Project 0: Setup FRDM-KL46Z

Check the FRDM Board

Follow the instructions in the link to verify that the board is working.

Install Required Software

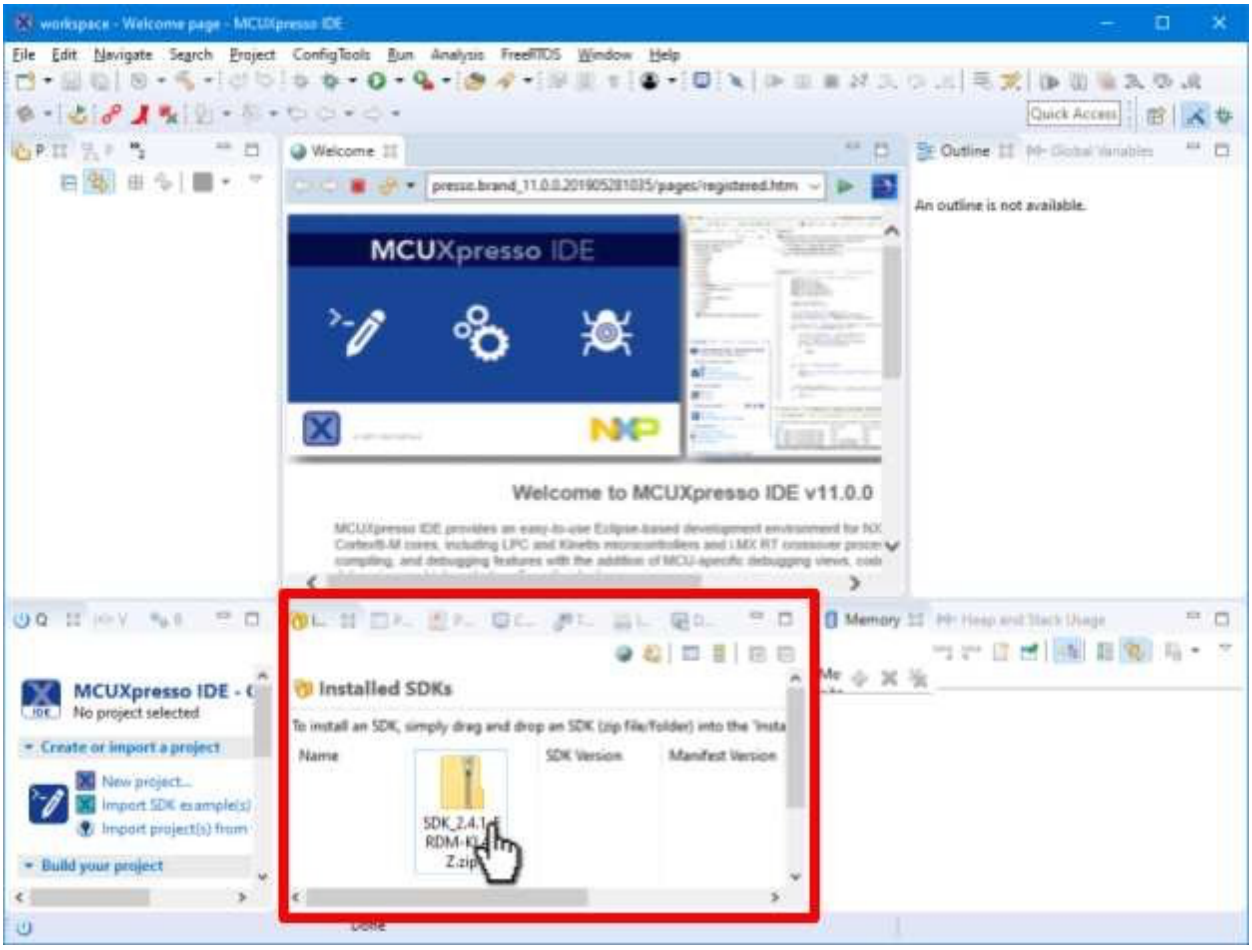
Install each of the following software packages for your OS (in this order):

1. PE Micro OpenSDA Drivers
    - Windows
    - Linux
    - (Not needed on Mac)
2. MCUXpresso IDE
    - Windows
    - Linux
    - Mac
3. Download, but don't unzip, MCUXpresso SDK Files
    - Windows
    - Linux
    - Mac

Setup a Project

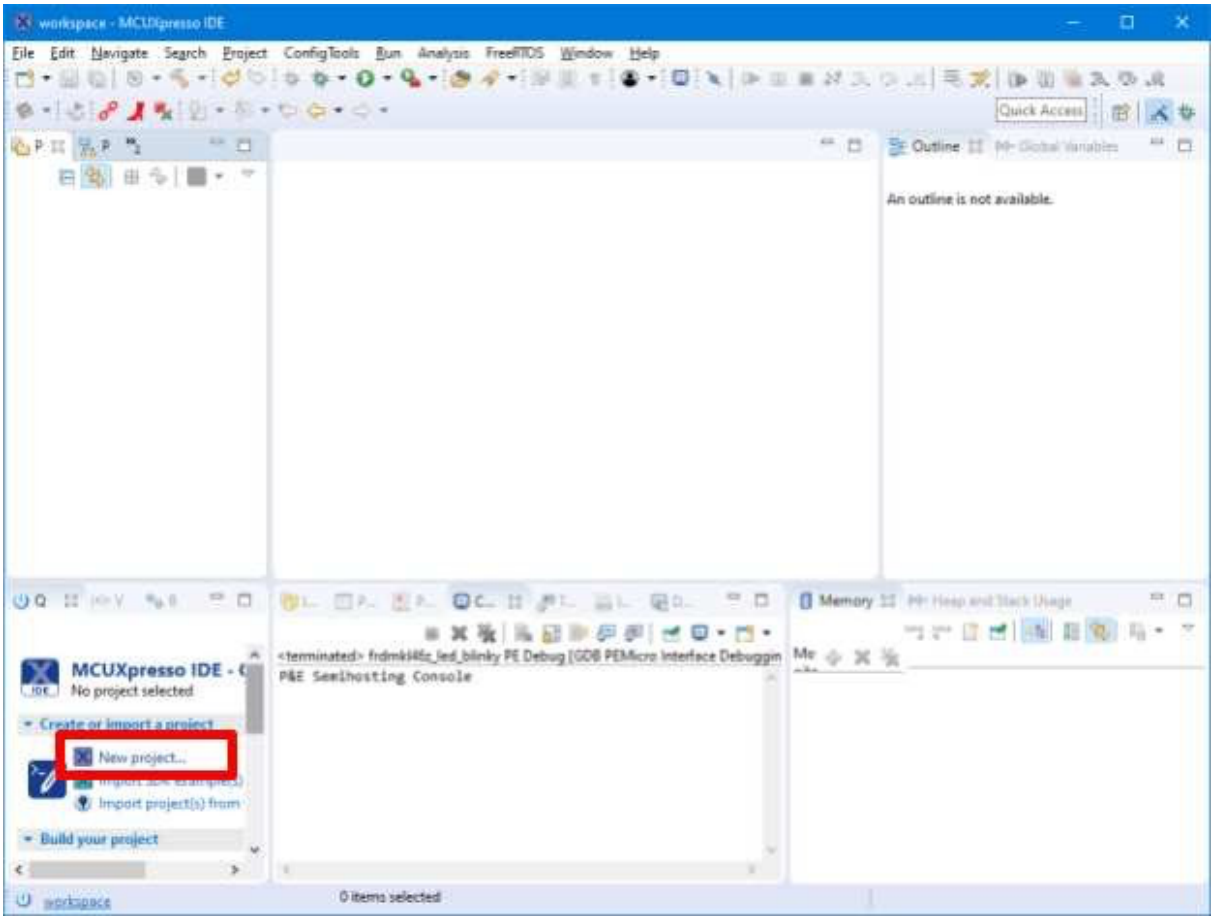
After installing MCUXpresso, start the program.

Install the FRDM-KL46Z SDK by dragging and dropping the downloaded SDK file onto the "Installed SDK's" window of MCUXpresso IDE.

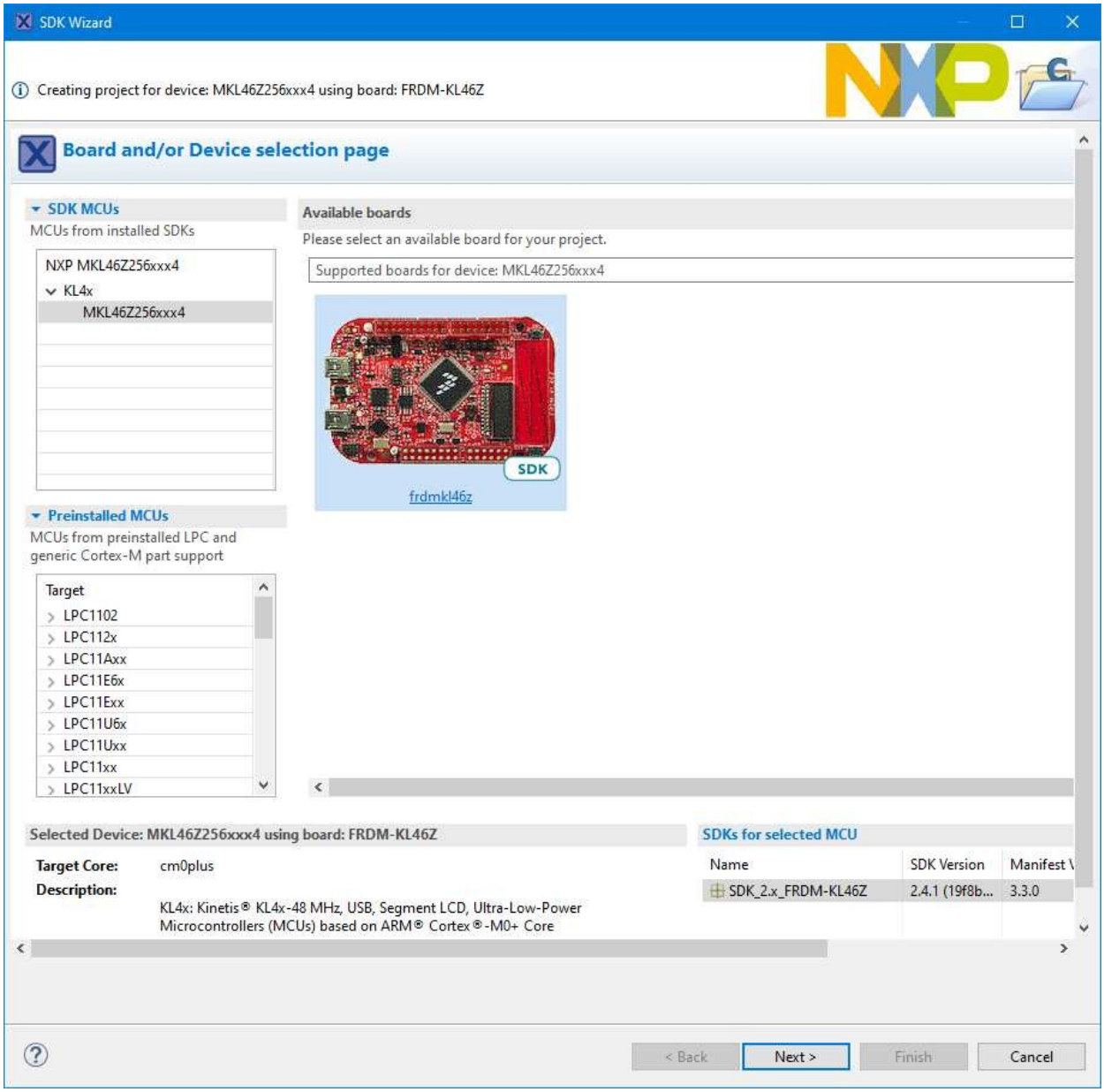


Approve the questions it asks, and you should see the SDK listing appear after import is complete.

Start a new project by clicking "New Project" in the Quickstart Menu



Select our board in the Board / Device Selection Page, then click next:



In the "Configure the Project" page, in the Project Name field, replace with "Blinky Hello World". Do not change anything else, click "Finish"

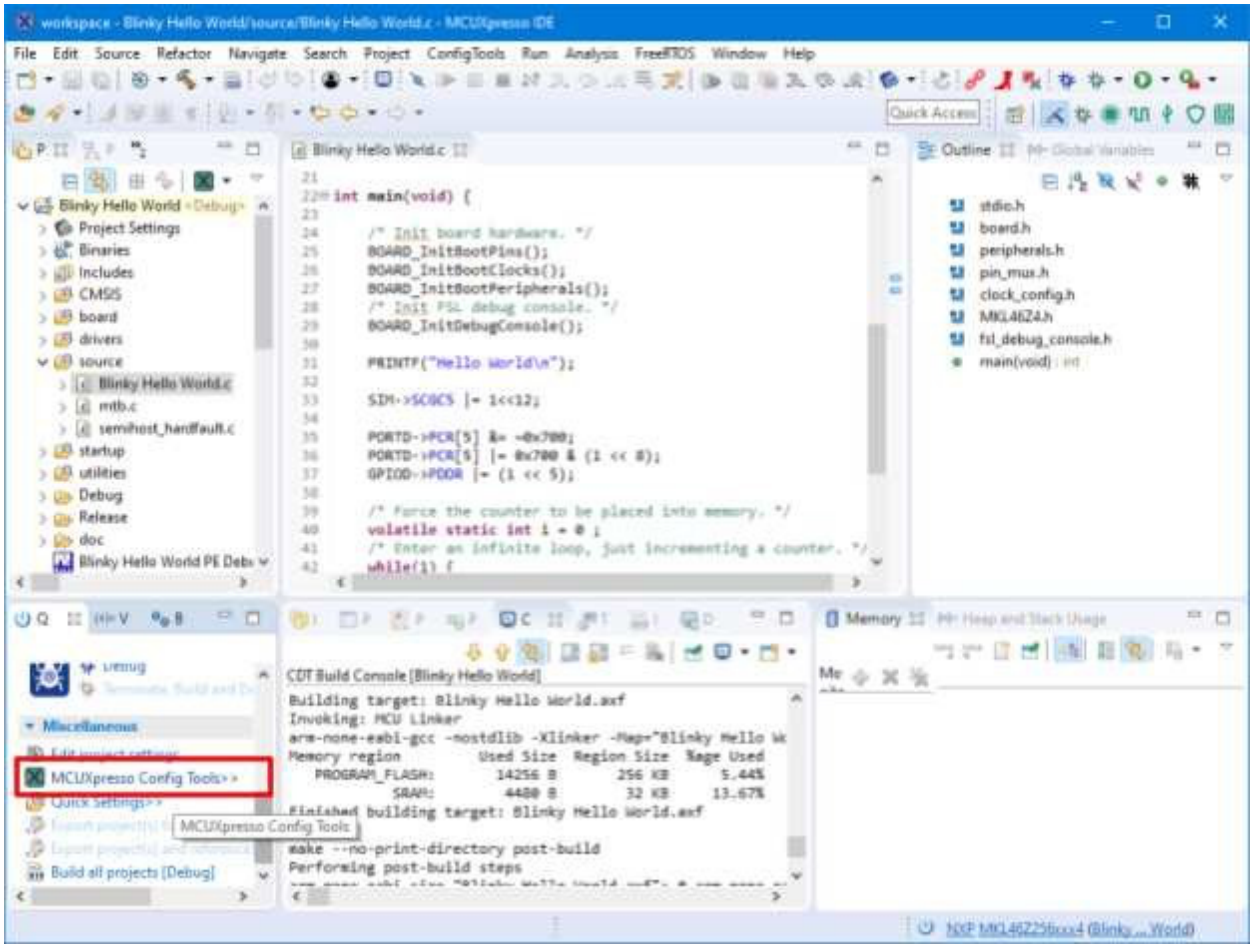
Add the highlighted new lines to the file:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL46Z4.h"
```

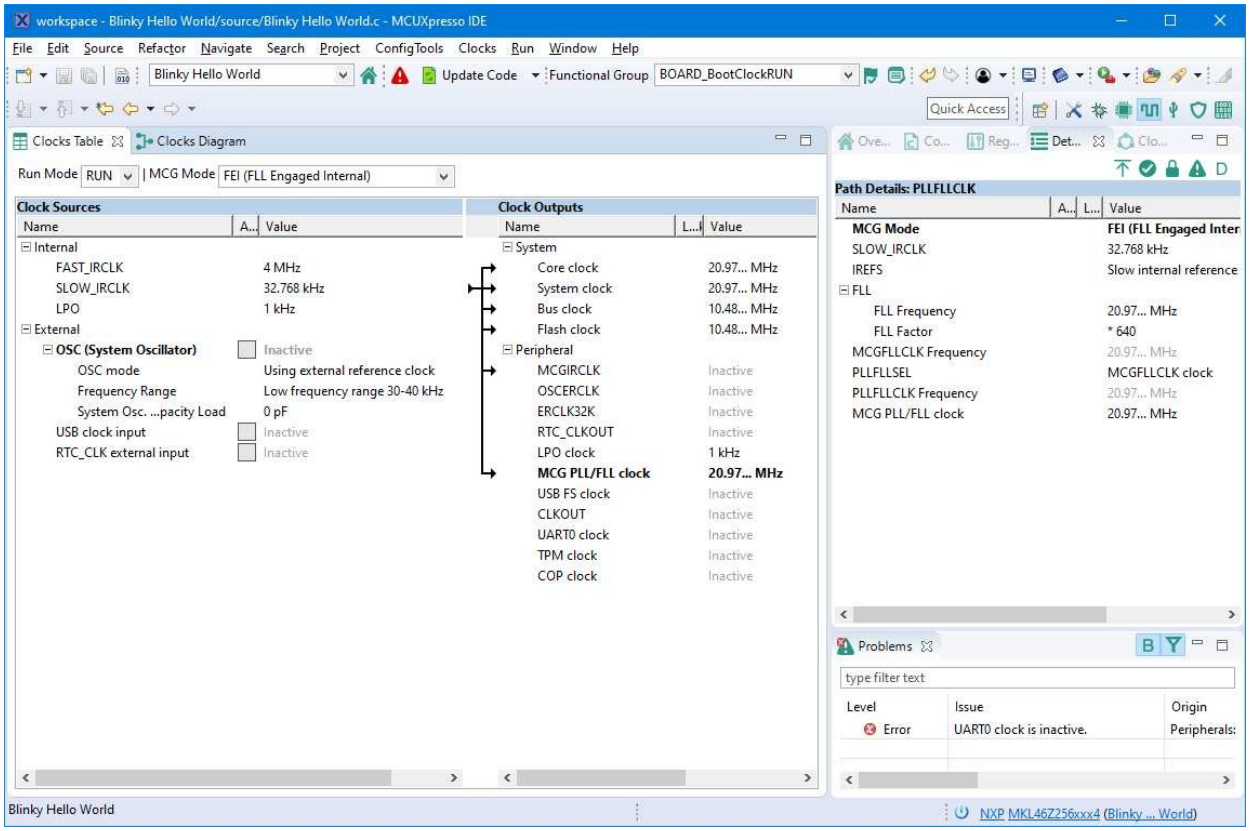
```
#include "fsl_debug_console.h"

/* 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");

     SIM->SCGC5 |= 1<<12;

     PORTD->PCR[5] &= ~0x700;
     PORTD->PCR[5] |= 0x700 & (1 << 8);
     GPIO->PDDR |= (1<<5);


    /* 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");

        if (i % 100000 == 0) {
                GPIOD->PTOR |= (1<<5);
            }

    }
    return 0 ;
}
```

Setup the Clock

In the Quickstart panel, select MCUXpresso Config Tools >> Open Clocks



This will bring up the Clock Configuration tools. Configuring the clocks for most devices is very complex, so we're going to use this "Wizard" style tool to configure the system and check to see if there are any errors.



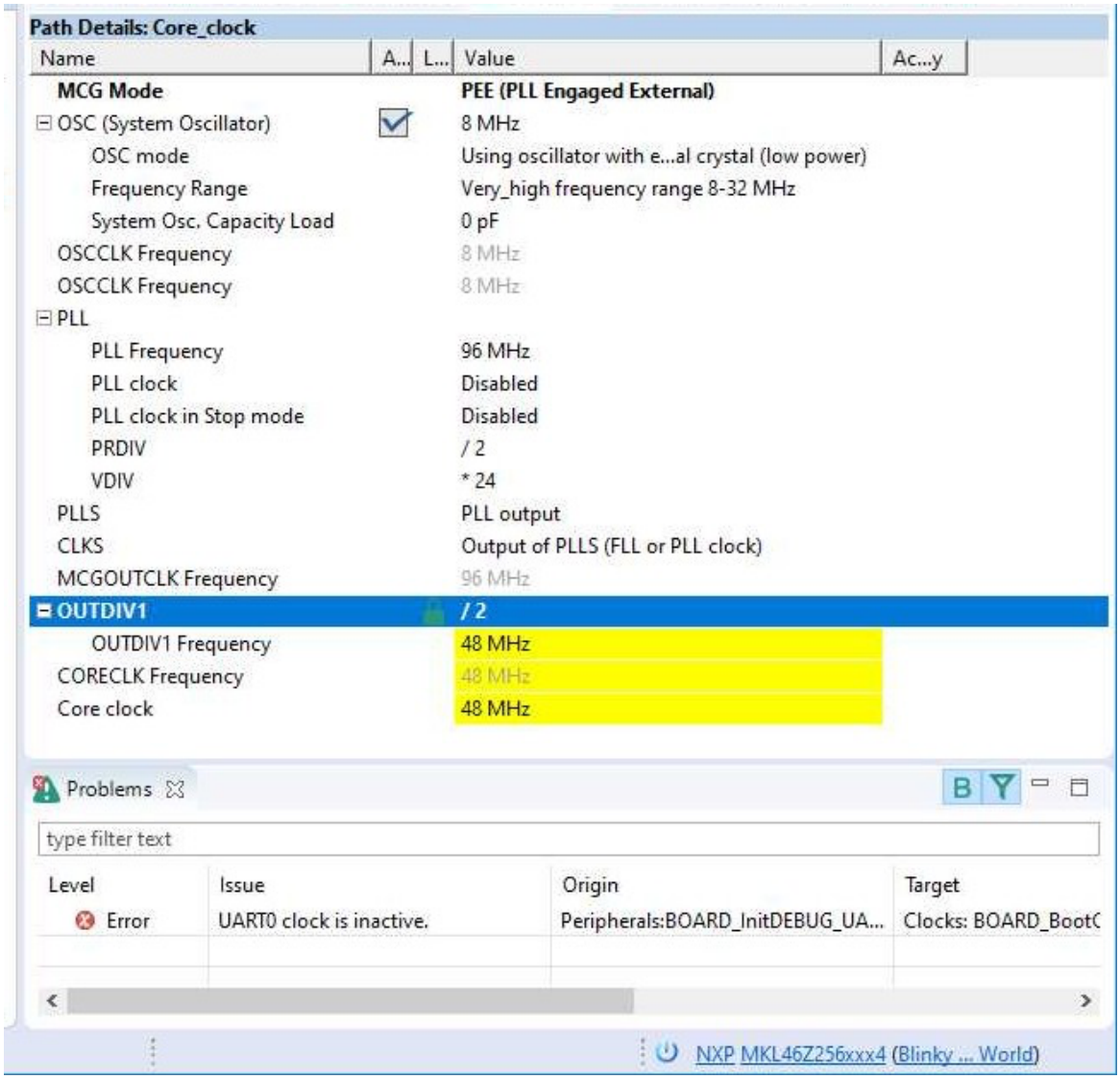
On this first view we have an overview of the current clock configuration. We're trying to maximize the performance of the device, which can handle up to a 48 MHz clock signal. We can see that with the default settings, our Core / System clock is running at 20.87MHz.

First, in the **clock sources pane**, click the box next to OSC (System Oscillator) and set the clock to these settings:
Frequency: 8MHz
OSC Mode: Using oscillator with external crystal (low power)
Frequency Range: Very High Frequency Range 8-32 MHz

Next, in the **clock outputs pane, click "Core Clock"**, and update these settings:

MCG Mode: PEE (PLL Engaged External)
OUTDIV1: /2

All the remaining settings should be at their defaults:

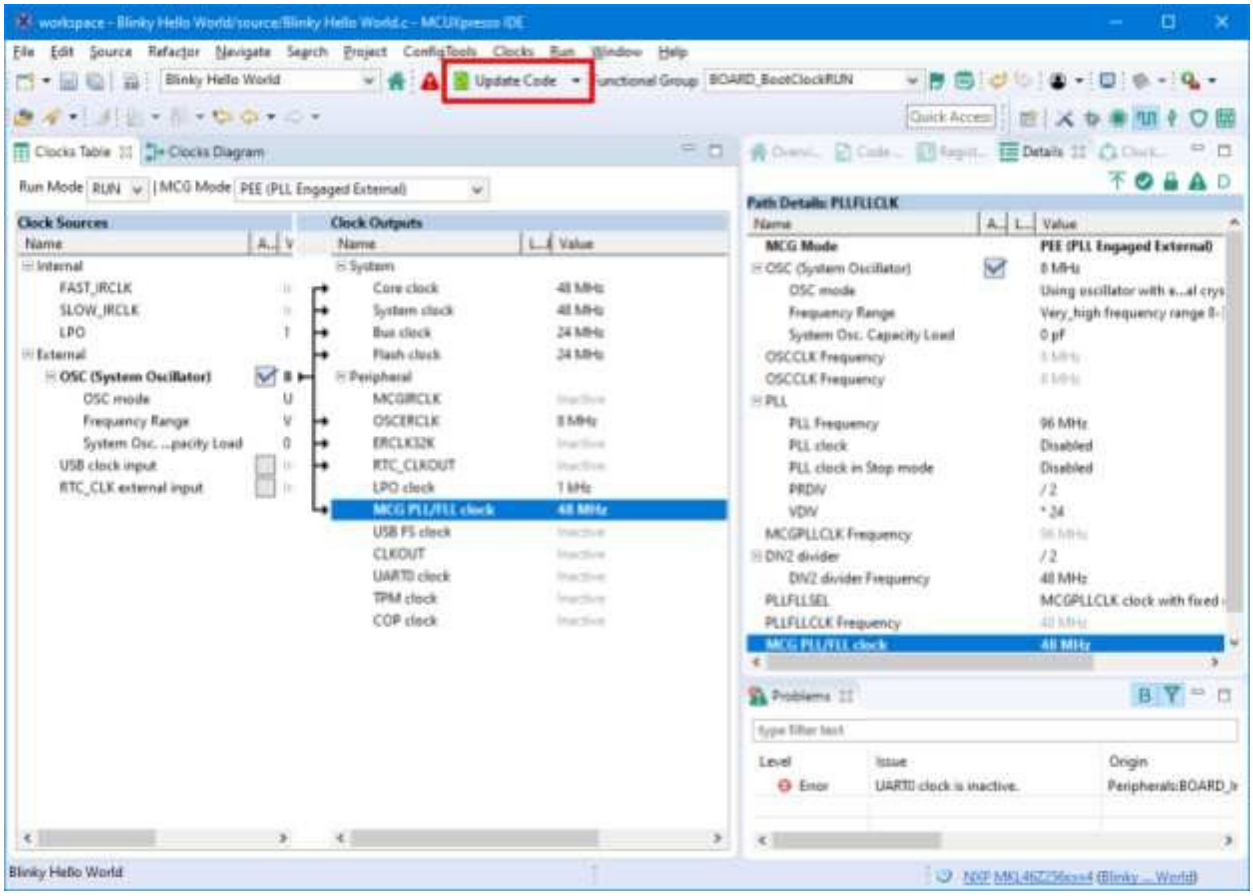


Now, the Core and System clocks should be 48MHz. We still need to set some internal clock sources. Next, in the **clock outputs pane, click "OSCERCLK"**. In the Path Details pane, set **OSCERCLK Output to Enabled.** You should see the OSCERCLK frequency change to 8 MHz.

Finally, in the **clock outputs pane, click "MCG PLL/FLL clock".** In the Path Details pane, change the **PLLFLLSEL to "MCGPLLCLK clock with fixed divide by 2"** The MCG PLL/FLL clock should now be 48MHz as well.

You should have only 1 error for the "UART0 clock is inactive." You can ignore this error safely. If you have other errors, you should review the above steps before continuing.

To finish and save the changes, click the "Update Code" button.

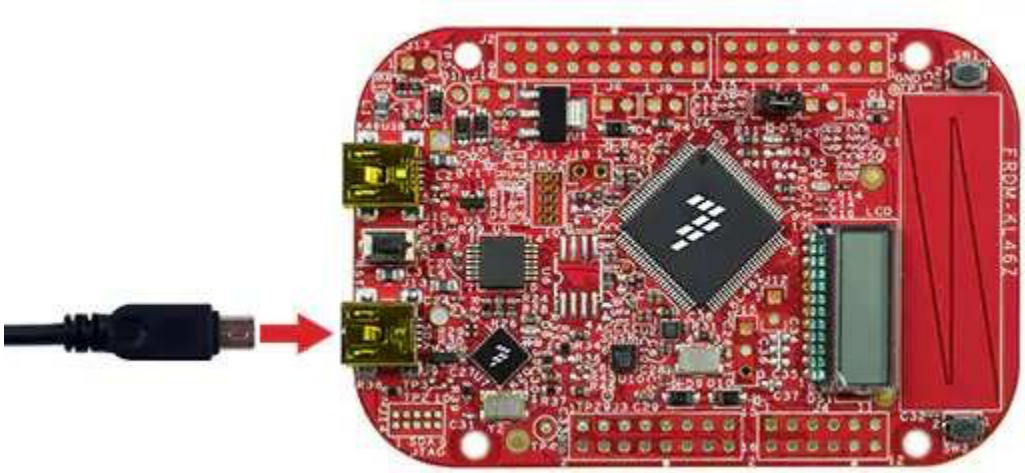


You should only have an error icon on the "Peripherals" category, you can ignore this. Click "OK". This will update your hardware clock configuration, and return you to the code view.

You must do all of the above steps EVERY time you create a new project.

Run your Program!

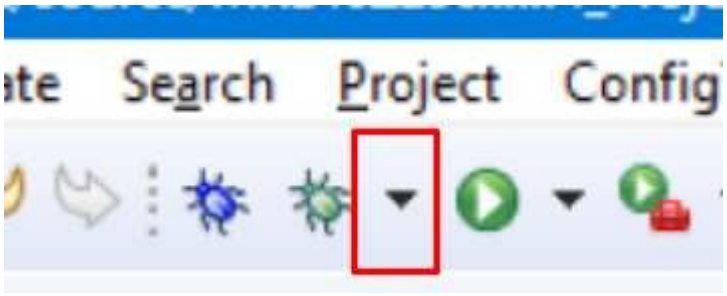
Using the provided USB Cable, plug in your board using the mini USB port marked "SDA":



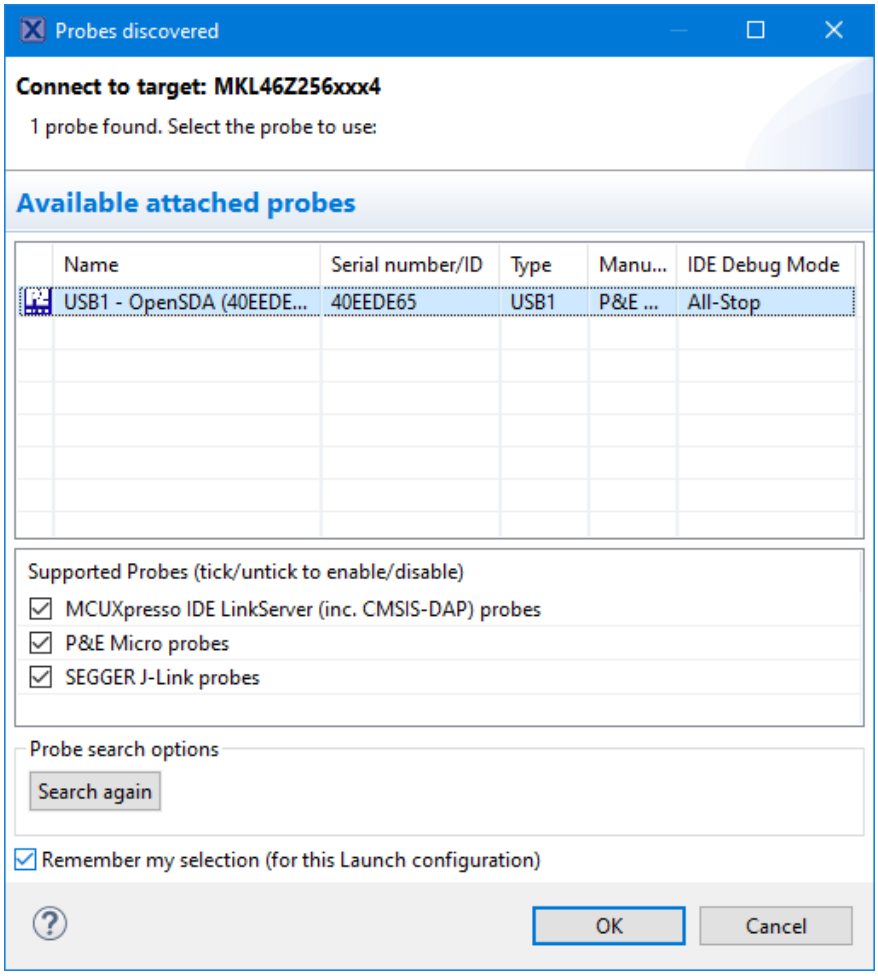
Make sure OpenSDA drivers load completely and successfully.

The board will show up as a storage device "FRDM-KL46Z". Open the drive, and open the SDA_INFO.HTM file. On the page that loads, look at the bootloader version (should be greater than 1.10) and the application version (should be version 1.14 or above). ***If not, give your board to an instructor or TA for firmware update.***

Select the dropdown menu next to the Green "Debug" icon, then select Debug As… -> PEMicro Probes. (You only need to do this once, you can use the Blue Debug icon after this for faster access.)



Then select the "OpenSDA" probe interface and click OK.



The program will be compiled, and loaded onto the FRDM board. The program will automatically stop on the first instruction inside of main: "`BOARD_InitBootPins();`"

To resume running your program, click the Resume button:

Looking back at your board, you should see the green light blinking! This is your program running, congratulations!

You should also notice a "Hello World" in the console tab below your code. This is your program talking to you from the debug interface!

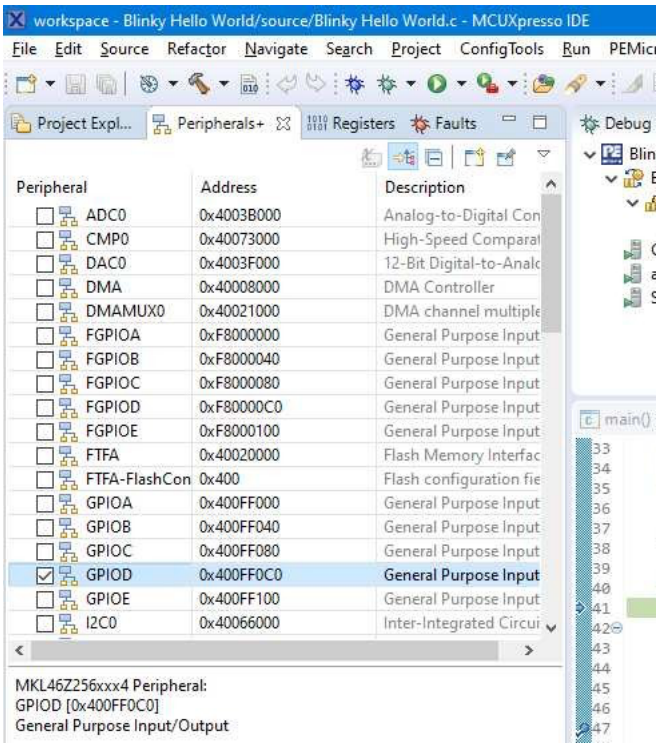
If you are using a Mac, and this step does not work, try restarting your system with the FRDM board connected. Make sure you restart MCUxpresso every time you open your lid and unplug and replug the FRDM USB cable. To keep plugging in the FRDM board until the bin file works when pasted in then try using MCUxpresso.

Now, click the red stop button to halt your debug session.

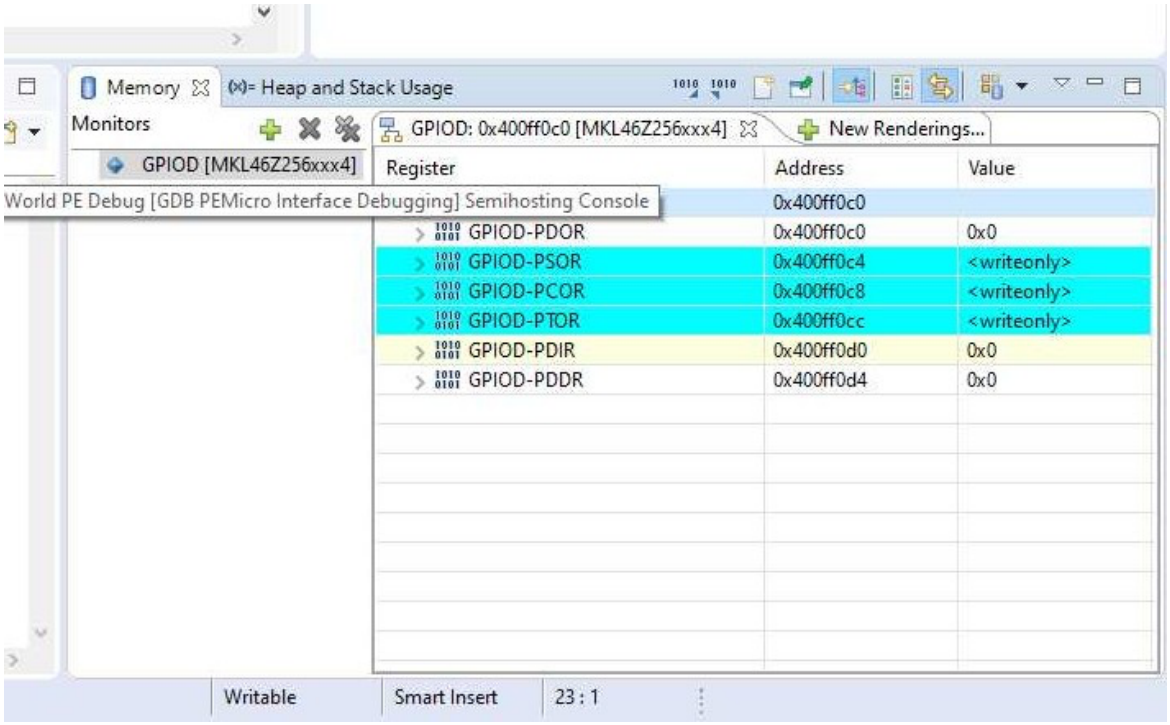
Next, set a breakpoint on the "`GPIOD->PTOR |= (1<<5);`" line by double clicking on the line number.

Click the Blue beetle to debug your program again, then click the resume button to run your program. The program should halt at your breakpoint.

Let's examine the state of memory at this point. In the upper left pane, switch to the "Peripherals" tab, and select the check box next to "GPIOD"



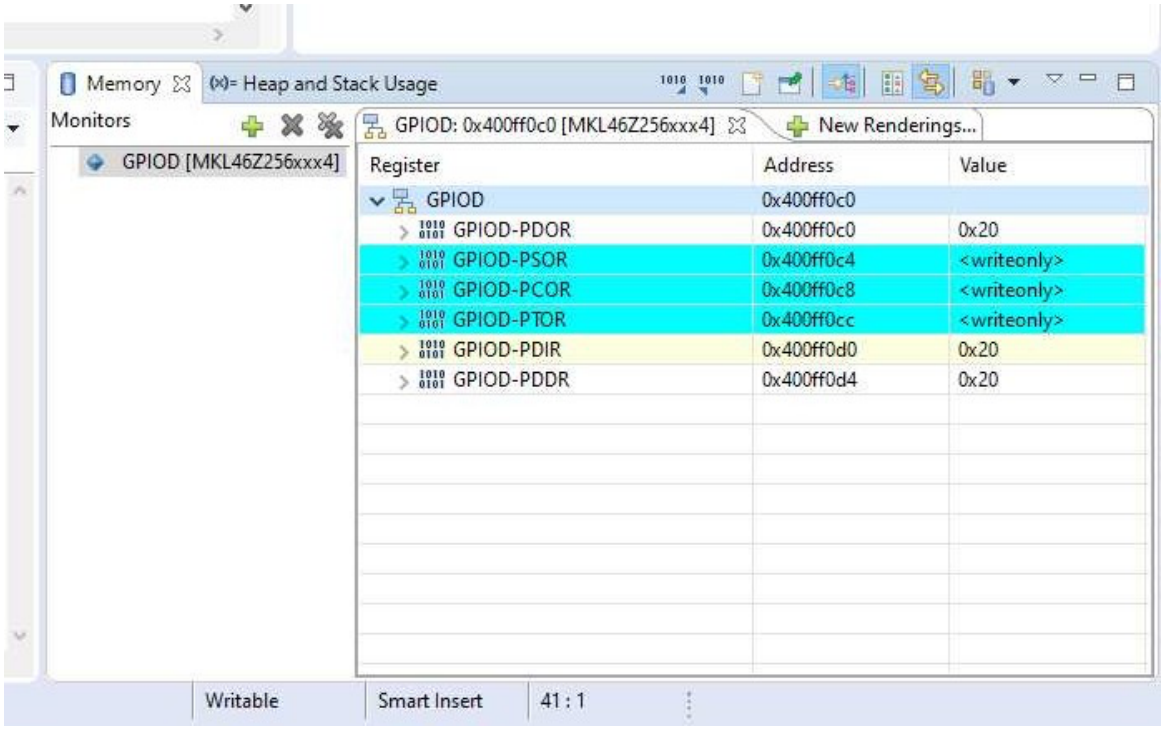
This will cause a new tab to open in the "Memory" pane on the bottom right:



This shows the state of memory in the GPIO peripheral registers. You can see that some registers are "write only" and others will show the current stored value.

Click the "Step Over" button to execute only the current line of code.

Now look back at your board, the LED should have turned off, and the memory state is updated:



These debugging tools are immensely valuable and helpful for identifying errors and flaws in your code, and understanding what the system is doing at every stage of execution. Use them to your advantage!