Lab #1: KSDK Project Generator

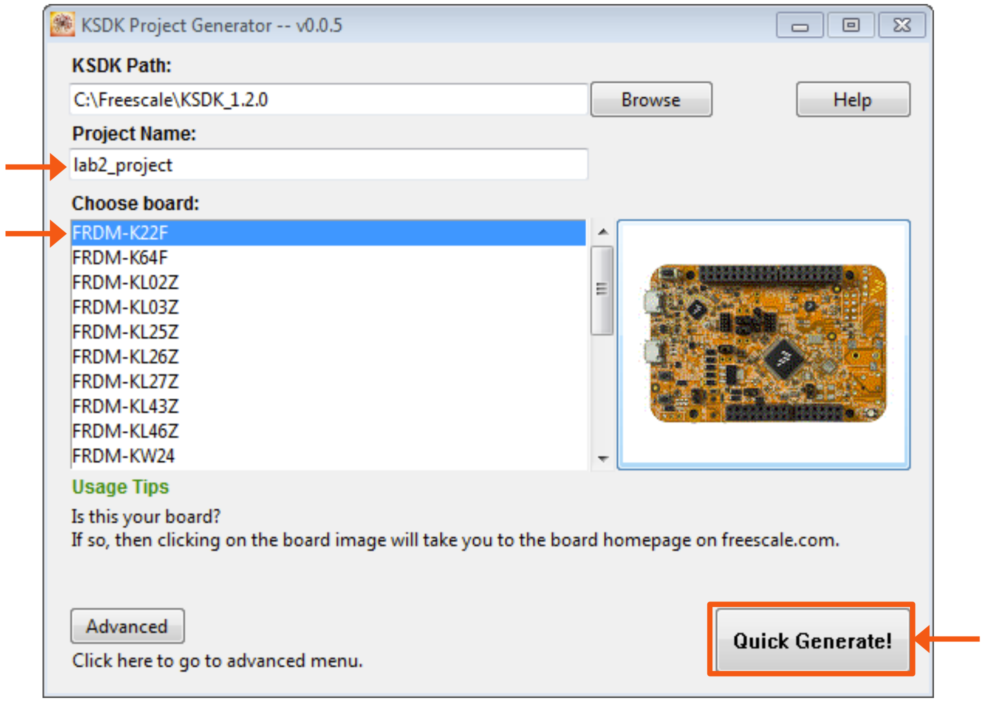
- ✓ **Objective 1 –** Familiarize yourself with the KSDK Project Generator and use the "Quick Generate" feature to create a project
- ✓ **Objective 2 –** Import and build your new project in KDS

KSDK Project Generator – Test Drive

- The Project Generator executable is located in the SDK Advanced folder on your lab computer’s Desktop
- Double click on the EXE file to launch the utility
- Spend some time clicking around and experimenting with the tool
  - Try creating a project using “Quick Generate”
- When you feel comfortable with the tool, proceed to the next slide

Objective 1: Create a Project for the Lab

- Launch the Project Generator (if you haven't already)
- Return to the main window. If you're in the Advanced view, click on the Return button
- Name your project lab2_project, choose the FRDM-K22F board, and click on the Quick Generate! button


KSDK Project Generator -- v0.0.5

KSDK Path:
C:\Freescale\KSDK_1.2.0

Project Name:
lab2_project

Choose board:
FRDM-K22F
FRDM-K64F
FRDM-KL02Z
FRDM-KL03Z
FRDM-KL25Z
FRDM-KL26Z
FRDM-KL27Z
FRDM-KL43Z
FRDM-KL46Z
FRDM-KW24

Usage Tips
Is this your board?
If so, then clicking on the board image will take you to the board homepage on freescale.com.

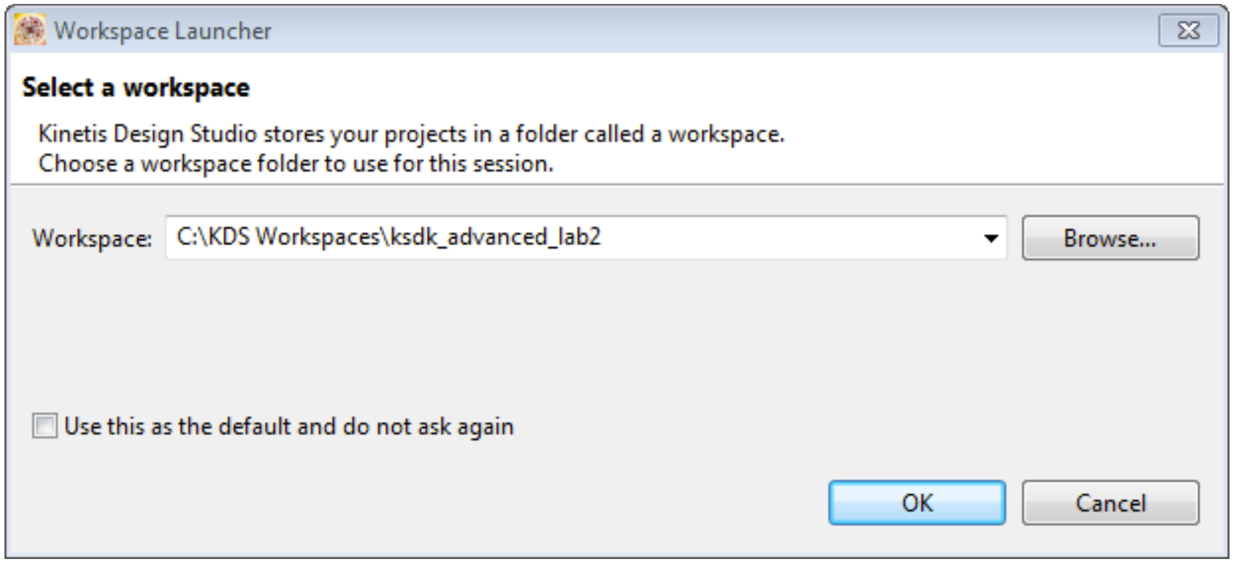
Click here to go to advanced menu.

The new project will be located in:

C:/KSDK_1.2.0/examples/frdmk22f/user_apps

Objective 2: Launch KDS

- Launch Kinetis Design Studio (KDS)
- Create a new workspace. If KDS does not prompt you to select/name a workspace when you launch it, go to File > Switch Workspace > Other and create the new workspace. See an example below:



Kinetis Design Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\KDS Workspaces\ksdk_advanced_lab2

*After selecting "OK", KDS will restart

Objective 2: Import Platform Library Into KDS

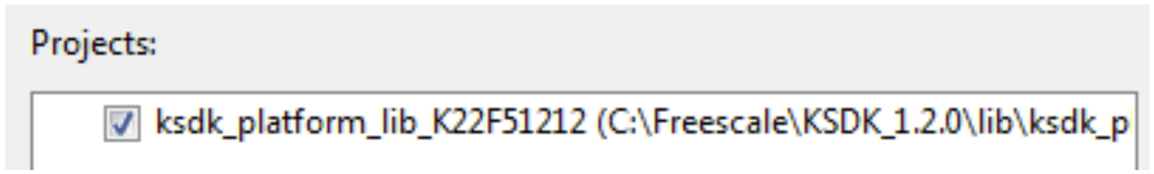
- Go to File > Import and select the Existing Projects into Workspace option under General. Click Next.

- Click on the Browse button next to "Select root directory" and navigate to

  C:/Freescale/KSDK_1.2.0/lib/ksdk_platform_lib/kds/K22F51212

  Click OK.

- The platform library for the K22F512 will now be listed in the Projects window. Click Finish.



Projects:

☑ ksdk_platform_lib_K22F51212 (C:\Freescale\KSDK_1.2.0\lib\ksdk_p

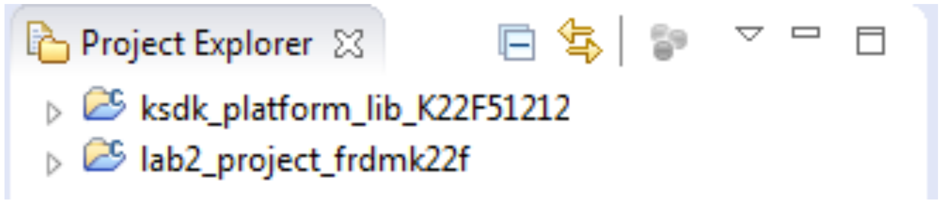
- The project will now be in your workspace

Objective 2: Import Project Into KDS

- Follow the same steps used to import the platform library, only point to:

  C:/Freescale/KSDK_1.2.0/examples/frdmk22f/user_apps/lab2_project/kds

- Your workspace should now contain two projects, as shown below



ksdk_platform_lib_K22F51212
lab2_project_frdmk22f

Objective 2: Build Application

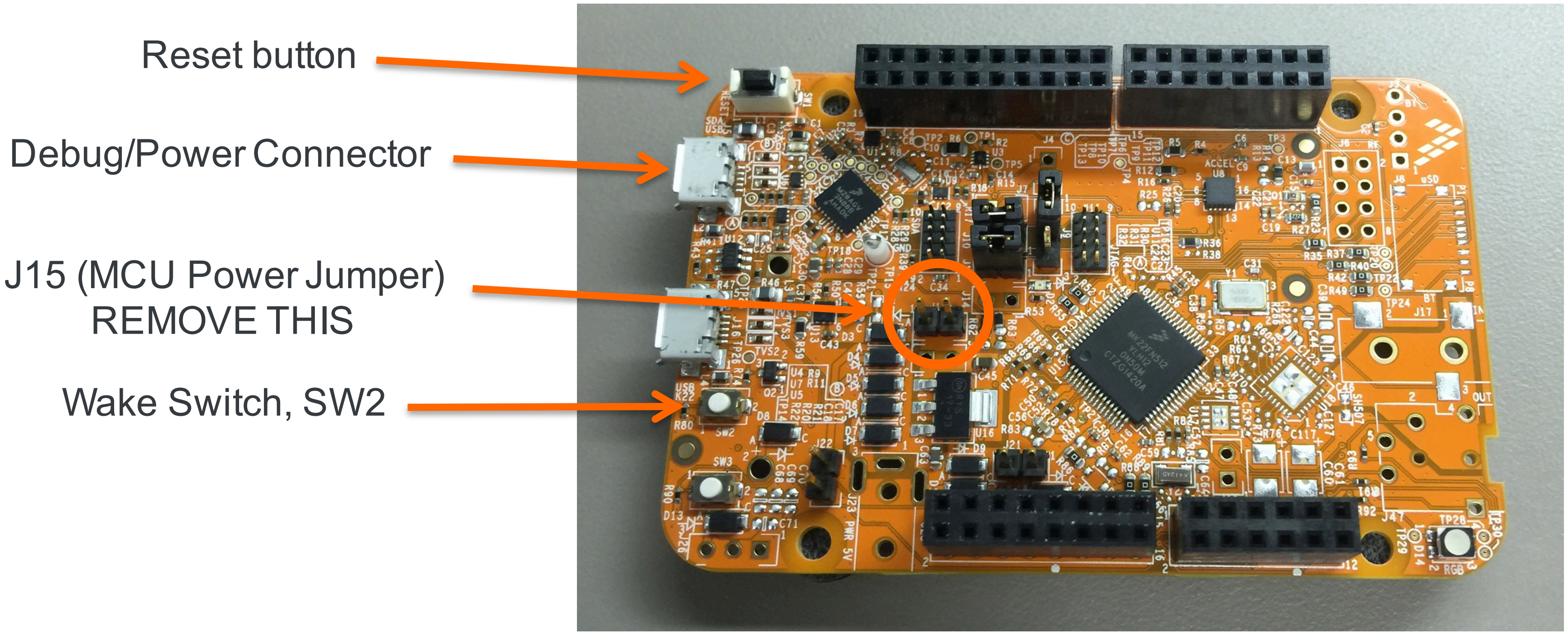
- Before building lab2_project you must build the platform library
- Select/highlight the KSDK platform library project and click on the hammer icon to build
- After the build completes, select/highlight lab2_project and click on the hammer icon
- Both will build without errors or warnings
- This is our starting point. Open up main.c and look at what is included. The source file is in lab2_project > source

Lab #2: System Services

- ✓ **Objective 1 –** Understand default clock configuration
- ✓ **Objective 2 –** Add source files to your new project
- ✓ **Objective 3 –** Run the demo as-is to utilize the Clock and Power Managers and view current consumption
- ✓ **Objective 4 –** Investigate and understand the application code
- ✓ **Objective 5 –** Add a new power mode to the demo application

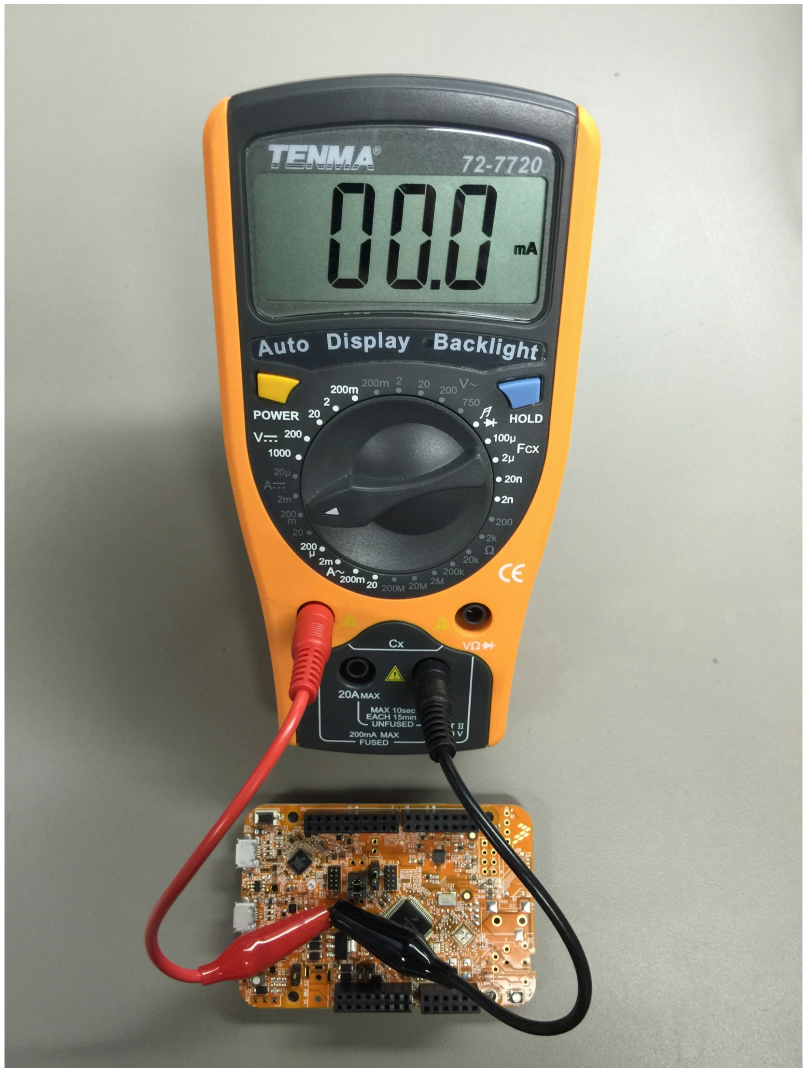
Getting to Know the FRDM-K22F Board

- The important sections of the board are shown below. Before applying power, familiarize yourself with these items



#FTF2015

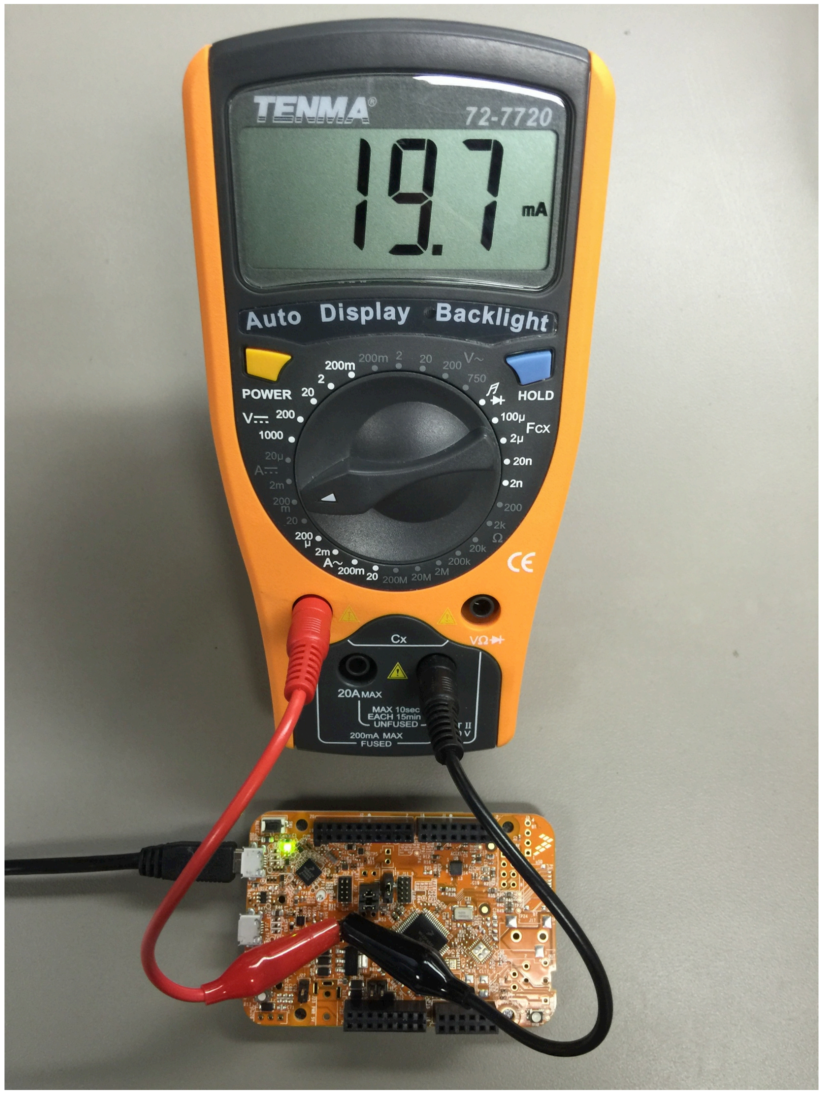
Preparing the Hardware – Part 1



1. Before powering the board, remove jumper J15
2. Connect the meter probes to the low current (200mA) terminals of the meter
3. Connect the probe clips to the MCU power jumper: J15. The positive lead goes to pin 1 (on the USB connector side) and the negative lead goes to pin 2 (on the MCU side)

Preparing the Hardware – Part 2

4. Connect the micro USB cable to the SDA/USB terminal on the board
   - When the board powers up, it will draw in the 10s of milliamps, depending on the application that was previously loaded into the flash memory



Objective 1: Understand Default Clock Settings

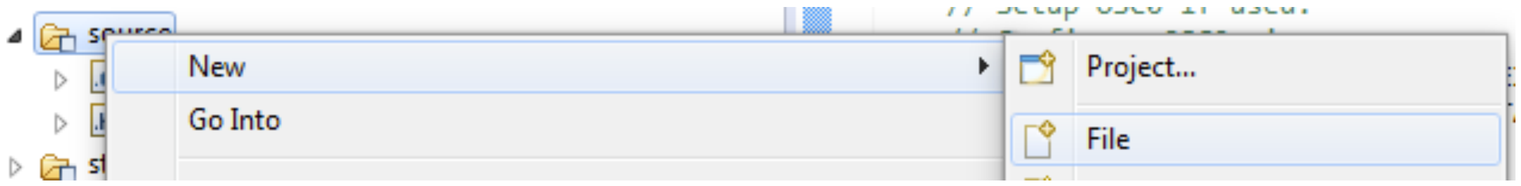
- Open the board.c file, located inside the board folder of lab2_project
- Starting at the top of the file, you'll see a set of pre-configured clock configurations for VLPR, RUN and HSRUN (high-speed run)
- Find the BOARD_ClockInit() function and look at the calls made by the Clock Manager to set up the clock for the project

Test Your Knowledge: What is the default clock frequency used by the project?

Objective 2: Add Source Files

- We have provided a system services example application for you to use in the lab, located in the KSDK Advanced folder on your Desktop
- Copy the lab2.c file from your Desktop into your newly created project in

  C:/Freescale/KSDK_1.2.0/examples/frdmk22f/user_apps/lab2_project
- With the source file copied, import the source file into your project by right-clicking on the source folder in lab2_project and selecting New > File

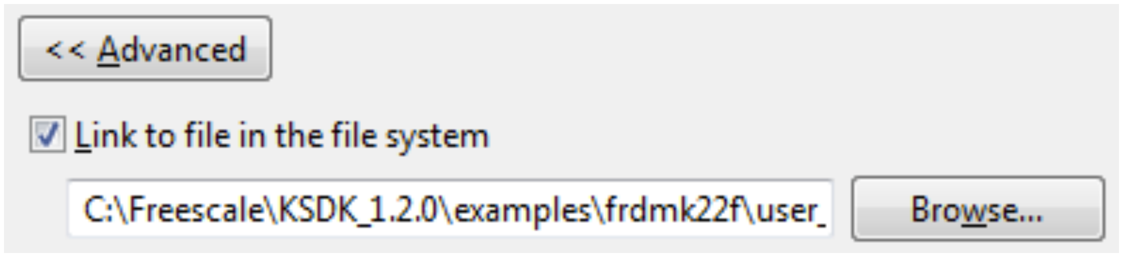


Objective 2: Add Source Files (contd.)

- In the window that appears, click on the Advanced button and then check the Link to file in the file system box

- Browse to the lab2_project folder and select the lab2.c file locate in:

  C:/Freescale/KSDK_1.2.0/examples/frdmk22f/user_apps/lab2_project



☑ Link to file in the file system

C:\Freescale\KSDK_1.2.0\examples\frdmk22f\user_

- The lab2.c source file is now part of your project

Objective 2: Add Source Files (contd.)

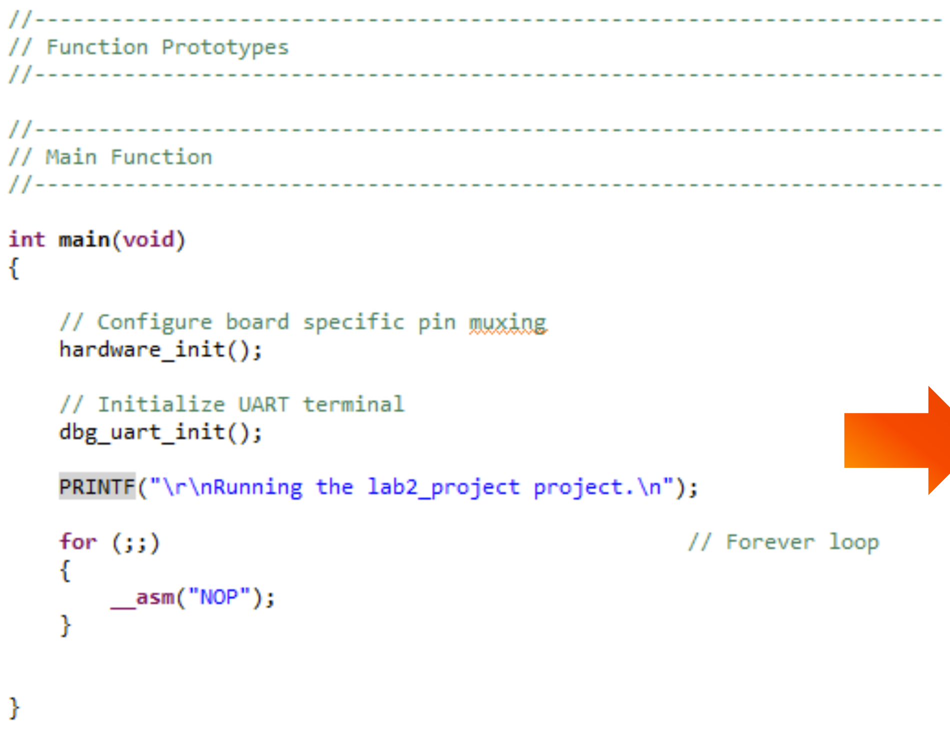
- Lastly, copy the hardware_init.c file from the KSDK Advanced folder on your Desktop into your project in

  C:/Freescale/KSDK_1.2.0/examples/frdmk22f/user_apps/lab2_project

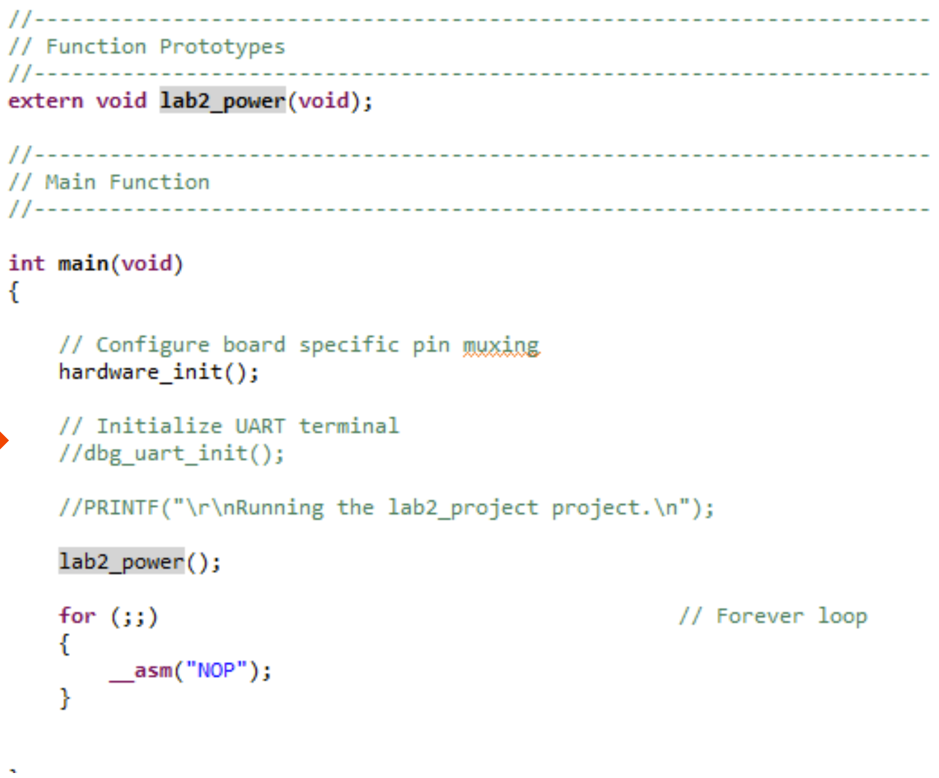
- Overwrite the file that is already present

Objective 2: Use New Source Code

- The source code in lab2.c is intended to act as the main application for the lab
- Modify main() in main.c to look like the following:



```
//-------------------------------------------------------------------------
// Function Prototypes
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
// Main Function
//-------------------------------------------------------------------------

int main(void)
{

    // Configure board specific pin muxing
    hardware_init();

    // Initialize UART terminal
    dbg_uart_init();

    PRINTF("\r\nRunning the lab2_project project.\n");

    for (;;)                                         // Forever loop
    {
        __asm("NOP");
    }


}
```

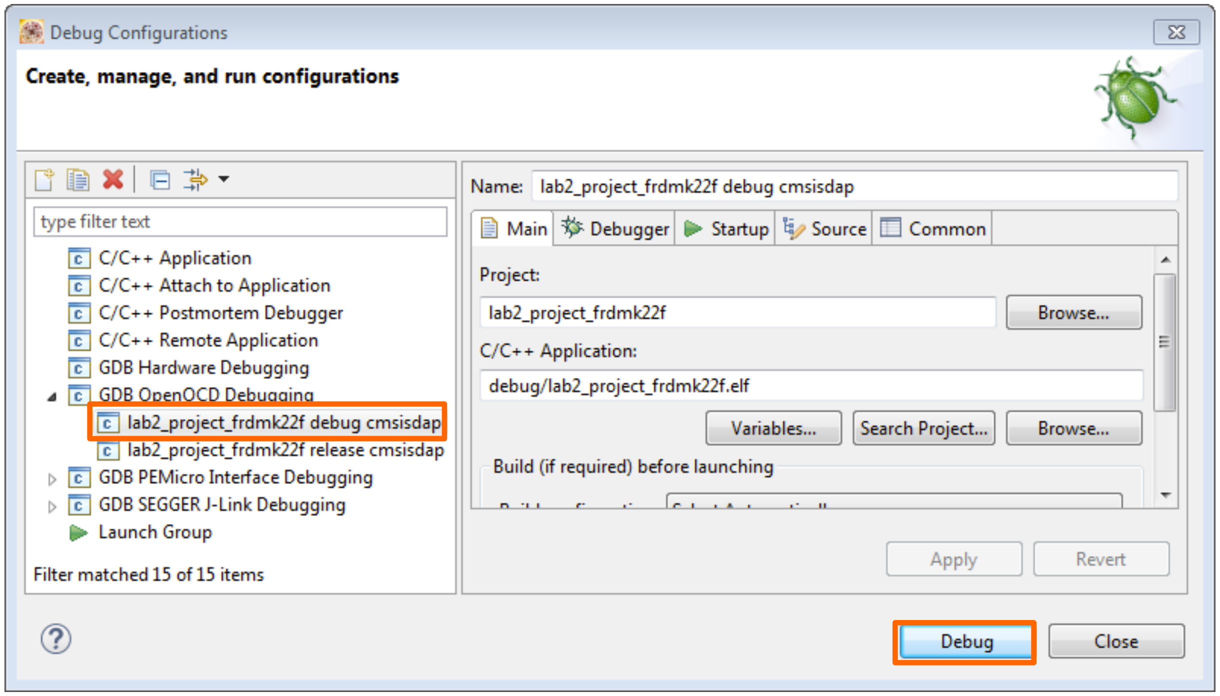


```
//-------------------------------------------------------------------------
// Function Prototypes
//-------------------------------------------------------------------------
extern void lab2_power(void);

//-------------------------------------------------------------------------
// Main Function
//-------------------------------------------------------------------------

int main(void)
{

    // Configure board specific pin muxing
    hardware_init();

    // Initialize UART terminal
    //dbg_uart_init();

    //PRINTF("\r\nRunning the lab2_project project.\n");

    lab2_power();

    for (;;)                                         // Forever loop
    {
        __asm("NOP");
    }
```

Objective 3: Running the Application

- Now that the new source is integrated into your project, re-build the application by highlighting lab2_project and click on the hammer icon

- To load the image onto the FRDM-K22F board, go to Run > Debug Configurations… Select the lab2_project_frdmk22f_debug_cmsisdap target under GDB OpenOCD Debugging. Click on the Debug button to start


Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - lab2_project_frdmk22f debug cmsisdap
  - lab2_project_frdmk22f release cmsisdap
- GDB PEMicro Interface Debugging
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 15 of 15 items

Name: lab2_project_frdmk22f debug cmsisdap

Main | Debugger | Startup | Source | Common

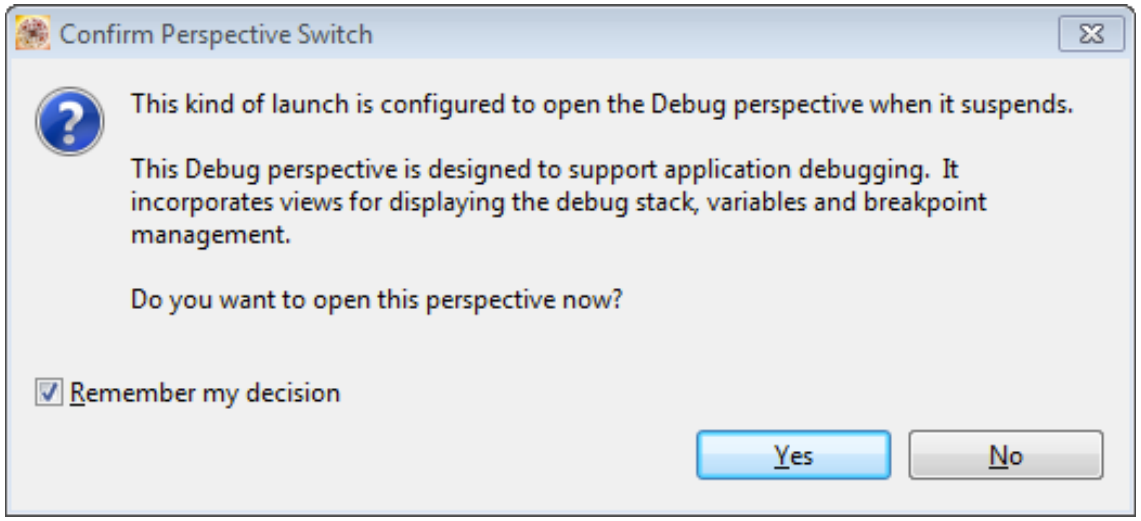
Project: lab2_project_frdmk22f

C/C++ Application: debug/lab2_project_frdmk22f.elf

Build (if required) before launching

Objective 3: Running the Application

- If this is your first time launching a project in this workspace, you will get a dialog window telling you about a perspective switch in the IDE. Check the Remember my decision box and select Yes.


Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

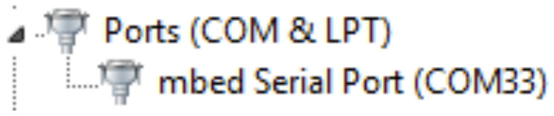
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

[✓] Remember my decision

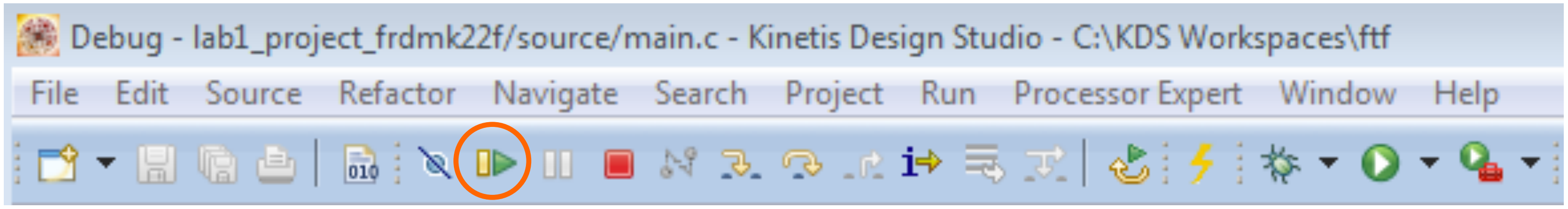
Objective 3: Running the Application

- Before starting the application, you must open up a terminal application to observe the demo's control interface. To determine the COM port, open the Windows Device Manager and look for the mbed Serial Port



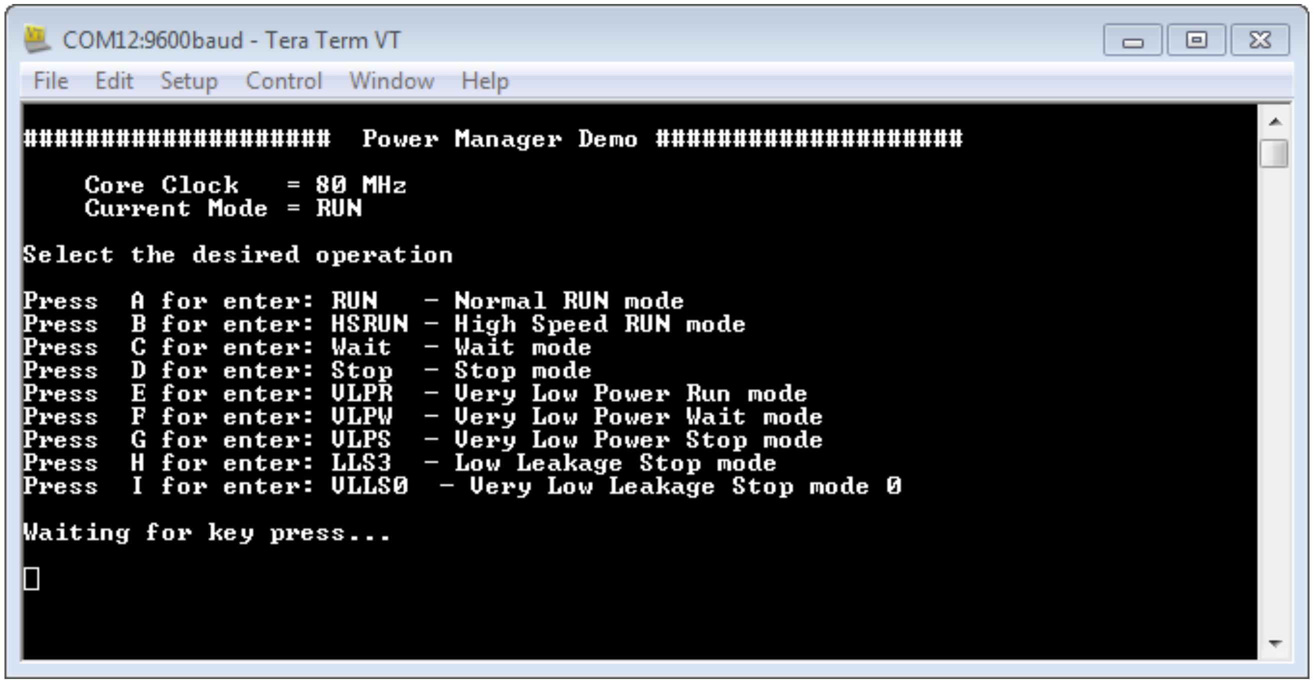
- Open your terminal application of choice on your PC and connect to the COM port with 9600-8-n-1

- In the KDS debugger, click on the Go button (little green arrow) to start the application


Debug - lab1_project_frdmk22f/source/main.c - Kinetis Design Studio - C:\KDS Workspaces\ftf
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Processor Expert
Window
Help

Objective 3: Running the Application

- The demo will output the following text to the screen:


COM12:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
####################  Power Manager Demo ####################

    Core Clock    = 80 MHz
    Current Mode  = RUN

Select the desired operation

Press  A for enter: RUN   - Normal RUN mode
Press  B for enter: HSRUN - High Speed RUN mode
Press  C for enter: Wait  - Wait mode
Press  D for enter: Stop  - Stop mode
Press  E for enter: VLPR  - Very Low Power Run mode
Press  F for enter: VLPW  - Very Low Power Wait mode
Press  G for enter: VLPS  - Very Low Power Stop mode
Press  H for enter: LLS3  - Low Leakage Stop mode
Press  I for enter: VLLS0  - Very Low Leakage Stop mode 0

Waiting for key press...
```

- You can enter various power modes by pressing letters on your keyboard. It is recommended to disconnect the debugger and power cycle the board prior to looking at low power numbers. Use the provided multi meter to measure the current and compare it to the K22F datasheet.

Objective 4: Explore and Understand the Application

- After running the application, take some time to look at the source code

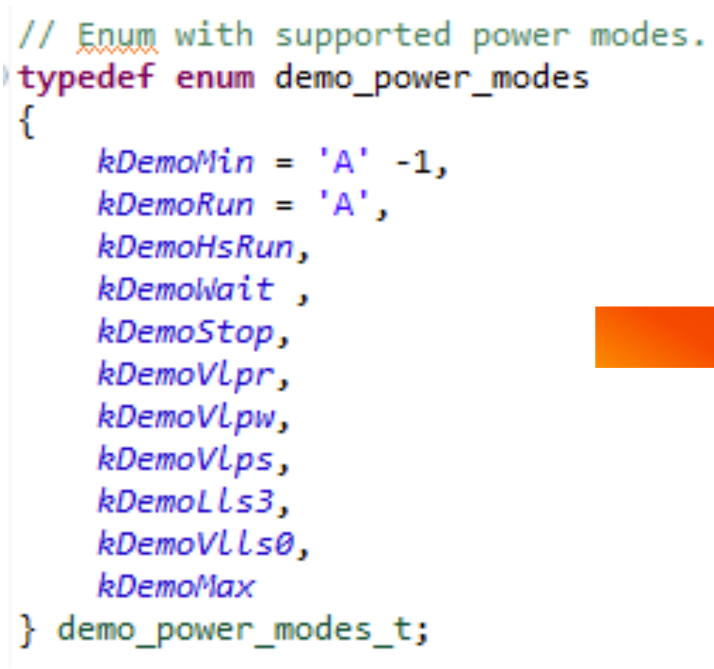
- The lab2.c file has steps listed/numbered, starting in the lab2_power() function

- Go through step-by-step to see how the Clock and Power Managers are configured in the application

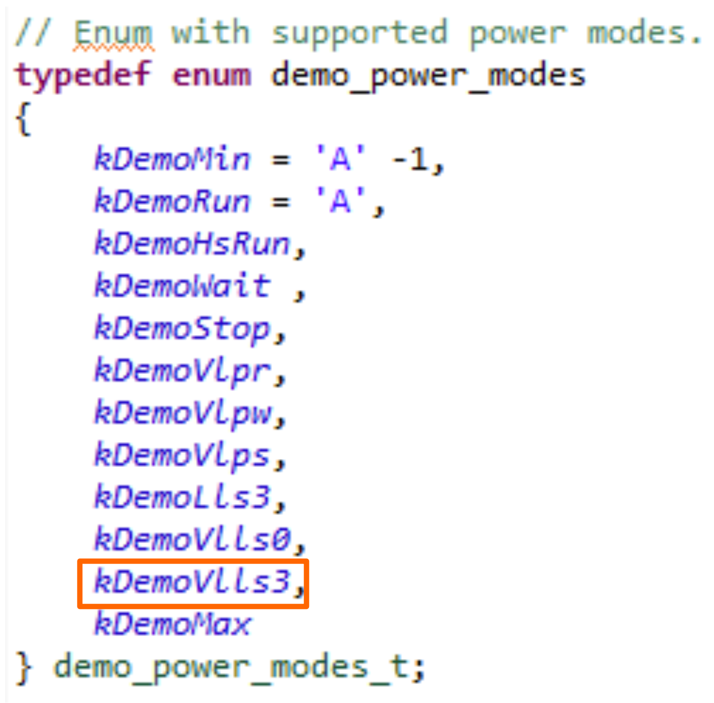
Objective 5: Add Support for VLLS3 Mode

- If you have time, modify the application to support an additional low power mode: VLLS3

1. The first step is to add support to the demo application for the new mode. There is an enum on line 63 where this should be done:



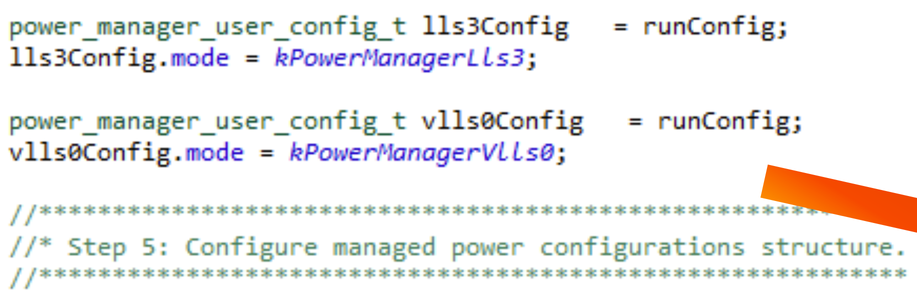
```
// Enum with supported power modes.
typedef enum demo_power_modes
{
    kDemoMin = 'A' -1,
    kDemoRun = 'A',
    kDemoHsRun,
    kDemoWait ,
    kDemoStop,
    kDemoVlpr,
    kDemoVlpw,
    kDemoVlps,
    kDemoLls3,
    kDemoVlls0,
    kDemoMax
} demo_power_modes_t;
```



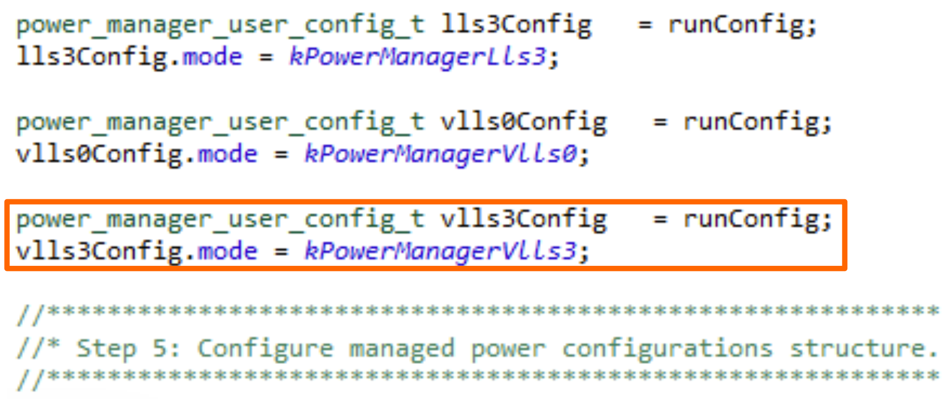
```
// Enum with supported power modes.
typedef enum demo_power_modes
{
    kDemoMin = 'A' -1,
    kDemoRun = 'A',
    kDemoHsRun,
    kDemoWait ,
    kDemoStop,
    kDemoVlpr,
    kDemoVlpw,
    kDemoVlps,
    kDemoLls3,
    kDemoVlls0,
    kDemoVlls3,
    kDemoMax
} demo_power_modes_t;
```

Objective 5: Add Support for VLLS3 Mode

2. Next, add a configuration structure for the new power mode. The easiest way to accomplish this is to copy the structure from another mode and use that as a template. These definitions are found in the section marked as Step 4 in lab2_power()



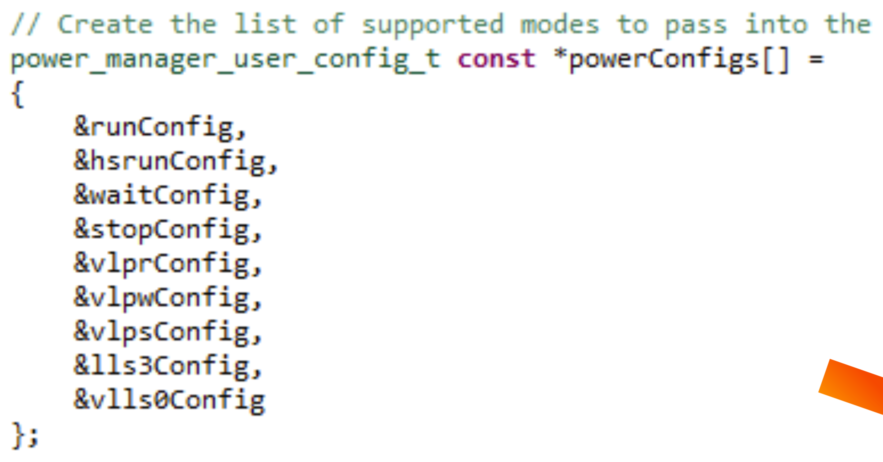
```
power_manager_user_config_t lls3Config   = runConfig;
lls3Config.mode = kPowerManagerLls3;

power_manager_user_config_t vlls0Config   = runConfig;
vlls0Config.mode = kPowerManagerVlls0;

//************************************************************
//* Step 5: Configure managed power configurations structure.
//************************************************************
```



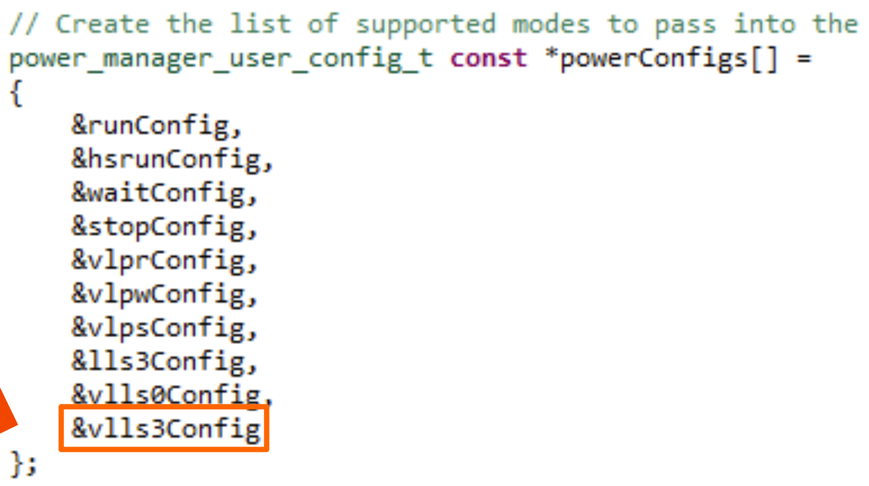
```
power_manager_user_config_t lls3Config   = runConfig;
lls3Config.mode = kPowerManagerLls3;

power_manager_user_config_t vlls0Config   = runConfig;
vlls0Config.mode = kPowerManagerVlls0;

power_manager_user_config_t vlls3Config   = runConfig;
vlls3Config.mode = kPowerManagerVlls3;

//************************************************************
//* Step 5: Configure managed power configurations structure.
//************************************************************
```

Objective 5: Add Support for VLLS3 Mode

3. Now, add the new power configuration to the array of supported power modes, which is part of the section marked as Step 5 in lab2_power()



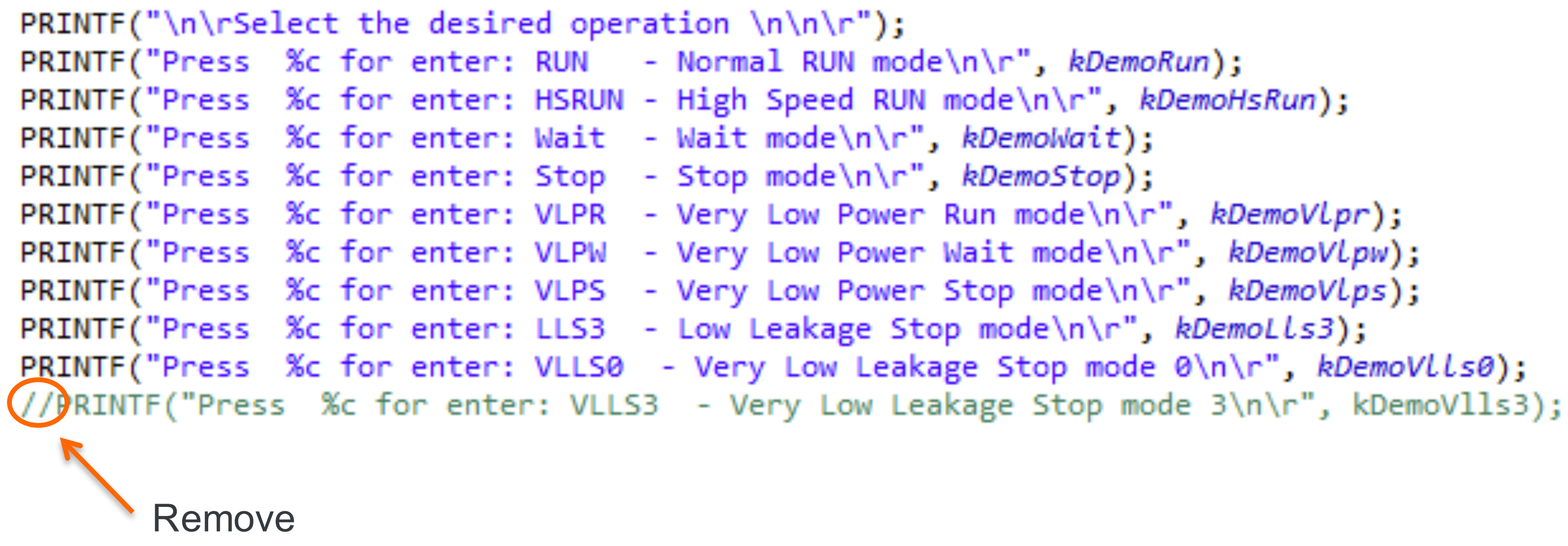
```
// Create the list of supported modes to pass into the
power_manager_user_config_t const *powerConfigs[] =
{
    &runConfig,
    &hsrunConfig,
    &waitConfig,
    &stopConfig,
    &vlprConfig,
    &vlpwConfig,
    &vlpsConfig,
    &lls3Config,
    &vlls0Config
};
```



```
// Create the list of supported modes to pass into the
power_manager_user_config_t const *powerConfigs[] =
{
    &runConfig,
    &hsrunConfig,
    &waitConfig,
    &stopConfig,
    &vlprConfig,
    &vlpwConfig,
    &vlpsConfig,
    &lls3Config,
    &vlls0Config,
    &vlls3Config
};
```

Objective 5: Add Support for VLLS3 Mode

4. Browse down to the while(1) loop in lab2_power() and remove the comment from the line that has a PRINTF statement for VLLS3 mode. It's about line 371, assuming the previous changes have been made correctly:

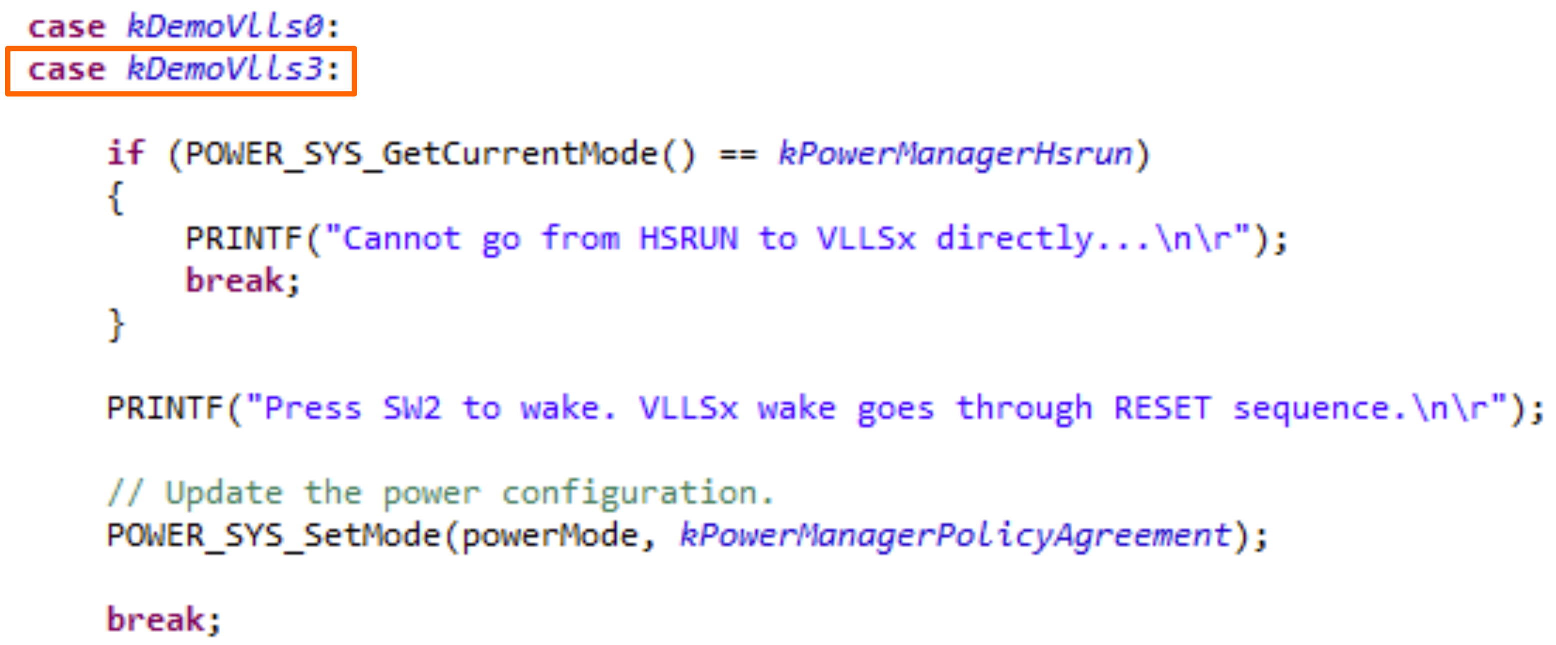


```
PRINTF("\n\rSelect the desired operation \n\n\r");
PRINTF("Press  %c for enter: RUN   - Normal RUN mode\n\r", kDemoRun);
PRINTF("Press  %c for enter: HSRUN - High Speed RUN mode\n\r", kDemoHsRun);
PRINTF("Press  %c for enter: Wait  - Wait mode\n\r", kDemoWait);
PRINTF("Press  %c for enter: Stop  - Stop mode\n\r", kDemoStop);
PRINTF("Press  %c for enter: VLPR  - Very Low Power Run mode\n\r", kDemoVlpr);
PRINTF("Press  %c for enter: VLPW  - Very Low Power Wait mode\n\r", kDemoVlpw);
PRINTF("Press  %c for enter: VLPS  - Very Low Power Stop mode\n\r", kDemoVlps);
PRINTF("Press  %c for enter: LLS3  - Low Leakage Stop mode\n\r", kDemoLls3);
PRINTF("Press  %c for enter: VLLS0  - Very Low Leakage Stop mode 0\n\r", kDemoVlls0);
//PRINTF("Press  %c for enter: VLLS3  - Very Low Leakage Stop mode 3\n\r", kDemoVlls3);
```

#FTF2015

Objective 5: Add Support for VLLS3 Mode

5. Lastly, we need to add a case statement in the while(1) loop in lab2_power() to handle the newly added VLLS3 mode. Since entry into VLLS3 is identical to that for VLLS0, have both share the same handler:



```
case kDemoVlls0:
case kDemoVlls3:

    if (POWER_SYS_GetCurrentMode() == kPowerManagerHsrun)
    {
        PRINTF("Cannot go from HSRUN to VLLSx directly...\n\r");
        break;
    }

    PRINTF("Press SW2 to wake. VLLSx wake goes through RESET sequence.\n\r");

    // Update the power configuration.
    POWER_SYS_SetMode(powerMode, kPowerManagerPolicyAgreement);

    break;
```

Objective 5: Add Support for VLLS3 Mode

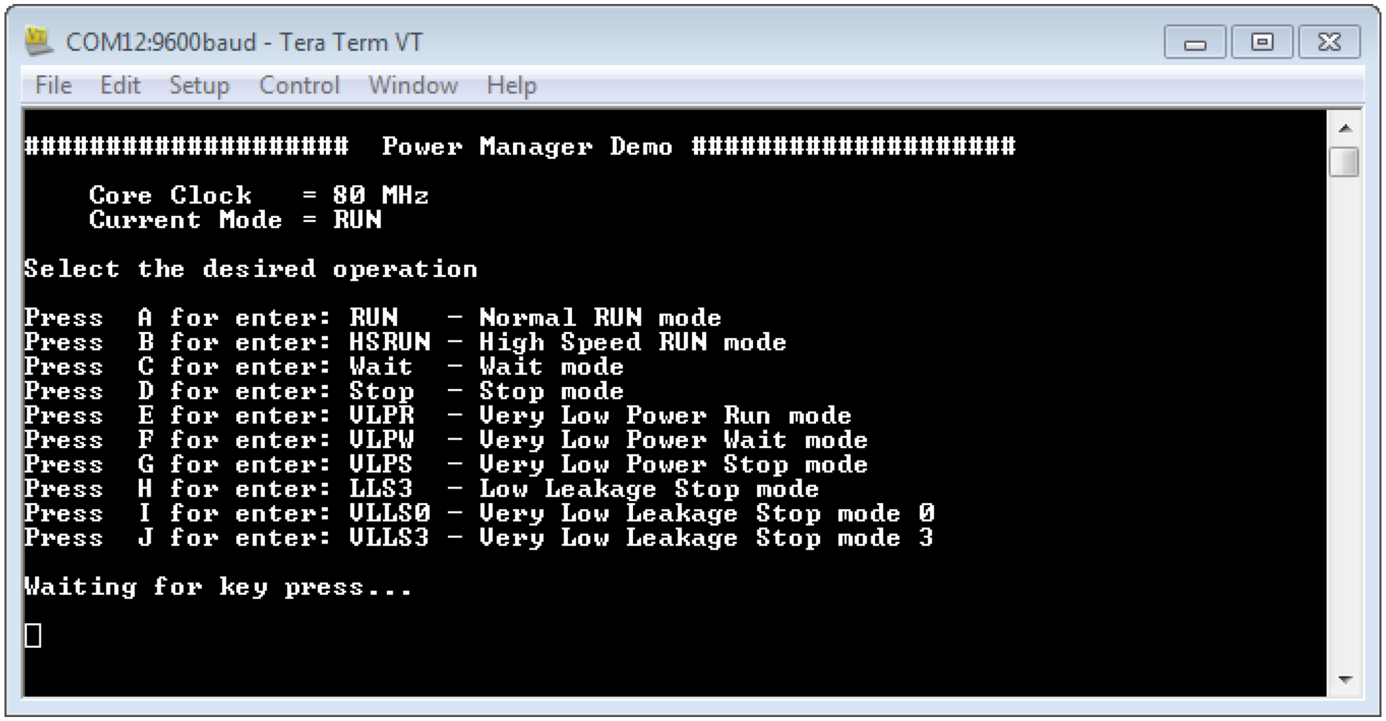
6. Re-build the application

7. Assuming the changes were made correctly, there will be no errors. At this point, download the application by launching the debugger. Reference prior sections is you need help doing this.

8. After loading the application, disconnect the debugger and power cycle the device

Objective 5: Add Support for VLLS3 Mode

- Once the board is plugged in again, launch your terminal application. Press the board reset button to reset and run the application. You will now see a new option for VLLS3. Try it out to see the power consumption in the new mode!


COM12:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
####################  Power Manager Demo ####################

    Core Clock   = 80 MHz
    Current Mode = RUN

Select the desired operation

Press  A for enter: RUN   - Normal RUN mode
Press  B for enter: HSRUN - High Speed RUN mode
Press  C for enter: Wait  - Wait mode
Press  D for enter: Stop  - Stop mode
Press  E for enter: VLPR  - Very Low Power Run mode
Press  F for enter: VLPW  - Very Low Power Wait mode
Press  G for enter: VLPS  - Very Low Power Stop mode
Press  H for enter: LLS3  - Low Leakage Stop mode
Press  I for enter: VLLS0 - Very Low Leakage Stop mode 0
Press  J for enter: VLLS3 - Very Low Leakage Stop mode 3

Waiting for key press...
```