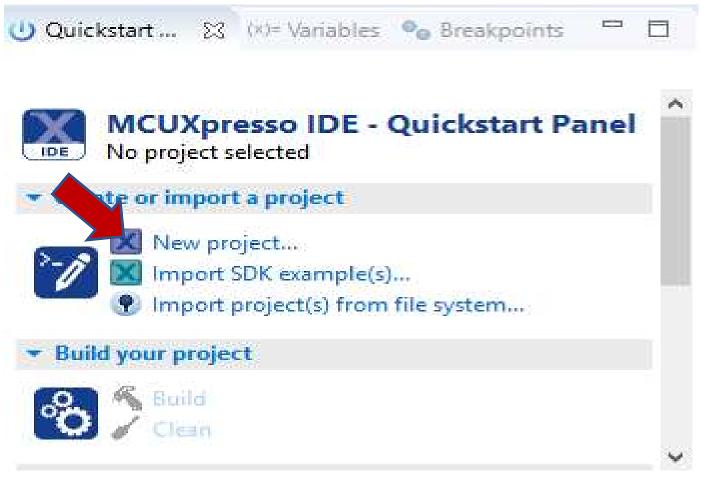
Installed SDKs

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

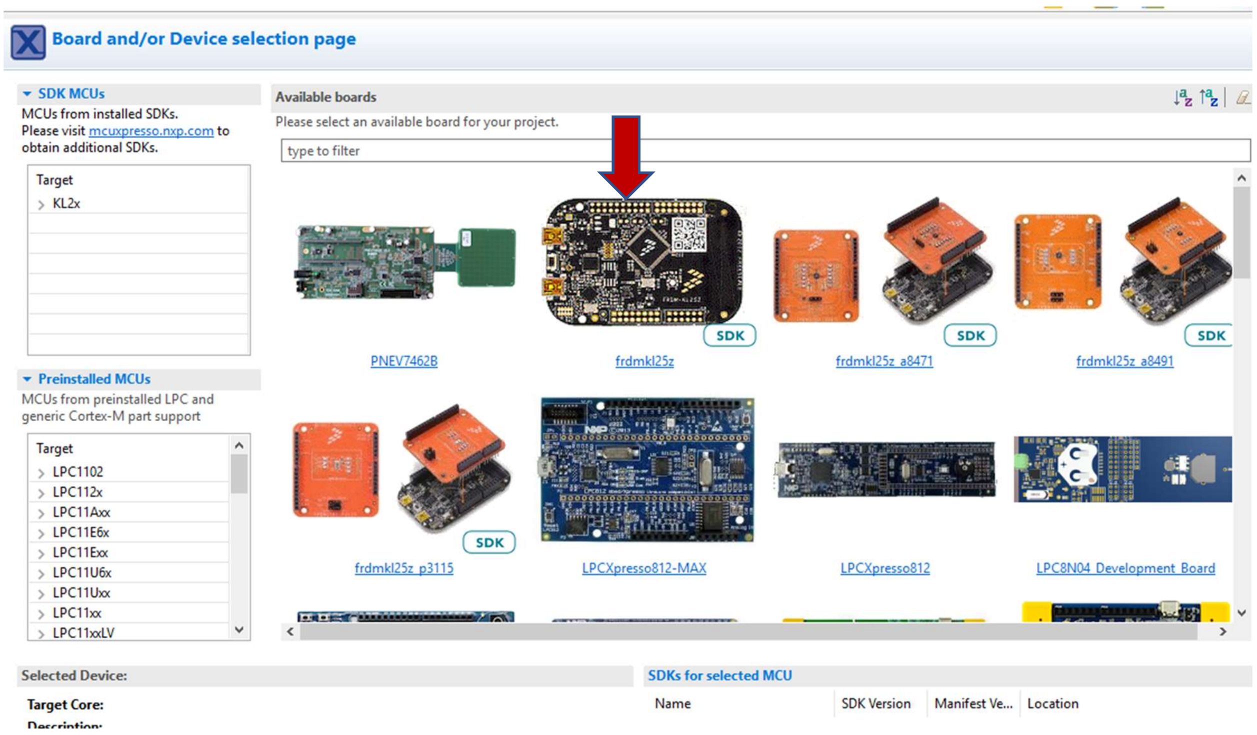
Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_FRDM-KL25Z	2.2.0	3.0.0	<Common>\SDK_2.2.0_FRDM-KL25Z.zip

Click on 'add software component' and select all. Then click on download SDK. Agree to software terms and conditions. The installation will take a few mins and download a zip file. Simply drag and drop the zip file to the installed SDKs box.

To confirm your SDK installation, click on new project option from the quick start panel, or just new icon from the menu bar and select New c/c++ project from under MCUXpresso IDE.



And see that the FRDM KL25Z SDK is added under the list of available boards.


Board and/or Device selection page

SDK MCUs
MCUs from installed SDKs.
Please visit mcuxpresso.nxp.com to obtain additional SDKs.

Target
> KL2x

Preinstalled MCUs
MCUs from preinstalled LPC and generic Cortex-M part support

Target
> LPC1102
> LPC112x
> LPC11Axx
> LPC11E6x
> LPC11Exx
> LPC11U6x
> LPC11Uxx
> LPC11xx
> LPC11xxLV

Available boards
Please select an available board for your project.
type to filter

PNEV7462B
frdmkl25z
frdmkl25z_a8471
frdmkl25z_a8491
frdmkl25z_p3115
LPCXpresso812-MAX
LPCXpresso812
LPC8N04_Development_Board

Selected Device:
Target Core:

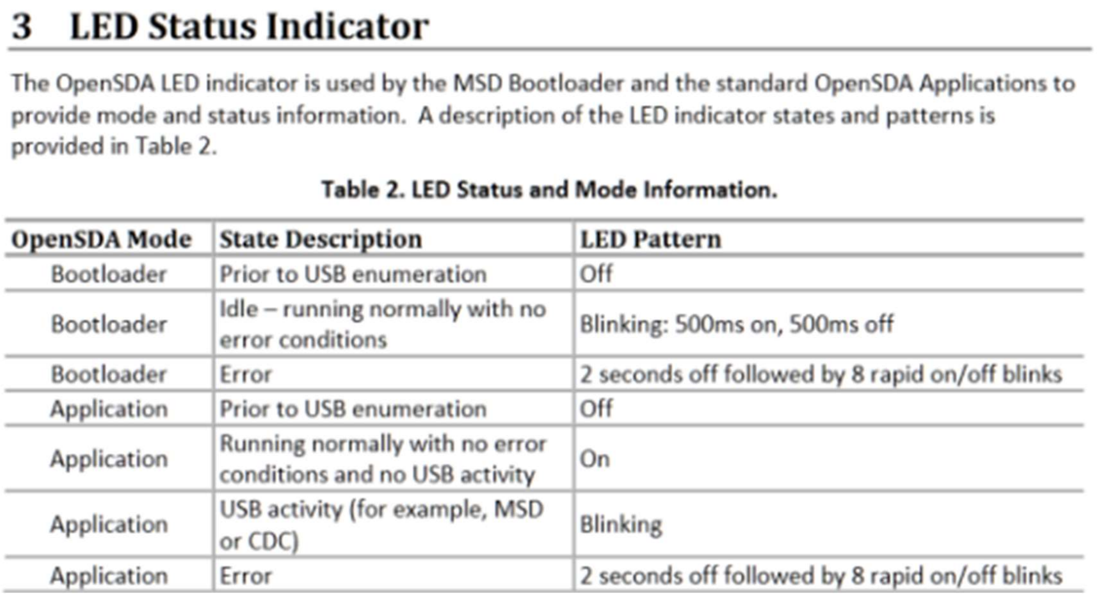
SDKs for selected MCU

Name	SDK Version	Manifest Ve...	Location

At this point you are done with your environment setup, so lets now move to starting a project.

Setting up your FRDM KL25Z

1. Download the FRDM-KL25Z Quick start package (if you don't have it already) from under 'Documents and Software' in the following link-> https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL25Z?&tid=vanFRDM-KL25Z
2. Plug in the USB cable normally with the open sda port and observe if you have a green light on. In case you don't, unplug and plug it in the bootloader mode wile pressing reset button. If you see 8 fast blinks followed by a 2 second gap, your board must be bricked. Follow the unbricking procedure below.
3. Plug in a USB cable (not included) from a USB host to the OpenSDA mini-B USB connector while pressing the tiny 'RST' or reset button between the OpenSDA and KL25Z USB connectors. This puts the board in bootloader mode. Release the button. The FRDMKL25Z will be powered by this USB connection and you will see green led blinking at 1hz, indicating that it's in idle mode. Check the blink pattern shown below.



3 LED Status Indicator

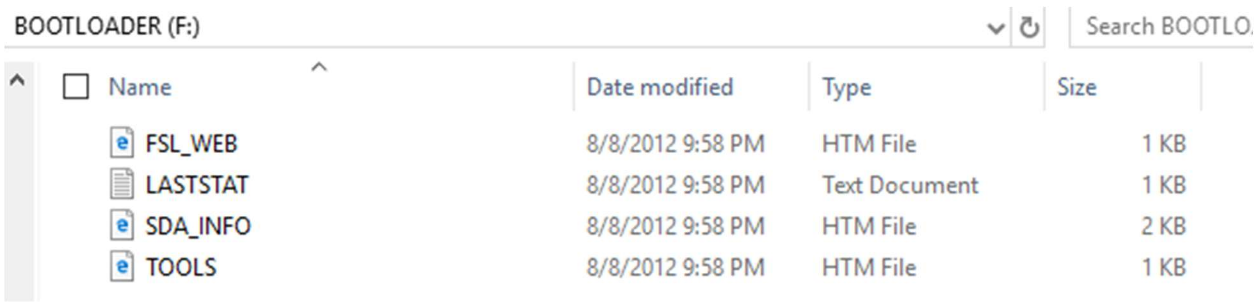
The OpenSDA LED indicator is used by the MSD Bootloader and the standard OpenSDA Applications to provide mode and status information. A description of the LED indicator states and patterns is provided in Table 2.

Table 2. LED Status and Mode Information.

OpenSDA Mode	State Description	LED Pattern
Bootloader	Prior to USB enumeration	Off
Bootloader	Idle – running normally with no error conditions	Blinking: 500ms on, 500ms off
Bootloader	Error	2 seconds off followed by 8 rapid on/off blinks
Application	Prior to USB enumeration	Off
Application	Running normally with no error conditions and no USB activity	On
Application	USB activity (for example, MSD or CDC)	Blinking
Application	Error	2 seconds off followed by 8 rapid on/off blinks

In case you see 8 blinks with 2 sec gap, detach, plug it in normally and see if you can see a steady green light. If not, you may have to follow the unbricking procedure.

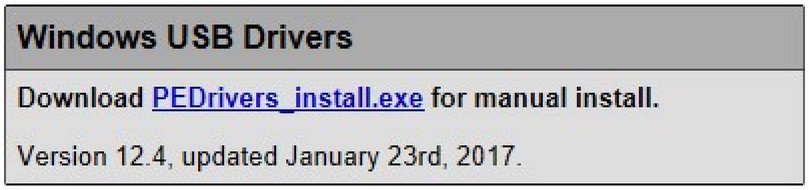
4. On connecting with the computer, A removable drive should now be visible in the host file system with a volume label of BOOTLOADER.


BOOTLOADER (F:)

Name	Date modified	Type	Size
FSL_WEB	8/8/2012 9:58 PM	HTM File	1 KB
LASTSTAT	8/8/2012 9:58 PM	Text Document	1 KB
SDA_INFO	8/8/2012 9:58 PM	HTM File	2 KB
TOOLS	8/8/2012 9:58 PM	HTM File	1 KB

OpenSDA is an open-standard serial and debug adapter. It bridges serial and debug communications between a USB host and an embedded target processor. OpenSDA software includes a flash-resident USB mass-storage device (MSD) bootloader and a collection of OpenSDA Applications. https://www.nxp.com/docs/en/user-guide/OPENSDAUG.pdf

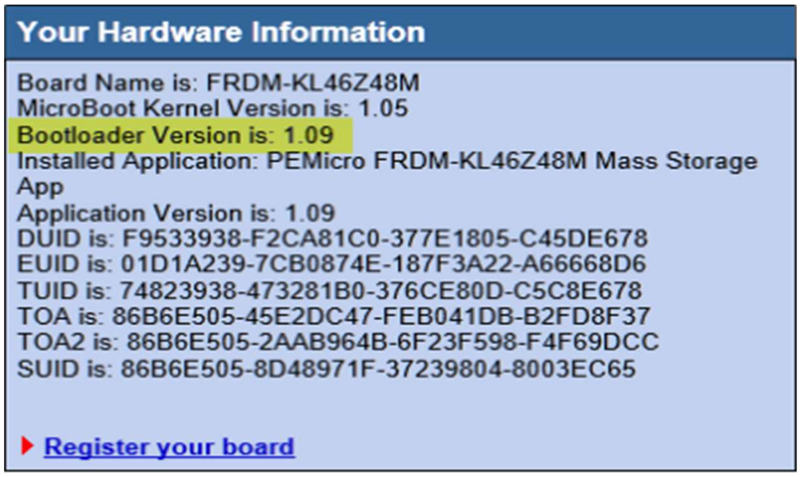
5. Download the Windows USB drivers from http://www.pemicro.com/opensda/index.cfm
   In case SDA_INFO.HTML doesn't automatically opens the pemicro website. Run the exe file. This will help the computer recognize the FRDM device.


Windows USB Drivers

Download PEDrivers_install.exe for manual install.

Version 12.4, updated January 23rd, 2017.

6. While in OpenSDA Bootloader mode, double-click SDA_INFO.HTML in the BOOTLOADER drive. A web browser will open the OpenSDA homepage containing the name and version of the installed Application. This information can also be read as text directly from SDA_INFO.HTML.

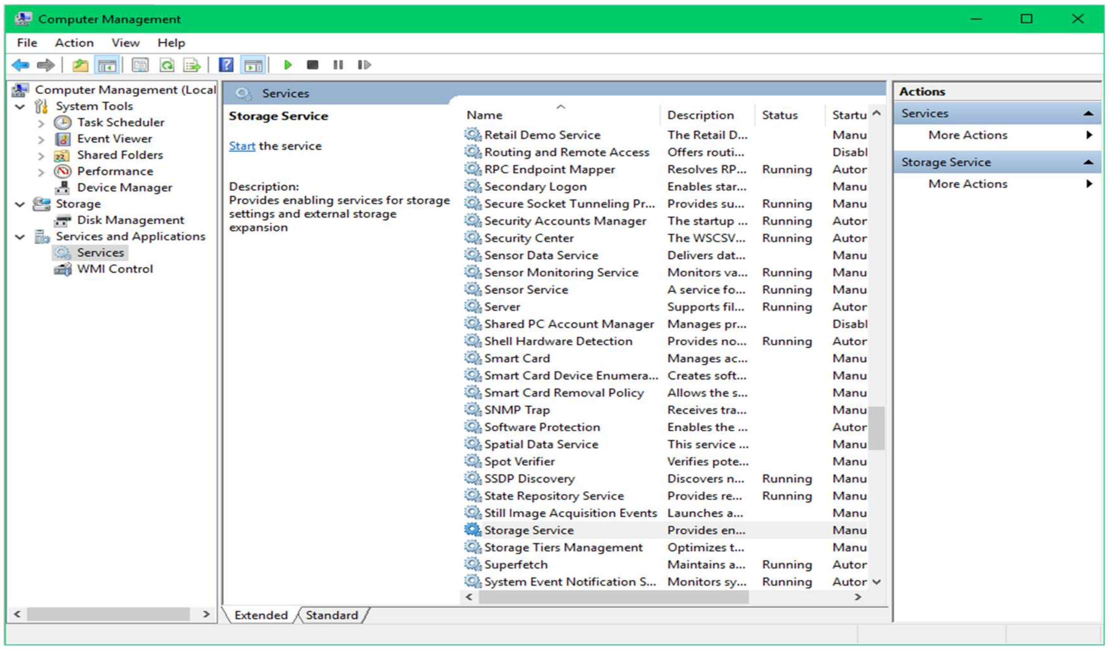

Your Hardware Information

Board Name is: FRDM-KL46Z48M
MicroBoot Kernel Version is: 1.05
Bootloader Version is: 1.09
Installed Application: PEMicro FRDM-KL46Z48M Mass Storage App
Application Version is: 1.09
DUID is: F9533938-F2CA81C0-377E1805-C45DE678
EUID is: 01D1A239-7CB0874E-187F3A22-A66668D6
TUID is: 74823938-473281B0-376CE80D-C5C8E678
TOA is: 86B6E505-45E2DC47-FEB041DB-B2FD8F37
TOA2 is: 86B6E505-2AAB964B-6F23F598-F4F69DCC
SUID is: 86B6E505-8D48971F-37239804-8003EC65

Register your board

Notice the bootloader version. If your machine is windows 7 or less, you can skip it this step. The firmware was developed long time back and is not compatible with Windows 8 or above. Hence we need to update the bootloader of the board. From version 1.09 to 1.11.

a. Unplug the board from host PC.
b. stop/disable several Windows 10 services which interfere with the bootloader.
   I. Go to Computer Management -> Services and Application > Services -> Scroll down to 'Storage Service' and click 'stop the service'.



Services

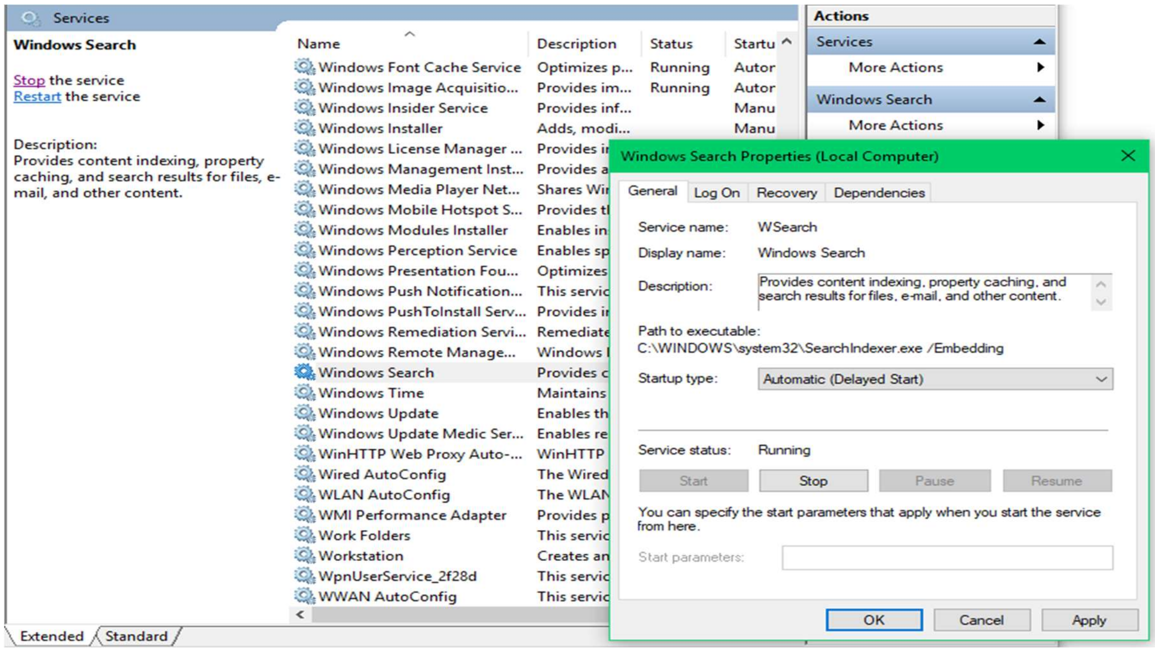
Storage Service

Start the service

Description:
Provides enabling services for storage settings and external storage expansion

Name	Description	Status	Startu
Retail Demo Service	The Retail D...		Manu
Routing and Remote Access	Offers routi...		Disabl
RPC Endpoint Mapper	Resolves RP...	Running	Autor
Secondary Logon	Enables star...		Manu
Secure Socket Tunneling Pr...	Provides su...	Running	Manu
Security Accounts Manager	The startup ...	Running	Autor
Security Center	The WSCSV...	Running	Autor
Sensor Data Service	Delivers dat...		Manu
Sensor Monitoring Service	Monitors va...	Running	Manu
Sensor Service	A service fo...	Running	Manu
Server	Supports fil...	Running	Autor
Shared PC Account Manager	Manages pr...		Disabl
Shell Hardware Detection	Provides no...	Running	Autor
Smart Card	Manages ac...		Manu
Smart Card Device Enumera...	Creates soft...		Manu
Smart Card Removal Policy	Allows the s...		Manu
SNMP Trap	Receives tra...		Manu
Software Protection	Enables the ...		Autor
Spatial Data Service	This service ...		Manu
Spot Verifier	Verifies pote...		Manu
SSDP Discovery	Discovers n...	Running	Manu
State Repository Service	Provides re...	Running	Manu
Still Image Acquisition Events	Launches a...		Manu
Storage Service	Provides en...		Manu
Storage Tiers Management	Optimizes t...		Manu
Superfetch	Maintains a...	Running	Autor
System Event Notification S...	Monitors sy...	Running	Autor

II. Scroll down further to ‘Windows Search’, right click -> properties -> under startup choose disabled -> okay and then ‘click stop the service’



Windows Search

Stop the service
Restart the service

Description:
Provides content indexing, property caching, and search results for files, e-mail, and other content.

Name	Description	Status	Startu
Windows Font Cache Service	Optimizes p...	Running	Autor
Windows Image Acquisitio...	Provides im...	Running	Autor
Windows Insider Service	Provides inf...		Manu
Windows Installer	Adds, modi...		Manu
Windows License Manager ...	Provides i		
Windows Management Inst...	Provides a		
Windows Media Player Net...	Shares Wir		
Windows Mobile Hotspot S...	Provides t		
Windows Modules Installer	Enables in		
Windows Perception Service	Enables sp		
Windows Presentation Fou...	Optimizes		
Windows Push Notification...	This servic		
Windows PushToInstall Serv...	Provides i		
Windows Remediation Servi...	Remediate		
Windows Remote Manage...	Windows		
Windows Search	Provides c		
Windows Time	Maintains		
Windows Update	Enables th		
Windows Update Medic Ser...	Enables re		
WinHTTP Web Proxy Auto-...	WinHTTP		
Wired AutoConfig	The Wired		
WLAN AutoConfig	The WLAN		
WMI Performance Adapter	Provides p		
Work Folders	This servic		
Workstation	Creates an		
WpnUserService_2f28d	This servic		
WWAN AutoConfig	This servic		

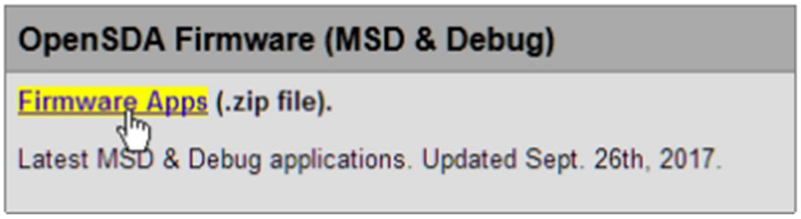
Windows Search Properties (Local Computer)

Service name: WSearch
Display name: Windows Search
Description: Provides content indexing, property caching, and search results for files, e-mail, and other content.
Path to executable:
C:\WINDOWS\system32\SearchIndexer.exe /Embedding
Startup type: Automatic (Delayed Start)
Service status: Running

You can specify the start parameters that apply when you start the service from here.

Start parameters:

III. Go to http://www.pemicro.com/opensda/ download the following

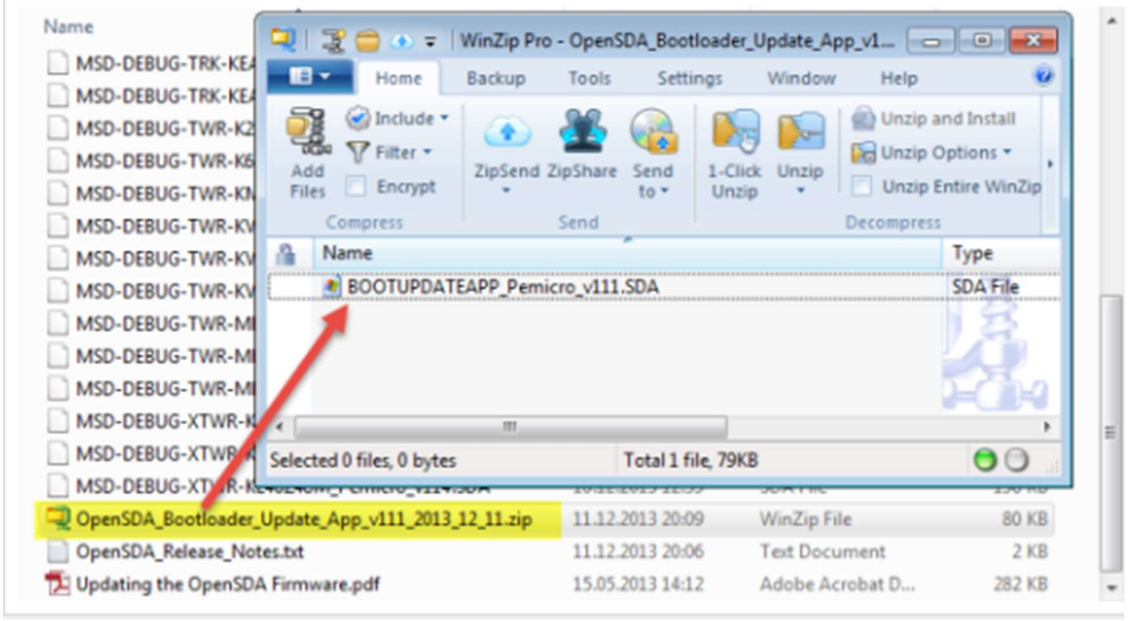


OpenSDA Firmware (MSD & Debug)

Firmware Apps (.zip file).

Latest MSD & Debug applications. Updated Sept. 26th, 2017.

Extract all files. Extract the 'OpenSDA_Bootloader_Update*.zip' inside that zip file as well.



Name	Type
BOOTUPDATEAPP_Pemicro_v111.SDA	SDA File

Selected 0 files, 0 bytes   Total 1 file, 79KB

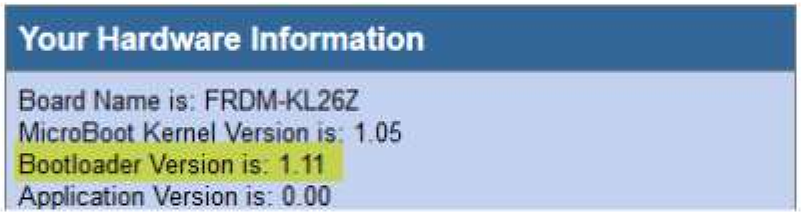
Name			
MSD-DEBUG-TRK-KEA			
MSD-DEBUG-TRK-KEA			
MSD-DEBUG-TWR-K2			
MSD-DEBUG-TWR-K6			
MSD-DEBUG-TWR-KM			
MSD-DEBUG-TWR-KV			
MSD-DEBUG-TWR-KV			
MSD-DEBUG-TWR-KV			
MSD-DEBUG-TWR-MI			
MSD-DEBUG-TWR-MI			
MSD-DEBUG-TWR-MI			
MSD-DEBUG-XTWR-K			
MSD-DEBUG-XTWR			
MSD-DEBUG-XTWR-K			
OpenSDA_Bootloader_Update_App_v111_2013_12_11.zip	11.12.2013 20:09	WinZip File	80 KB
OpenSDA_Release_Notes.txt	11.12.2013 20:06	Text Document	2 KB
Updating the OpenSDA Firmware.pdf	15.05.2013 14:12	Adobe Acrobat D...	282 KB

You will need the BOOTUPDATEAPP_Pemicro_v111.SDA.

IV. Press first the Reset button (and keep it pressed) while power the board with the Open SDA USB connector. And the board shows up as Bootloader drive as mentioned above.

V. Copy the update file mentioned above to the bootloader drive and then unpower the board.

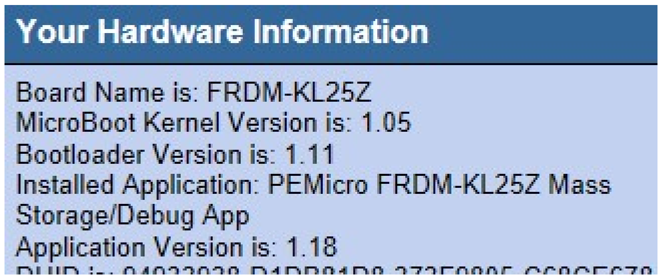
VI. Power the board again without pressing the reset button, this should open up the bootloader driver. Click on SDA_INFO.HTML file and you can see that the bootloader has been updated



Your Hardware Information

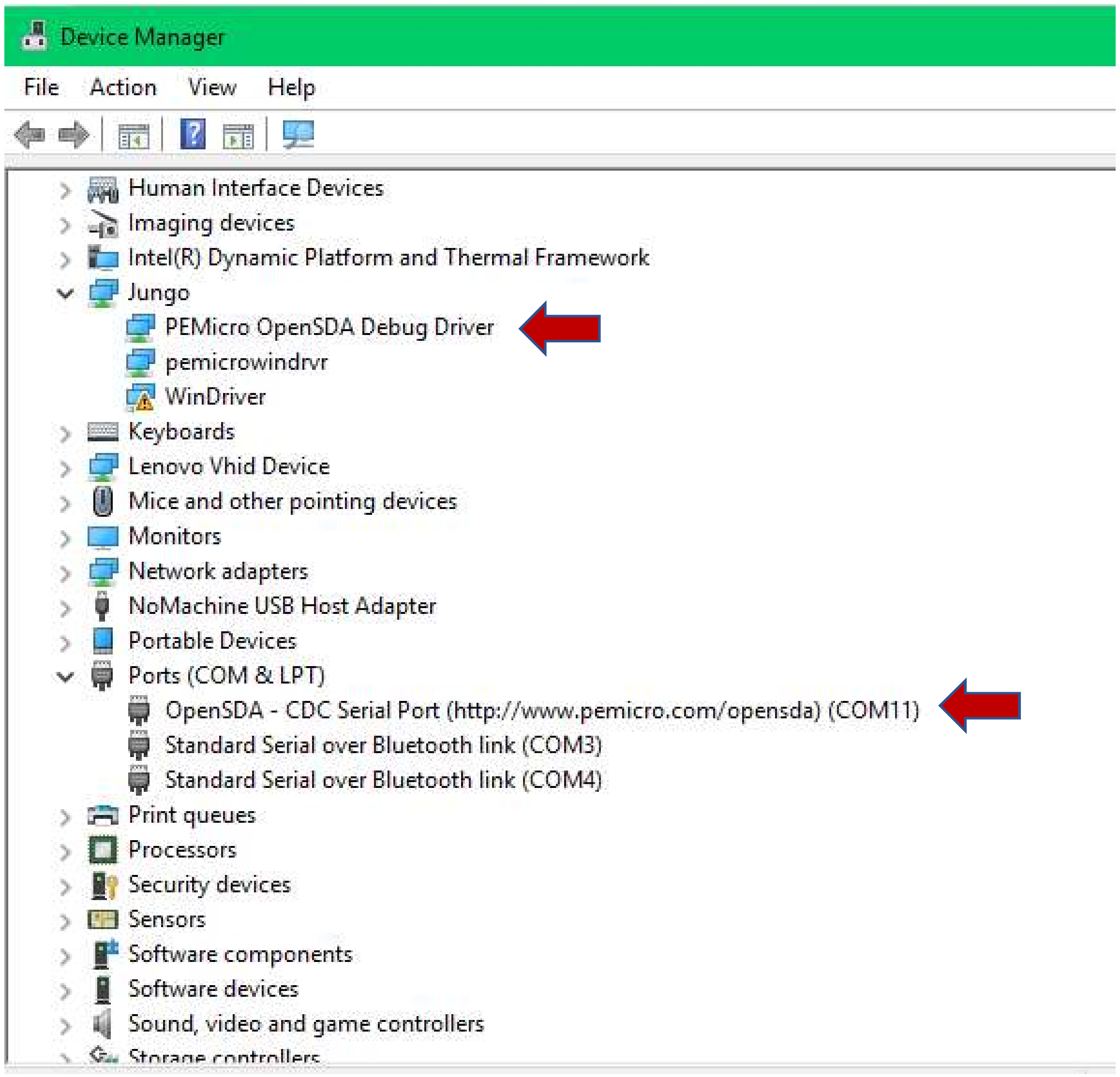
Board Name is: FRDM-KL26Z
MicroBoot Kernel Version is: 1.05
Bootloader Version is: 1.11
Application Version is: 0.00

7. Now go to your earlier downloaded OpenSDA firmware, MSD & DEBUG (look above) Select the 'MSD-DEBUG-FRDM-KL25Z_Pemicro_v118.SDA' file and copy it to your bootloader drive. After that, un power your board.
8. Power the board again, in the normal mode (without pressing the reset button). The board will show up with it's name as the device. And the green LED will always be on. Lastly click on SDA_INFO.HTML again and the final hardware information should look something like this:


Your Hardware Information

Board Name is: FRDM-KL25Z
MicroBoot Kernel Version is: 1.05
Bootloader Version is: 1.11
Installed Application: PEMicro FRDM-KL25Z Mass Storage/Debug App
Application Version is: 1.18

Go to device manager and see if you have them:



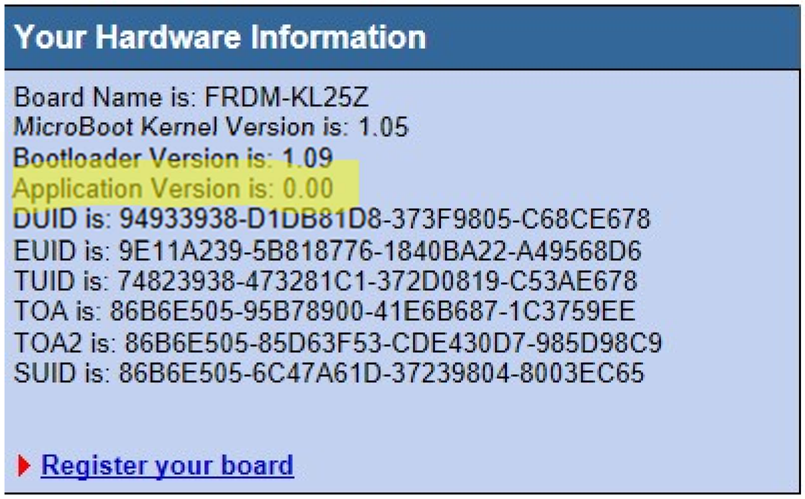
- Human Interface Devices
- Imaging devices
- Intel(R) Dynamic Platform and Thermal Framework
- Jungo
  - PEMicro OpenSDA Debug Driver
  - pemicrowindrvr
  - WinDriver
- Keyboards
- Lenovo Vhid Device
- Mice and other pointing devices
- Monitors
- Network adapters
- NoMachine USB Host Adapter
- Portable Devices
- Ports (COM & LPT)
  - OpenSDA - CDC Serial Port (http://www.pemicro.com/opensda) (COM11)
  - Standard Serial over Bluetooth link (COM3)
  - Standard Serial over Bluetooth link (COM4)
- Print queues
- Processors
- Security devices
- Sensors
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers

At this point you are all set and ready to start with your project!!

How to Unbrick your board?

FRDM KL25Z will work without a problem on windows7, but for windows 8 and above, it doesn't work. Its firmware is outdated. If on normal plug in your green LED is off and in booth loader more, you LED pattern shows the error code, that means the OpenSDA debug application has been erased and the board does not boot into debug mode anymore. It's extremely easy to brick your board if you are using windows 10. Bootloader gets stuck by the information send by the windows 10 machine and erases the debug application.

1. Plug it in, in the bootloader mode holding the reset button. You will be able to see the boot loader drive. Click on 'SDA_INFO' HTML file. And you should see something like this:



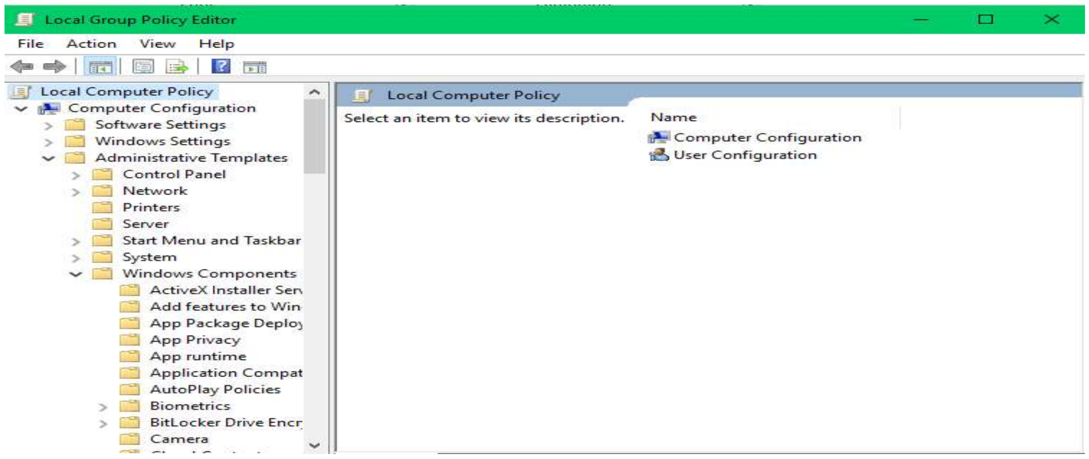
Your Hardware Information

Board Name is: FRDM-KL25Z
MicroBoot Kernel Version is: 1.05
Bootloader Version is: 1.09
Application Version is: 0.00
DUID is: 94933938-D1DB81D8-373F9805-C68CE678
EUID is: 9E11A239-5B818776-1840BA22-A49568D6
TUID is: 74823938-473281C1-372D0819-C53AE678
TOA is: 86B6E505-95B78900-41E6B687-1C3759EE
TOA2 is: 86B6E505-85D63F53-CDE430D7-985D98C9
SUID is: 86B6E505-6C47A61D-37239804-8003EC65

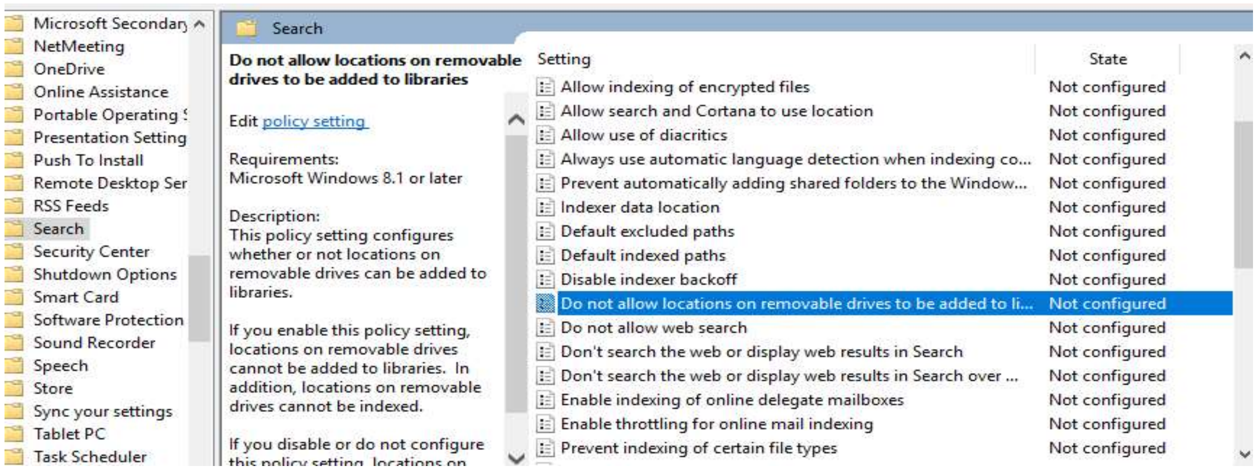
Register your board

   Observe that the application version is 0.00 that means there is no debug application any more on the board and hence you can't debug anymore. This should confirm that your board is definitely bricked.

2. We need to prevent windows 10 to write anything to the virtual msd device. Press Windows+R to open 'Run' and type 'gpedit.msc' to launch the Microsoft configuration console. In case you do not have gpedit.msc enabled. Follow this tutorial -> https://www.youtube.com/watch?v=J28r5u5Wqy4
3. This opens a Local Group Policy Editor. Go to **Computer Configuration -> Administrative Templates -> Windows Components** , and scroll all the way down to 'search' and click on it.



Click on ‘Do not allow locations on removable devices to be added to libraries’ option and double click on it.



Microsoft Secondary
NetMeeting
OneDrive
Online Assistance
Portable Operating S
Presentation Setting
Push To Install
Remote Desktop Ser
RSS Feeds
Search
Security Center
Shutdown Options
Smart Card
Software Protection
Sound Recorder
Speech
Store
Sync your settings
Tablet PC
Task Scheduler

Search

Do not allow locations on removable drives to be added to libraries

Edit policy setting

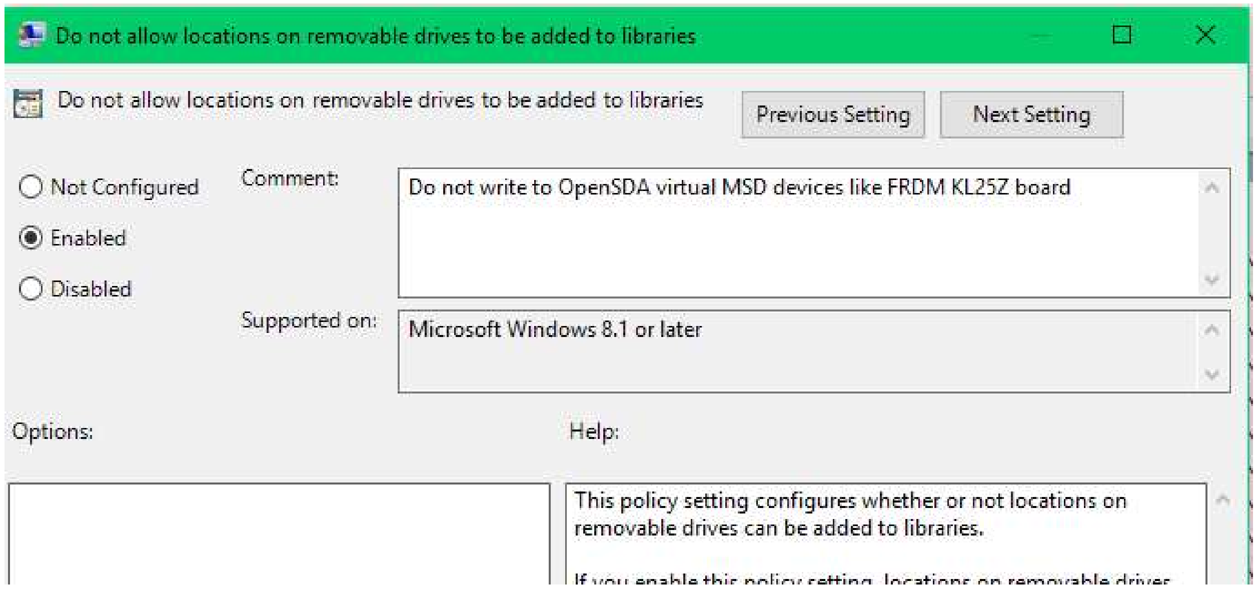
Requirements:
Microsoft Windows 8.1 or later

Description:
This policy setting configures whether or not locations on removable drives can be added to libraries.

If you enable this policy setting, locations on removable drives cannot be added to libraries. In addition, locations on removable drives cannot be indexed.

If you disable or do not configure this policy setting, locations on

Setting	State
Allow indexing of encrypted files	Not configured
Allow search and Cortana to use location	Not configured
Allow use of diacritics	Not configured
Always use automatic language detection when indexing co...	Not configured
Prevent automatically adding shared folders to the Window...	Not configured
Indexer data location	Not configured
Default excluded paths	Not configured
Default indexed paths	Not configured
Disable indexer backoff	Not configured
Do not allow locations on removable drives to be added to li...	Not configured
Do not allow web search	Not configured
Don't search the web or display web results in Search	Not configured
Don't search the web or display web results in Search over ...	Not configured
Enable indexing of online delegate mailboxes	Not configured
Enable throttling for online mail indexing	Not configured
Prevent indexing of certain file types	Not configured



Do not allow locations on removable drives to be added to libraries

Do not allow locations on removable drives to be added to libraries

Not Configured [✓ Enabled] Disabled

Comment: Do not write to OpenSDA virtual MSD devices like FRDM KL25Z board

Supported on: Microsoft Windows 8.1 or later

Options:

Help:

This policy setting configures whether or not locations on removable drives can be added to libraries.

If you enable this policy setting, locations on removable drives

Write an informative comment in the comment section and change from ‘Not Configure’ to ‘Enabled’. Press Apply and then Ok. Now under the search settings, the options should show Enabled.

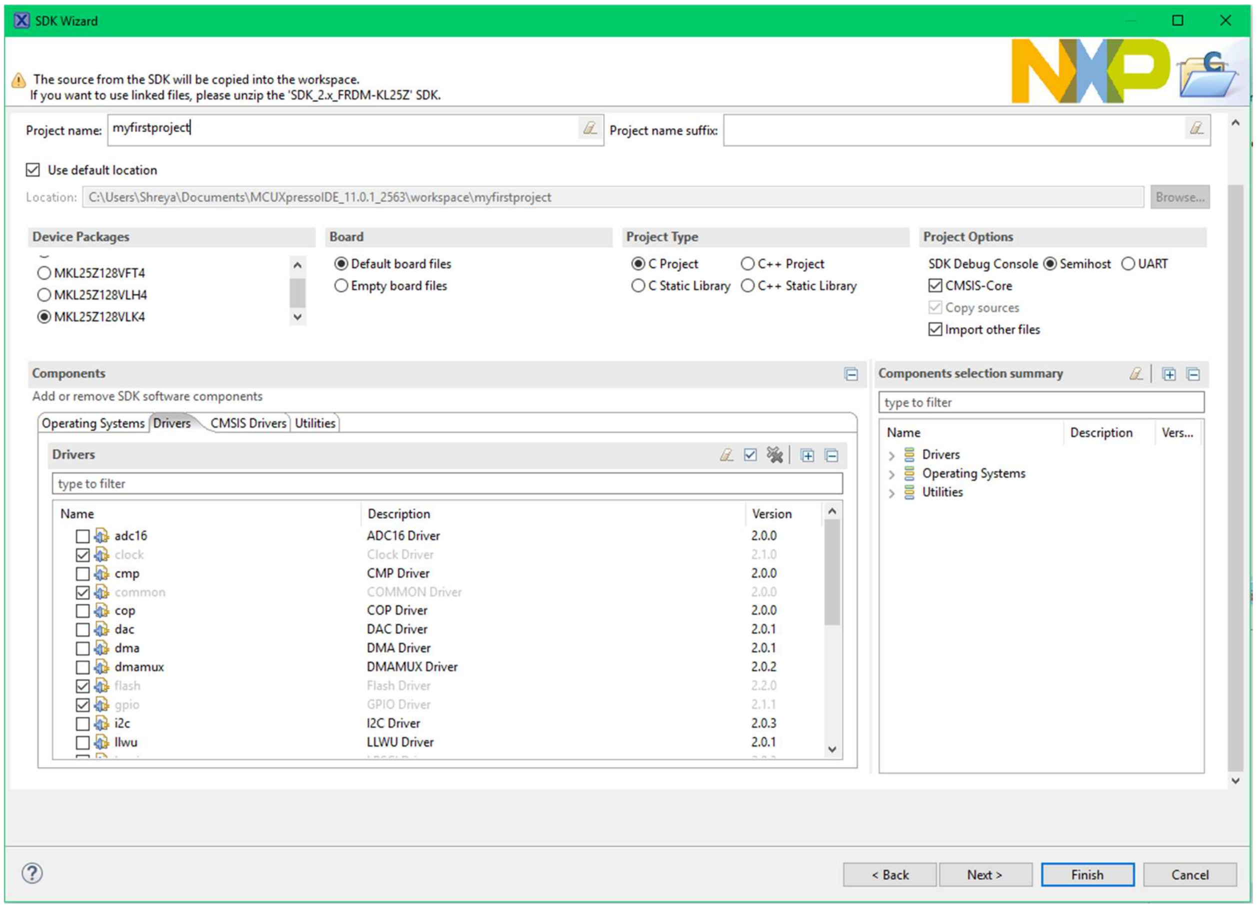
4. Restart your PC and plug your device in bootloader mode. The green LED now blinks at 1Hz. Phew!! Still a little more to go!

   Follow the normal steps as mentioned under the set up of KL25Z.

Getting Started with your First Project

Important link -> https://www.nxp.com/docs/en/user-guide/MCUXSDKGSUG.pdf

Go to new project button from the Quickstart menu or new from the menu bar. Click on the KL25Z SDK from the board and device selection page then click next.



SDK Wizard

The source from the SDK will be copied into the workspace.
If you want to use linked files, please unzip the 'SDK_2.x_FRDM-KL25Z' SDK.

Project name: myfirstproject
Project name suffix:
Use default location [✓]
Location: C:\Users\Shreya\Documents\MCUXpressoIDE_11.0.1_2563\workspace\myfirstproject

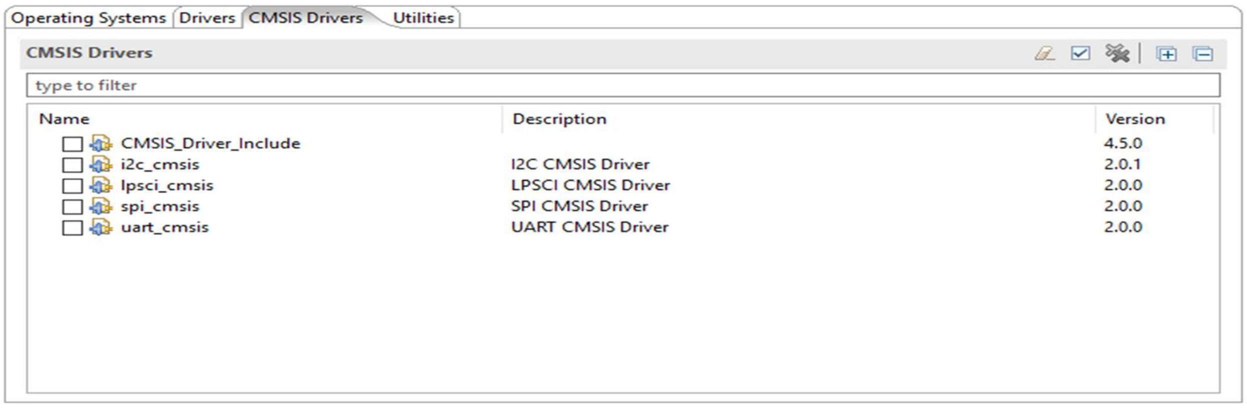
Device Packages: MKL25Z128VFT4, MKL25Z128VLH4, [✓ MKL25Z128VLK4]
Board: [✓ Default board files], Empty board files
Project Type: [✓ C Project], C++ Project, C Static Library, C++ Static Library
Project Options: SDK Debug Console [✓ Semihost] UART; CMSIS-Core [✓]; Copy sources [✓]; Import other files [✓]

Components — Add or remove SDK software components

Name	Description	Version
☐ adc16	ADC16 Driver	2.0.0
☑ clock	Clock Driver	2.1.0
☐ cmp	CMP Driver	2.0.0
☑ common	COMMON Driver	2.0.0
☐ cop	COP Driver	2.0.0
☐ dac	DAC Driver	2.0.1
☐ dma	DMA Driver	2.0.1
☐ dmamux	DMAMUX Driver	2.0.2
☑ flash	Flash Driver	2.2.0
☑ gpio	GPIO Driver	2.1.1
☐ i2c	I2C Driver	2.0.3
☐ llwu	LLWU Driver	2.0.1

Components selection summary: Drivers, Operating Systems, Utilities

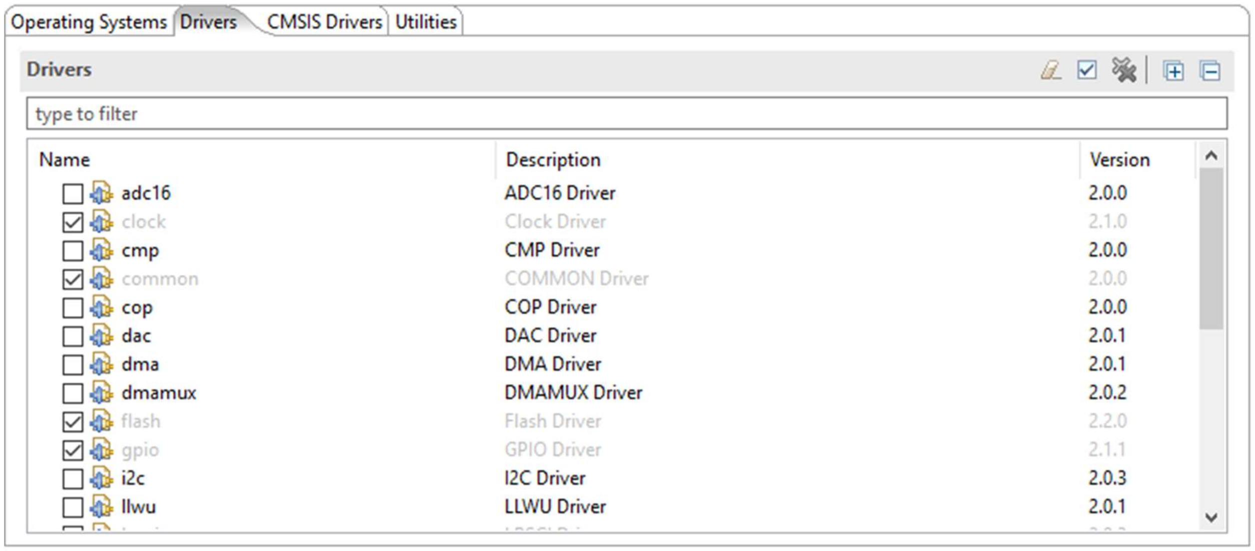
The SDK wizard as shown above allows us to choose operating systems, add remove drivers. The CMSIS (Cortex Microcontroller Software Interface Standard.) is an ARM standard. It is a portable software across all Cortex implementation and is usually written in c/c++.



CMSIS Drivers

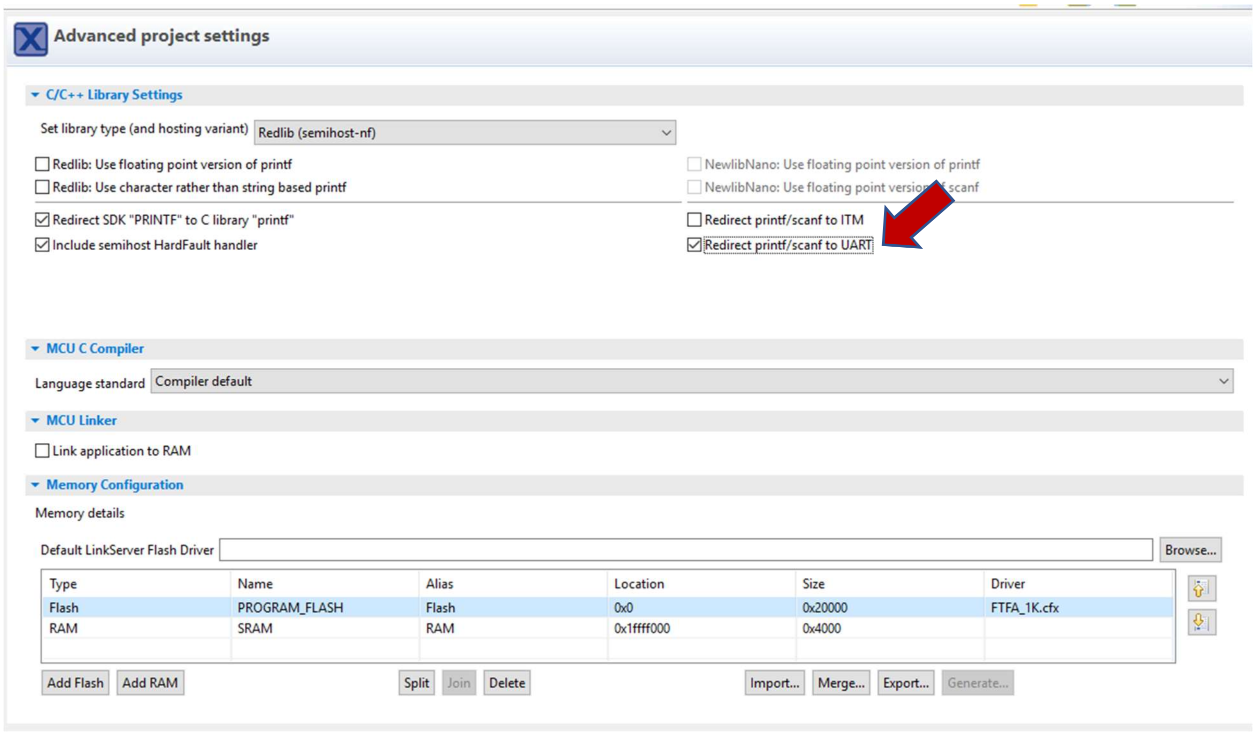
Name	Description	Version
☐ CMSIS_Driver_Include		4.5.0
☐ i2c_cmsis	I2C CMSIS Driver	2.0.1
☐ lpsci_cmsis	LPSCI CMSIS Driver	2.0.0
☐ spi_cmsis	SPI CMSIS Driver	2.0.0
☐ uart_cmsis	UART CMSIS Driver	2.0.0

For the purpose of this project we do not need to include any of the CMSIS drivers



Name	Description	Version
☐ adc16	ADC16 Driver	2.0.0
☑ clock	Clock Driver	2.1.0
☐ cmp	CMP Driver	2.0.0
☑ common	COMMON Driver	2.0.0
☐ cop	COP Driver	2.0.0
☐ dac	DAC Driver	2.0.1
☐ dma	DMA Driver	2.0.1
☐ dmamux	DMAMUX Driver	2.0.2
☑ flash	Flash Driver	2.2.0
☑ gpio	GPIO Driver	2.1.1
☐ i2c	I2C Driver	2.0.3
☐ llwu	LLWU Driver	2.0.1

These are the peripheral drivers and are more vendor specific. Make sure you have gpio driver selected. No need to change the rest. Under operating systems, select 'baremetal' if it's not already selected. Bare metal programming refers to writing firmware which directly runs on the Target (hardware) without any underlying abstraction such as operating systems. This is what we are going to do in this class. Let the utilities be as it is. Finally give a project name, select the device package – MKL25Z128VLK4 , select 'c project' and click next.



Set library type (and hosting variant): Redlib (semihost-nf)

☐ Redlib: Use floating point version of printf
☐ Redlib: Use character rather than string based printf
☐ NewlibNano: Use floating point version of printf
☐ NewlibNano: Use floating point version of scanf
☑ Redirect SDK "PRINTF" to C library "printf"
☑ Include semihost HardFault handler
☐ Redirect printf/scanf to ITM
☑ Redirect printf/scanf to UART

Language standard: Compiler default

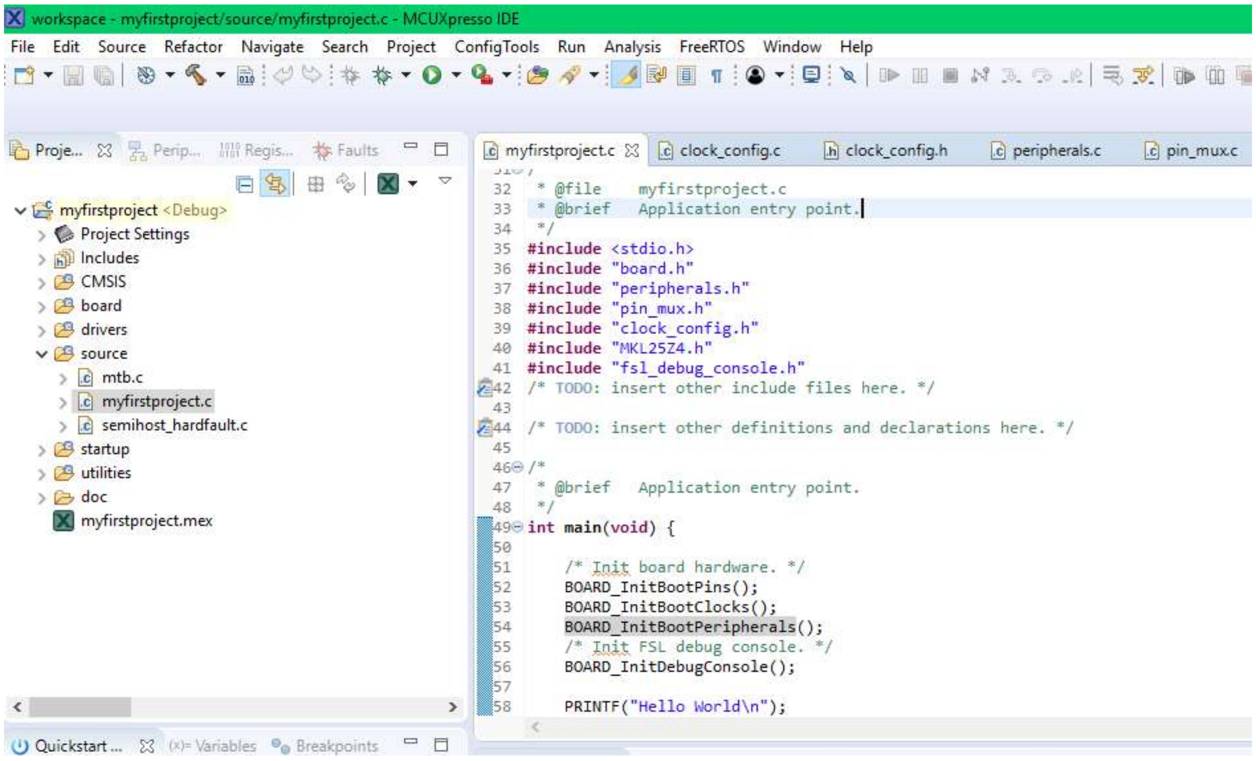
☐ Link application to RAM

Default LinkServer Flash Driver:

Type	Name	Alias	Location	Size	Driver
Flash	PROGRAM_FLASH	Flash	0x0	0x20000	FTFA_1K.cfx
RAM	SRAM	RAM	0x1ffff000	0x4000	

In the advanced project settings, select 'Redirect printf/scanf to UART' and click finish.

Our project gets generated and we see the following perspective view



```
 * @file    myfirstproject.c
 * @brief   Application entry point.
 */
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"
#include "fsl_debug_console.h"
/* TODO: insert other include files here. */

/* TODO: insert other definitions and declarations here. */

/*
 * @brief   Application entry point.
 */
int main(void) {

    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
    /* Init FSL debug console. */
    BOARD_InitDebugConsole();

    PRINTF("Hello World\n");
```

We would go into more details about the project structure and each files later. For now, we have an auto generated entry point application where we can add out own code. Our goal is to blink an led.

So first let's decide which port and pin number corresponds to out Led of interest, For that we need to check the pinout first. From the pinout mentioned in the introduction we see that we have 3 available LEDs. Lets choose LED3 PTD1 which is Port D pin 1. We need to do the following:

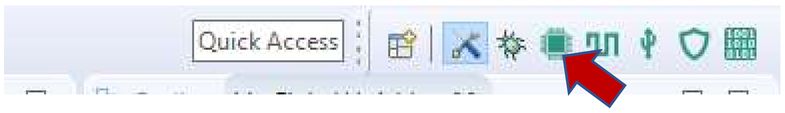
a. **Clocks Enable** for Port D. For any peripheral, port etc the clocks must be enabled or it won't work or throw an exception

b. **Set Pin Mux** the pins. Every pin has chip specific alternate functions. In our scenario we need to set the pin as normal gpio

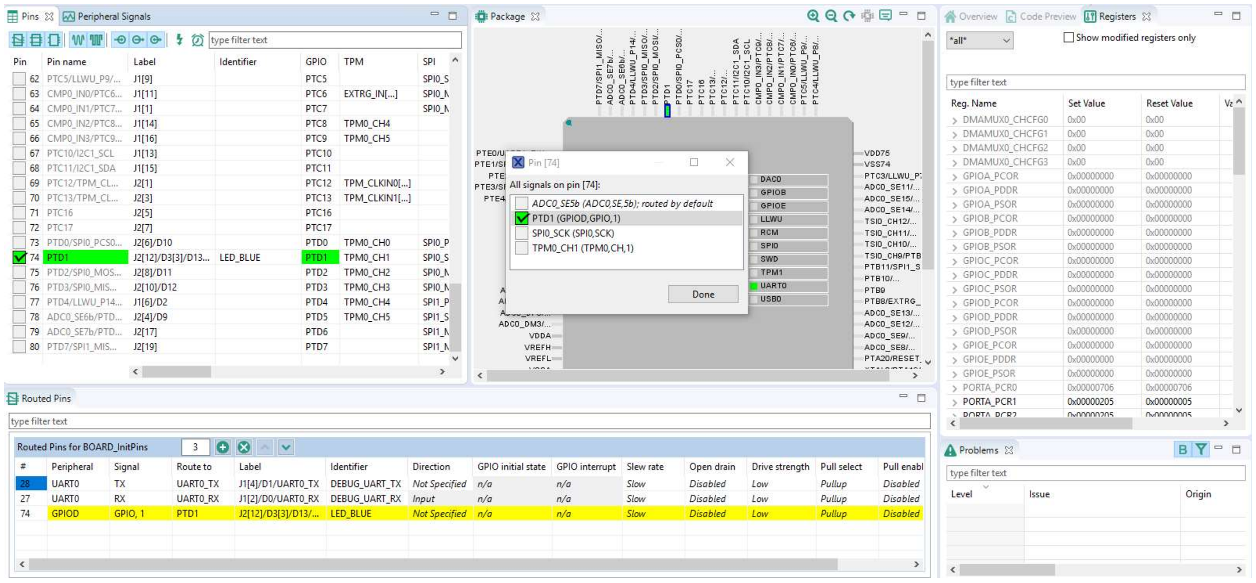
c. **Configure** the pins. This needs to be set as either input or output.

d. **Set pin val :** 1 or 0 for light and no light with delay in between to emulate blink

To Enable clock for port D and set Pin mux, MCUXpresso provides us a Pins tool. **Pins** tool is an easy-to-use tool for configuration of device pins. The **Pins** tool software helps create, inspect, change, and modify any element of pin configuration and device muxing.



Click on it and you would see a screen like this:



Pin	Pin name	Label	Identifier	GPIO	TPM	SPI
62	PTC5/LLWU_P9/...	J1[9]		PTC5		SPI0_S
63	CMP0_IN0/PTC6...	J1[11]		PTC6	EXTRG_IN[...]	SPI0_M
64	CMP0_IN1/PTC7...	J1[1]		PTC7		SPI0_M
65	CMP0_IN2/PTC8...	J1[14]		PTC8	TPM0_CH4	
66	CMP0_IN3/PTC9...	J1[16]		PTC9	TPM0_CH5	
67	PTC10/I2C1_SCL	J1[13]		PTC10		
68	PTC11/I2C1_SDA	J1[15]		PTC11		
69	PTC12/TPM_CL...	J2[1]		PTC12	TPM_CLKIN0[...]	
70	PTC13/TPM_CL...	J2[3]		PTC13	TPM_CLKIN1[...]	
71	PTC16	J2[5]		PTC16		
72	PTC17	J2[7]		PTC17		
73	PTD0/SPI0_PCS0...	J2[6]/D10		PTD0	TPM0_CH0	SPI0_P
✓ 74	PTD1	J2[12]/D3[3]/D13...	LED_BLUE	PTD1	TPM0_CH1	SPI0_S
75	PTD2/SPI0_MOS...	J2[8]/D11		PTD2	TPM0_CH2	SPI0_M
76	PTD3/SPI0_MIS...	J2[10]/D12		PTD3	TPM0_CH3	SPI0_M
77	PTD4/LLWU_P14...	J1[6]/D2		PTD4	TPM0_CH4	SPI1_P
78	ADC0_SE6b/PTD...	J2[4]/D9		PTD5	TPM0_CH5	SPI1_S
79	ADC0_SE7b/PTD...	J2[17]		PTD6		SPI1_M
80	PTD7/SPI1_MIS...	J2[19]		PTD7		SPI1_M

Pin [74]

All signals on pin [74]:

- [ ] *ADC0_SE5b (ADC0,SE,5b); routed by default*
- [✓] PTD1 (GPIOD,GPIO,1)
- [ ] SPI0_SCK (SPI0,SCK)
- [ ] TPM0_CH1 (TPM0,CH,1)

Done

Routed Pins for BOARD_InitPins 3

#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive strength	Pull select	Pull enabl
28	UART0	TX	UART0_TX	J1[4]/D1/UART0_TX	DEBUG_UART_TX	*Not Specified*	*n/a*	*n/a*	*Slow*	*Disabled*	*Low*	*Pullup*	*Disabled*
27	UART0	RX	UART0_RX	J1[2]/D0/UART0_RX	DEBUG_UART_RX	*Input*	*n/a*	*n/a*	*Slow*	*Disabled*	*Low*	*Pullup*	*Disabled*
74	GPIOD	GPIO, 1	PTD1	J2[12]/D3[3]/D13/...	LED_BLUE	*Not Specified*	*n/a*	*n/a*	*Slow*	*Disabled*	*Low*	*Pullup*	*Disabled*

Registers: *all* ☐ Show modified registers only

Reg. Name	Set Value	Reset Value
DMAMUX0_CHCFG0	0x00	0x00
DMAMUX0_CHCFG1	0x00	0x00
DMAMUX0_CHCFG2	0x00	0x00
DMAMUX0_CHCFG3	0x00	0x00
GPIOA_PCOR	0x00000000	0x00000000
GPIOA_PDDR	0x00000000	0x00000000
GPIOA_PSOR	0x00000000	0x00000000
GPIOB_PCOR	0x00000000	0x00000000
GPIOB_PDDR	0x00000000	0x00000000
GPIOB_PSOR	0x00000000	0x00000000
GPIOC_PCOR	0x00000000	0x00000000
GPIOC_PDDR	0x00000000	0x00000000
GPIOC_PSOR	0x00000000	0x00000000
GPIOD_PCOR	0x00000000	0x00000000
GPIOD_PDDR	0x00000000	0x00000000
GPIOD_PSOR	0x00000000	0x00000000
GPIOE_PCOR	0x00000000	0x00000000
GPIOE_PDDR	0x00000000	0x00000000
GPIOE_PSOR	0x00000000	0x00000000
PORTA_PCR0	0x00000706	0x00000706
PORTA_PCR1	0x00000205	0x00000005

Problems

Level	Issue	Origin

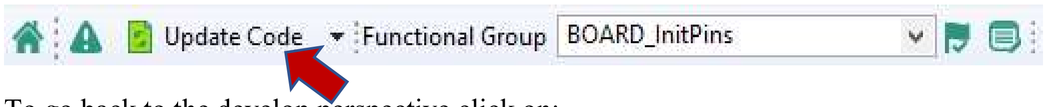
In the **Pins** view on the left, scroll down to LED_BLUE, PTD1 and click on it, then click on the pin functionality and select GPIO, done. When you hover over PTD1, you can see the details as follows

Pin No.: 74, ADC0_SE5b/PTD1/SPI0_SCK/TPM0_CH1
ADC0 analog channel 5b;General purpose IO, Port D, bit 1;SPI0 Serial Clock Output;TPM0 channel 1

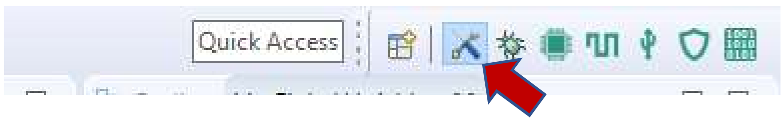
Routed by default and dedicated signals:
ADC0_SE5b (ADC0,SE,5b) - Single-ended channel 5b; Analog; Input; **routed by default**

Pin is routed to signals:
PTD1 (GPIOD,GPIO,1) - General purpose IO pin 1; Digital; Input, Output; features: interrupt
Pin is routed in functions:
BOARD_InitPins

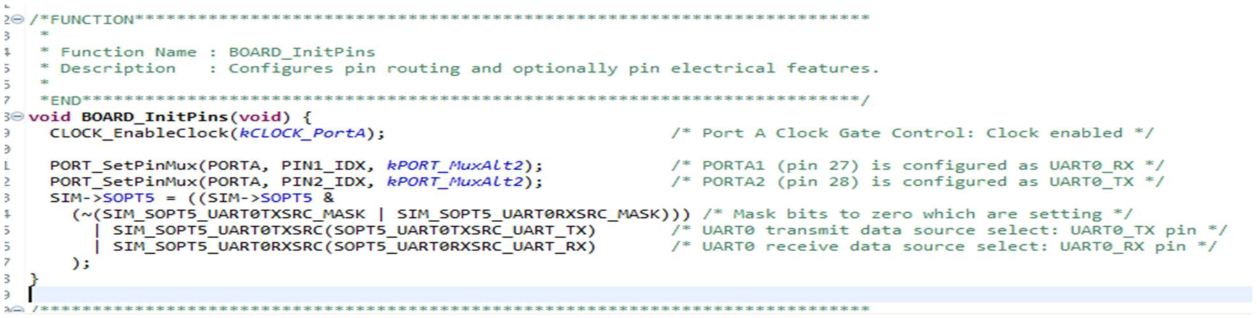
This gives us pin information. We know the default routing was some ADC0.. but now its GPIO after our selection. This selection is reflected in the function BOARD_InitPins. Click 'update code' from the menu and click OK.



To go back to the develop perspective click on:

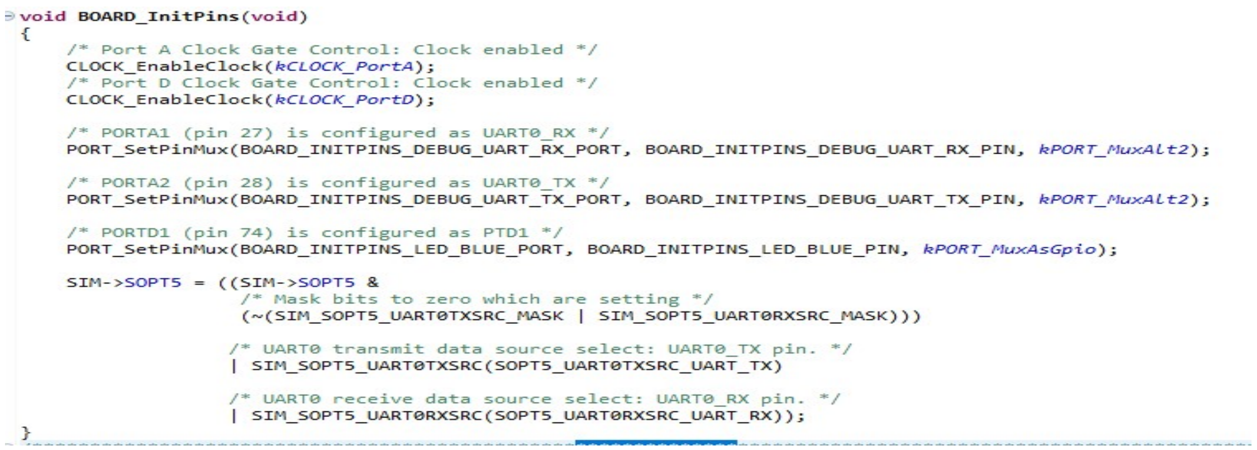


Before:



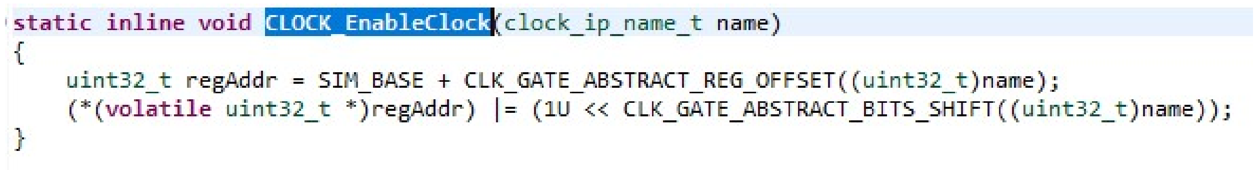
```
/*FUNCTION**********************************************************************
 *
 * Function Name : BOARD_InitPins
 * Description   : Configures pin routing and optionally pin electrical features.
 *
 *END**************************************************************************/
void BOARD_InitPins(void) {
  CLOCK_EnableClock(kCLOCK_PortA);                           /* Port A Clock Gate Control: Clock enabled */

  PORT_SetPinMux(PORTA, PIN1_IDX, kPORT_MuxAlt2);            /* PORTA1 (pin 27) is configured as UART0_RX */
  PORT_SetPinMux(PORTA, PIN2_IDX, kPORT_MuxAlt2);            /* PORTA2 (pin 28) is configured as UART0_TX */
  SIM->SOPT5 = ((SIM->SOPT5 &
    (~(SIM_SOPT5_UART0TXSRC_MASK | SIM_SOPT5_UART0RXSRC_MASK))) /* Mask bits to zero which are setting */
      | SIM_SOPT5_UART0TXSRC(SOPT5_UART0TXSRC_UART_TX)       /* UART0 transmit data source select: UART0_TX pin */
      | SIM_SOPT5_UART0RXSRC(SOPT5_UART0RXSRC_UART_RX)       /* UART0 receive data source select: UART0_RX pin */
    );
}

/*******************************************************************************
```

Now:



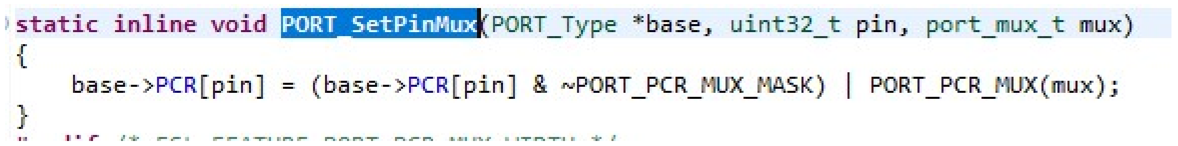
```
void BOARD_InitPins(void)
{
    /* Port A Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortA);
    /* Port D Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortD);

    /* PORTA1 (pin 27) is configured as UART0_RX */
    PORT_SetPinMux(BOARD_INITPINS_DEBUG_UART_RX_PORT, BOARD_INITPINS_DEBUG_UART_RX_PIN, kPORT_MuxAlt2);

    /* PORTA2 (pin 28) is configured as UART0_TX */
    PORT_SetPinMux(BOARD_INITPINS_DEBUG_UART_TX_PORT, BOARD_INITPINS_DEBUG_UART_TX_PIN, kPORT_MuxAlt2);

    /* PORTD1 (pin 74) is configured as PTD1 */
    PORT_SetPinMux(BOARD_INITPINS_LED_BLUE_PORT, BOARD_INITPINS_LED_BLUE_PIN, kPORT_MuxAsGpio);

    SIM->SOPT5 = ((SIM->SOPT5 &
                   /* Mask bits to zero which are setting */
                   (~(SIM_SOPT5_UART0TXSRC_MASK | SIM_SOPT5_UART0RXSRC_MASK)))

                  /* UART0 transmit data source select: UART0_TX pin. */
                  | SIM_SOPT5_UART0TXSRC(SOPT5_UART0TXSRC_UART_TX)

                  /* UART0 receive data source select: UART0_RX pin. */
                  | SIM_SOPT5_UART0RXSRC(SOPT5_UART0RXSRC_UART_RX));
}
```

Expanding clock enable:



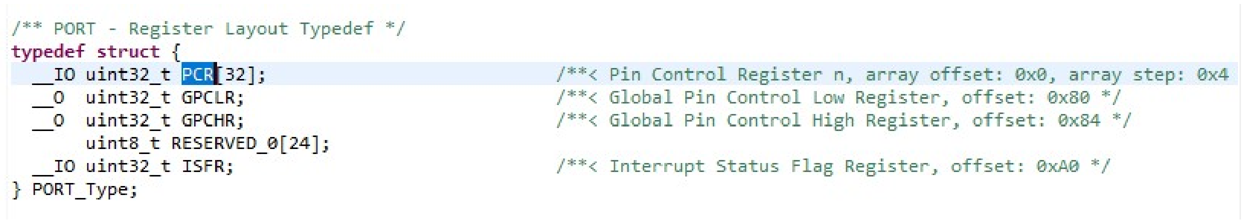
```
static inline void CLOCK_EnableClock(clock_ip_name_t name)
{
    uint32_t regAddr = SIM_BASE + CLK_GATE_ABSTRACT_REG_OFFSET((uint32_t)name);
    (*(volatile uint32_t *)regAddr) |= (1U << CLK_GATE_ABSTRACT_BITS_SHIFT((uint32_t)name));
}
```

Expanding the function Port_setPinMux from fsl_port.h Please note, the PCR, or pin control register has been used to mux the pin.

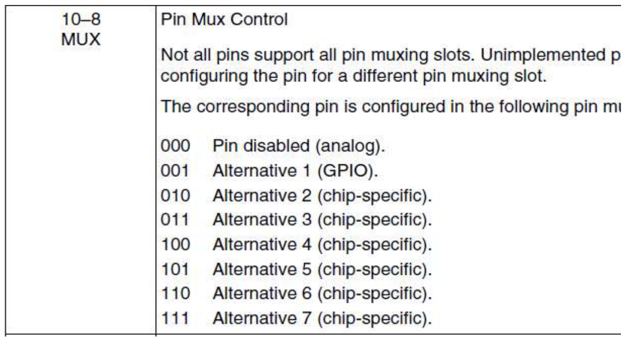


```
static inline void PORT_SetPinMux(PORT_Type *base, uint32_t pin, port_mux_t mux)
{
    base->PCR[pin] = (base->PCR[pin] & ~PORT_PCR_MUX_MASK) | PORT_PCR_MUX(mux);
}
```

Port Memory Map.



```
/** PORT - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCR[32];                          /**< Pin Control Register n, array offset: 0x0, array step: 0x4
  __O  uint32_t GPCLR;                            /**< Global Pin Control Low Register, offset: 0x80 */
  __O  uint32_t GPCHR;                            /**< Global Pin Control High Register, offset: 0x84 */
       uint8_t RESERVED_0[24];
  __IO uint32_t ISFR;                             /**< Interrupt Status Flag Register, offset: 0xA0 */
} PORT_Type;
```

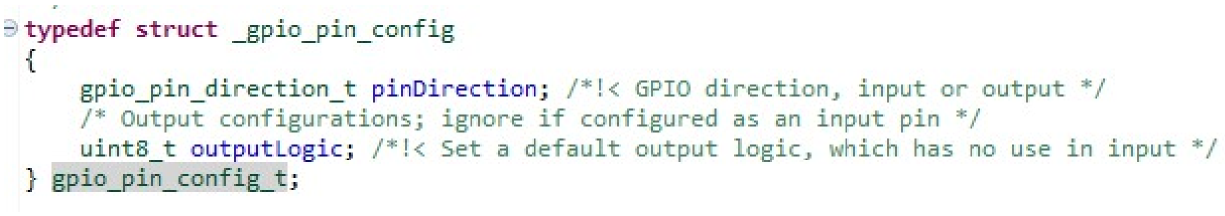


10–8 MUX	Pin Mux Control Not all pins support all pin muxing slots. Unimplemented p configuring the pin for a different pin muxing slot. The corresponding pin is configured in the following pin mu 000 Pin disabled (analog). 001 Alternative 1 (GPIO). 010 Alternative 2 (chip-specific). 011 Alternative 3 (chip-specific). 100 Alternative 4 (chip-specific). 101 Alternative 5 (chip-specific). 110 Alternative 6 (chip-specific). 111 Alternative 7 (chip-specific).

I will let you do the math and verify.

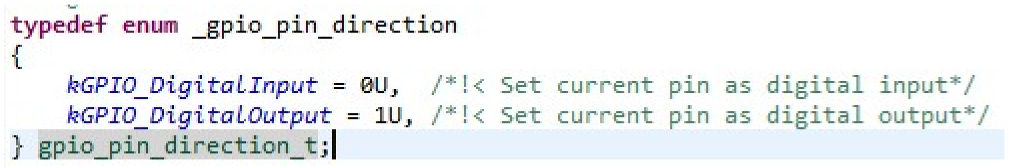
SIDE NOTE: You can see what inside any function by hovering you mouse over it or by pressing CTRL and then clicking on the function. It takes you to the file containing the function.

You can see from above that as of now we have accomplished 2 things, clock enabling for PORT D and Setting the Pin mux, Now we configure the pin as input or output. For this we have a struct in 'fsl_gpio.h' called gpio_pin_config_t



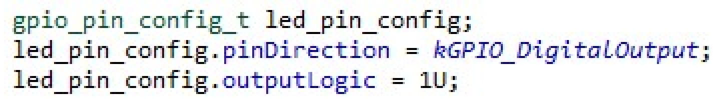
```
typedef struct _gpio_pin_config
{
    gpio_pin_direction_t pinDirection; /*!< GPIO direction, input or output */
    /* Output configurations; ignore if configured as an input pin */
    uint8_t outputLogic; /*!< Set a default output logic, which has no use in input */
} gpio_pin_config_t;
```

The gpio_pin_direction_t is an enum which has 2 elements



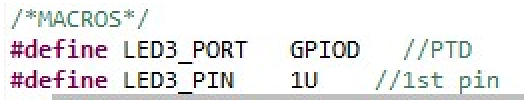
```
typedef enum _gpio_pin_direction
{
    kGPIO_DigitalInput = 0U,  /*!< Set current pin as digital input*/
    kGPIO_DigitalOutput = 1U, /*!< Set current pin as digital output*/
} gpio_pin_direction_t;
```

So, we call this struct in out main program and populate it. The pin's direction needs to be configured as an output

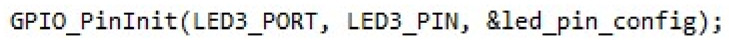


```
gpio_pin_config_t led_pin_config;
led_pin_config.pinDirection = kGPIO_DigitalOutput;
led_pin_config.outputLogic = 1U;
```

At this point we just populated the struct, we need to call this in a GPIO init function which internally sets it's direction.



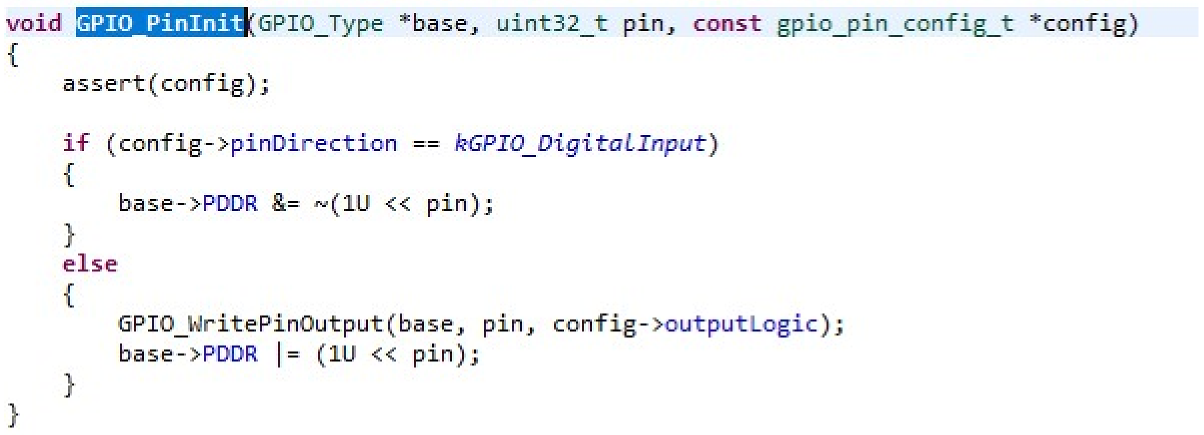
```
/*MACROS*/
#define LED3_PORT   GPIOD   //PTD
#define LED3_PIN    1U    //1st pin
```



```
GPIO_PinInit(LED3_PORT, LED3_PIN, &led_pin_config);
```

It's always better to put the PORTS and PIN in a MACRO so that if we ever must change it, we can just go and change the MACRO value.

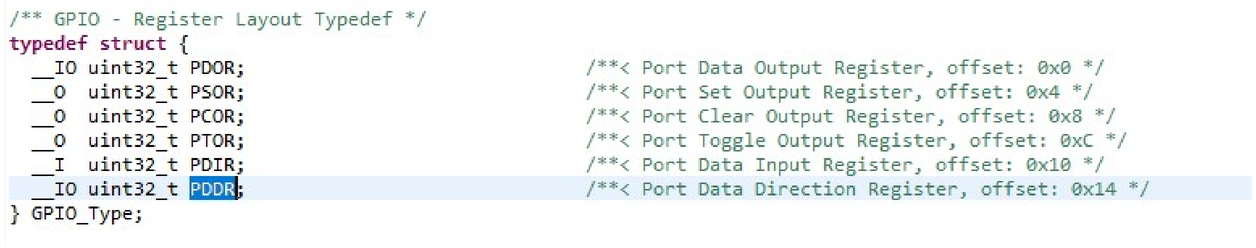
When we go inside the GPIO_PinInit function we see that it basically changes the PDDR register.



```
void GPIO_PinInit(GPIO_Type *base, uint32_t pin, const gpio_pin_config_t *config)
{
    assert(config);

    if (config->pinDirection == kGPIO_DigitalInput)
    {
        base->PDDR &= ~(1U << pin);
    }
    else
    {
        GPIO_WritePinOutput(base, pin, config->outputLogic);
        base->PDDR |= (1U << pin);
    }
}
```

PDDR is port data direction register. Setting it 1 makes the pin as output , clearing it makes it input.

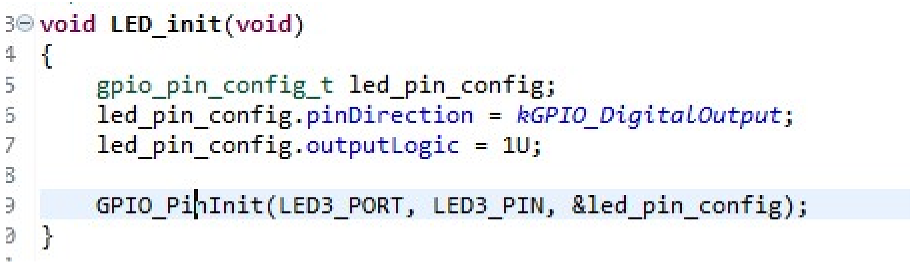
It's worth while to look at the gpio register overlay



```
/** GPIO - Register Layout Typedef */
typedef struct {
  __IO uint32_t PDOR;                      /**< Port Data Output Register, offset: 0x0 */
  __O  uint32_t PSOR;                      /**< Port Set Output Register, offset: 0x4 */
  __O  uint32_t PCOR;                      /**< Port Clear Output Register, offset: 0x8 */
  __O  uint32_t PTOR;                      /**< Port Toggle Output Register, offset: 0xC */
  __I  uint32_t PDIR;                      /**< Port Data Input Register, offset: 0x10 */
  __IO uint32_t PDDR;                      /**< Port Data Direction Register, offset: 0x14 */
} GPIO_Type;
```

Many of the gpio based APIs in fsl_gpio.c will internally set or reset one of these registers.

At this point we can group all the steps taken for Configuring the LED into a function called LED_init()

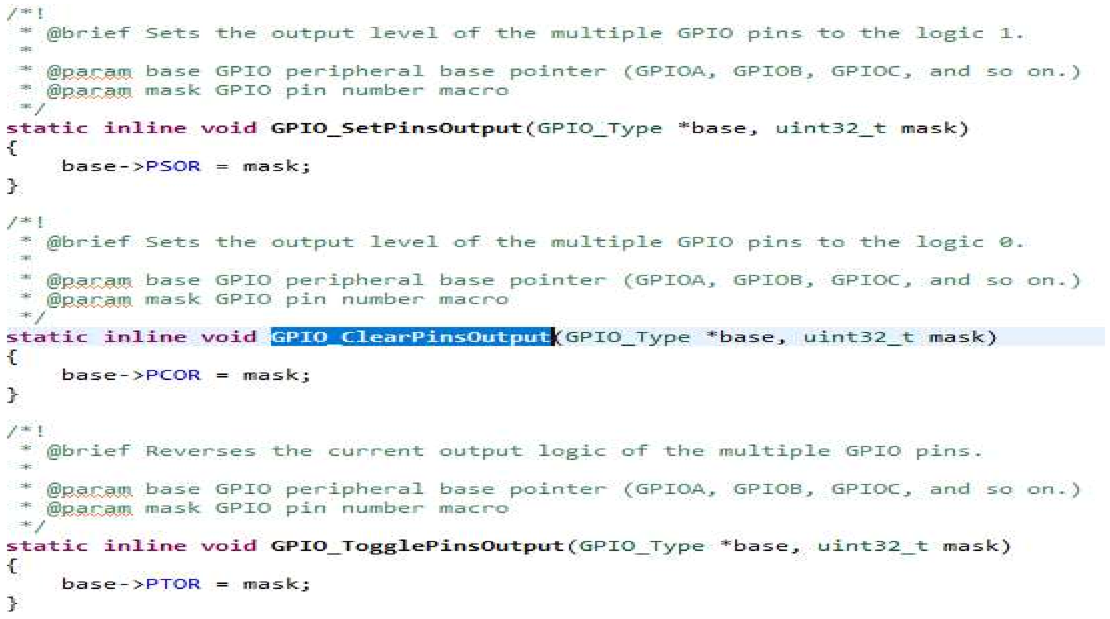


```
void LED_init(void)
{
    gpio_pin_config_t led_pin_config;
    led_pin_config.pinDirection = kGPIO_DigitalOutput;
    led_pin_config.outputLogic = 1U;

    GPIO_PinInit(LED3_PORT, LED3_PIN, &led_pin_config);
}
```

Now we need to set and clear the value of the pin with a delay in between to create a toggling effect. For that we use the following 2 functions:

```
GPIO_ClearPinsOutput(LED3_PORT, 1 << LED3_PIN);
```

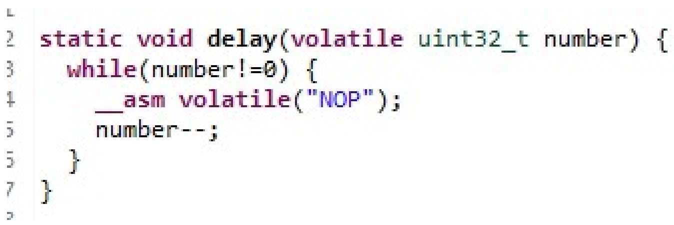
```
GPIO_SetPinsOutput(LED3_PORT, 1 << LED3_PIN);
```



```
/*!
 * @brief Sets the output level of the multiple GPIO pins to the logic 1.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
static inline void GPIO_SetPinsOutput(GPIO_Type *base, uint32_t mask)
{
    base->PSOR = mask;
}

/*!
 * @brief Sets the output level of the multiple GPIO pins to the logic 0.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
static inline void GPIO_ClearPinsOutput(GPIO_Type *base, uint32_t mask)
{
    base->PCOR = mask;
}

/*!
 * @brief Reverses the current output logic of the multiple GPIO pins.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
static inline void GPIO_TogglePinsOutput(GPIO_Type *base, uint32_t mask)
{
    base->PTOR = mask;
}
```

As you can see the functions use the set, clear and toggle registers from the gpio register layout typedef.

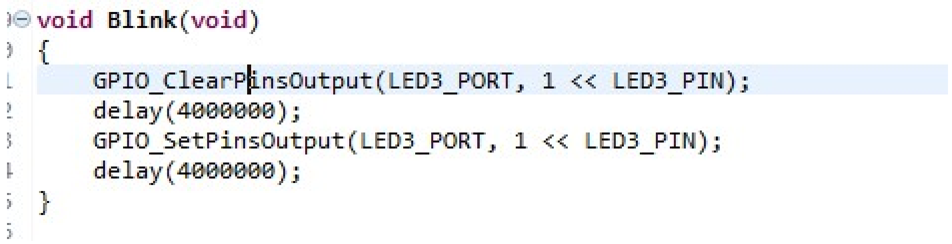
The only remaining thing at this point is the delay, In the function below, I am giving a delay of 4 million cpu cycles. The function _asm volatile("NOP") does no operation other than consume a cpu cycle,



```
static void delay(volatile uint32_t number) {
  while(number!=0) {
    __asm volatile("NOP");
    number--;
  }
}
```

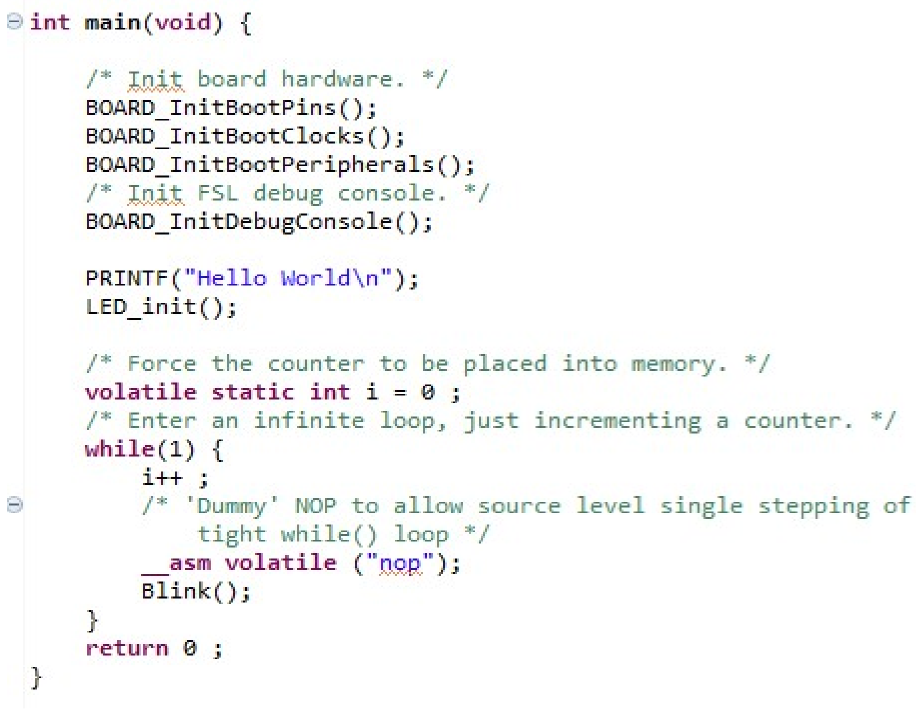
I have used volatile as some compilers can optimize out the operation which will not give us the delay we need.

Combining All the above, we have another function called blink:



```
void Blink(void)
{
    GPIO_ClearPinsOutput(LED3_PORT, 1 << LED3_PIN);
    delay(4000000);
    GPIO_SetPinsOutput(LED3_PORT, 1 << LED3_PIN);
    delay(4000000);
}
```

We are done at this point. We can simply call the functions in relevant places.

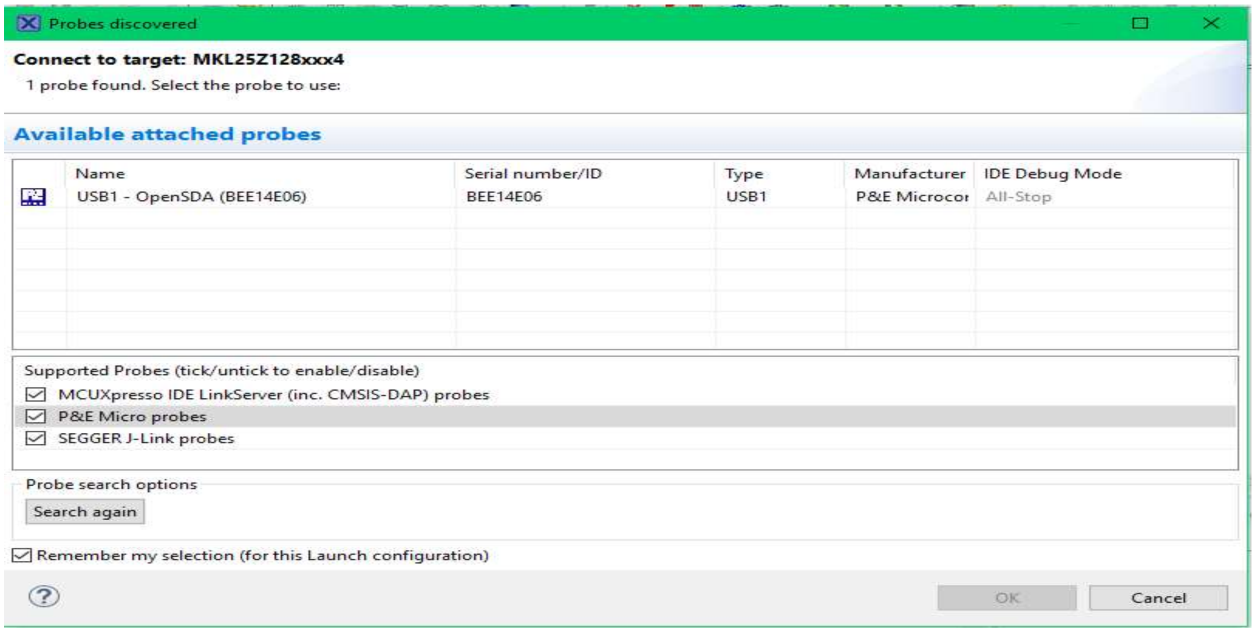


```
int main(void) {

    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
    /* Init FSL debug console. */
    BOARD_InitDebugConsole();

    PRINTF("Hello World\n");
    LED_init();

    /* Force the counter to be placed into memory. */
    volatile static int i = 0 ;
    /* Enter an infinite loop, just incrementing a counter. */
    while(1) {
        i++ ;
        /* 'Dummy' NOP to allow source level single stepping of
            tight while() loop */
        __asm volatile ("nop");
        Blink();
    }
    return 0 ;
}
```

Now we build the project. For that select the project you wish to build from the Project View or workspace, right click on it, then click ‘Build Project’ or simply select the project and click the hammer icon from the menu bar.

If the build is successful without any errors, then click the blue bug from the menu bar. This puts our project in the debug mode.

You should see your probe in this window that pops up next. If not remove and plug in again.



Probes discovered

Connect to target: MKL25Z128xxx4

1 probe found. Select the probe to use:

Available attached probes

Name	Serial number/ID	Type	Manufacturer	IDE Debug Mode
USB1 - OpenSDA (BEE14E06)	BEE14E06	USB1	P&E Microcor	All-Stop

Supported Probes (tick/untick to enable/disable)

☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
☑ P&E Micro probes
☑ SEGGER J-Link probes

Probe search options

Search again

☑ Remember my selection (for this Launch configuration)

Select -> okay.

After successful debug, to run the program, Click on and to stop the debugger and return back to the program . In case you feel like you need to make some changes, terminate the debugger pressing the red square icon mentioned, make changes, build again and then debug and run.

Congratulations! The blue LED is blinking!