MCXNx4x Power Management Lab

Objectives

In this lab, you will learn

→ about the Power Management features of the MCX Nx4x MCU

→ how to measure current on the FRDM-MCXN947 board

→ how to measure current with MCU-Link Pro and MCUXpresso IDE

→ about the Power Management component in the MCUXpresso SDK

Pre-Requisites

This guide uses the MCUXpresso IDE to take advantage of the integrated Energy Measurement tool for the MCU-Link Pro. However, other IDEs are supported with this application, including VS Code and IAR. See the application Readme to import into those IDEs.

a. Install MCUXpresso IDE v11.8.1 or later
b. Install an MCUXpresso SDK package supporting the MCX N947 device into the IDE
c. MCU-LINK_installer_3.108 or later is required for MCUXpresso IDE.
   If the firmware in the MCU-Link is older, it needs to be updated. This is needed for the onboard MCU-Link, or the standalone probes MCU-Link or MCU-Link Pro.
d. Terminal Program, like PuTTY or Tera Term, or use Terminal integrated into MCUXpresso IDE

Hardware Requirements

This guide is written for the FRDM-MCXN947 board. But the application also runs on the MCX-N9XX-EVK board.

- FRDM-MCXN947 board with USB Type-C cable
- MCU-Link Pro (for energy measurements) with micro-USB cable

Preparing the Hardware

i. Configure the MCU-Link Pro current measurement range:
   a. Change the 3 jumpers J16/17/18 to short pins 2-3
   b. This changes the range from 350 mA to 50 mA
ii. On the FRDM-MCXN947 board open the J24 jumper for the MCU current
iii. Connect MCU-Link Pro to FRDM-MCXN947

Signal	MCU-Link Pro	FRDM	Color	Wire Type
I_{DD_IN}	J9 current in	J24-1	Yellow	Female-Female
I_{DD_OUT}	J9 current out	J24-2	Red	Female-Female
V_{SS}	J9 GND	J10-4	Black	Female-Female

Table 1: MCU-Link – FRDM board connections

The following picture shows the connection between the FRDM and MCU-Link Pro boards:

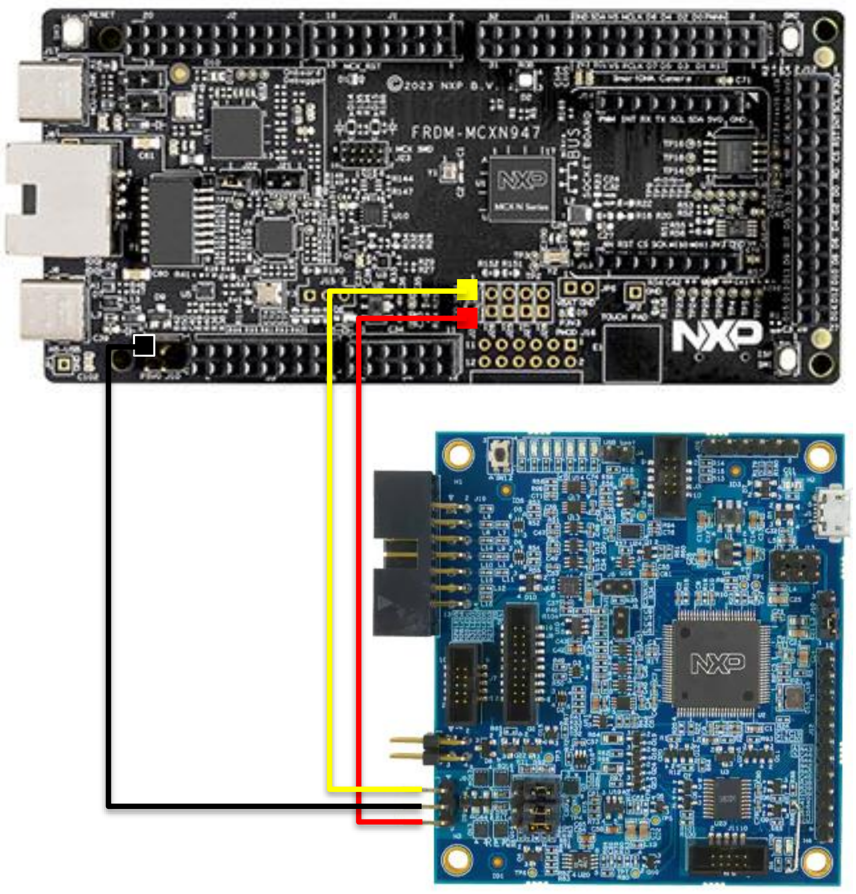


Figure 1 MCU-Link – FRDM board connections

iv. Connect MCU-Link Pro to PC via USB – using **J1**

v. Connect FRDM-MCXN947 to PC via USB – using **J17**

vi. Now all is ready for the power manager labs

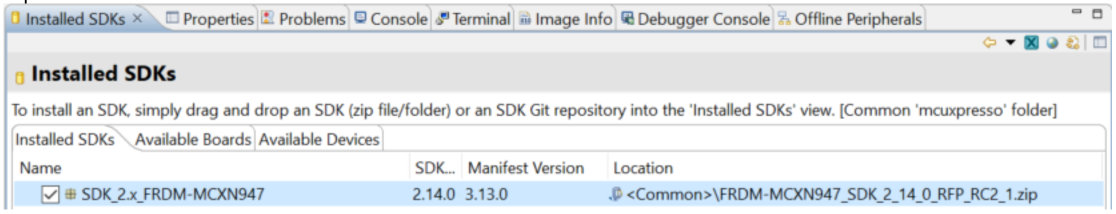
Labs

1. Lab 1: Run Power Manager Example on FRDM

This lab will focus on importing the project into MCUXpresso IDE, building the power manager example, and finally downloading it to the FRDM board. You will see the application menu print to the terminal.

1.1. Open MCUXpresso IDE and select your workspace.

1.2. If you have not already installed the **SDK_2.x_FRDM-MCXN947**, install it by dragging the downloaded zip file of the SDK to the **Installed SDKs** tab.



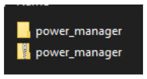
Installed SDKs

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

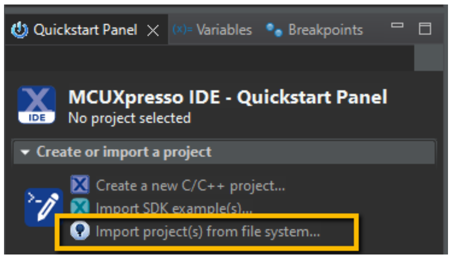
Installed SDKs | Available Boards | Available Devices

Name	SDK...	Manifest Version	Location
☑ SDK_2.x_FRDM-MCXN947	2.14.0	3.13.0	<Common>\FRDM-MCXN947_SDK_2_14_0_RFP_RC2_1.zip

1.3. Locate the previously downloaded **power_manger.zip,** then unzip it.

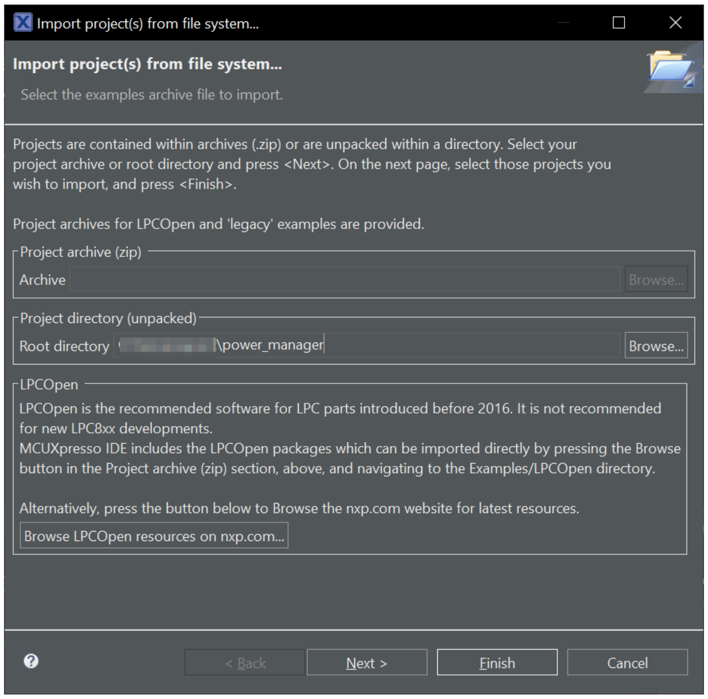


1.4. From the MCUXpresso IDE Quickstart Panel select **Import project(s) from file system…**



Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import project(s) from file system...

1.5. A new window will pop up, look for the Project directory (unpacked) Root Directory section, then click **Browse...**
1.6. Look for the extracted **power_manager** folder then click **Open**.
1.7. Click **Next>**


Import project(s) from file system...

Import project(s) from file system...

Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archives for LPCOpen and 'legacy' examples are provided.

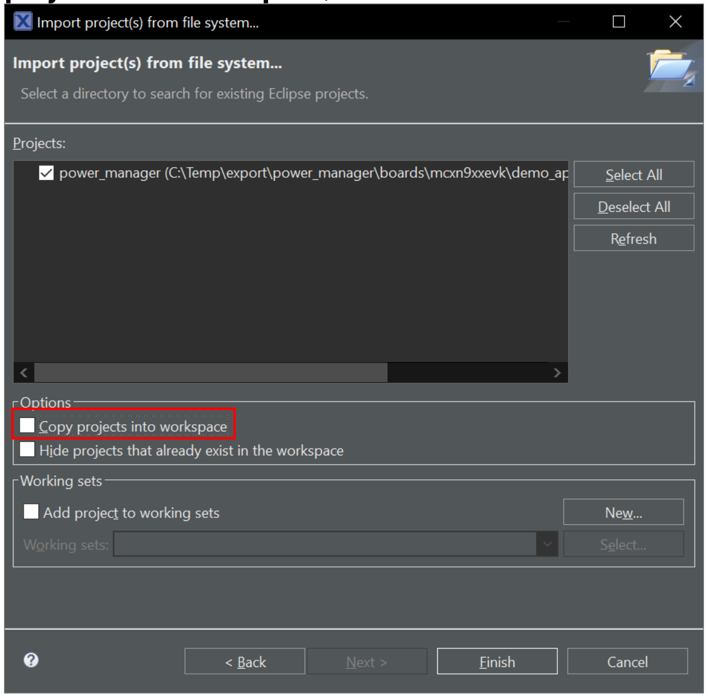
Project archive (zip)
Archive — Browse...

Project directory (unpacked)
Root directory [redacted]\power_manager — Browse...

LPCOpen
LPCOpen is the recommended software for LPC parts introduced before 2016. It is not recommended for new LPC8xx developments.
MCUXpresso IDE includes the LPCOpen packages which can be imported directly by pressing the Browse button in the Project archive (zip) section, above, and navigating to the Examples/LPCOpen directory.

Alternatively, press the button below to Browse the nxp.com website for latest resources.

1.8. In the Import project(s) from file system window, select the **power_manager** demo, **uncheck Copy project into a workspace**, then click Finish


Import project(s) from file system...

Select a directory to search for existing Eclipse projects.

Projects:

☑ power_manager (C:\Temp\export\power_manager\boards\mcxn9xxevk\demo_ap

Options

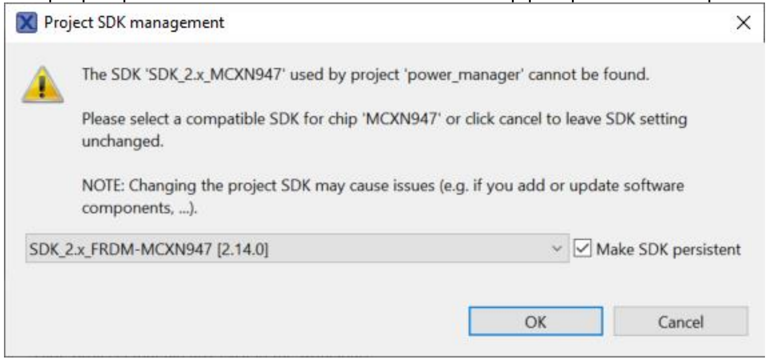
☐ Copy projects into workspace

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

1.9. If a pop-up like below is seen, select the appropriate SDK package, and click OK.


Project SDK management

The SDK 'SDK_2.x_MCXN947' used by project 'power_manager' cannot be found.

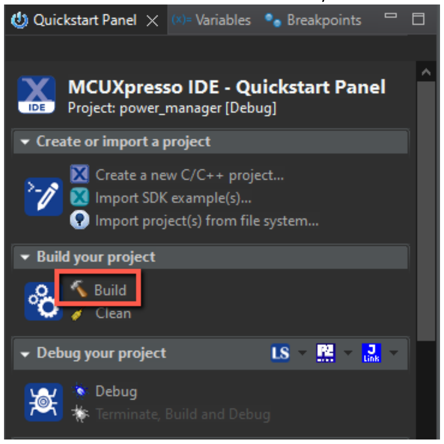
Please select a compatible SDK for chip 'MCXN947' or click cancel to leave SDK setting unchanged.

NOTE: Changing the project SDK may cause issues (e.g. if you add or update software components, ...).

SDK_2.x_FRDM-MCXN947 [2.14.0]   ☑ Make SDK persistent

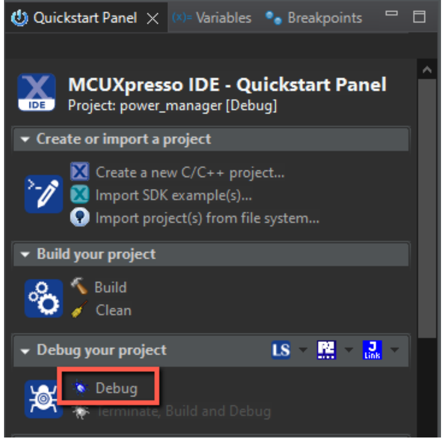
1.10. Make sure the power_manager project is selected in the Project Explorer window.

1.11. Inside the Quickstart Panel, click the **Build** button.

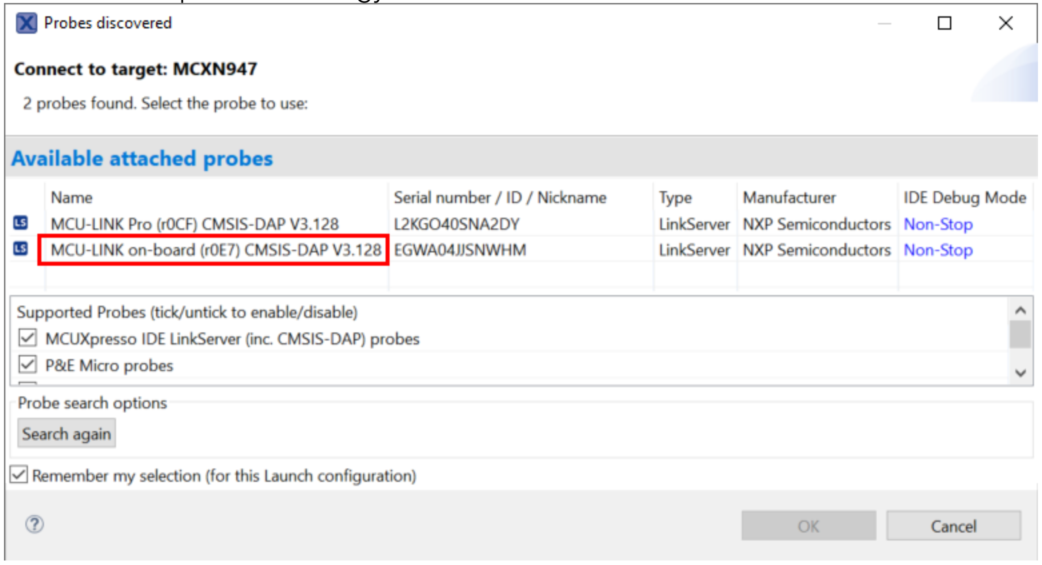

Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: power_manager [Debug]
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug

Download and run the power manager application.

1.12. Make sure that power_manager project is highlighted, then from the Quickstart Panel select **Debug**.


Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: power_manager [Debug]
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug

1.13. Note that we have two MCU-Link probes connected, and the probe needs to be selected for debug. Select the "**on-board**" MCU-Link probe. We are using the on-board probe for debugging, and the other MCU-Link Pro probe for energy measurement. Then click **OK**.



Probes discovered

Connect to target: MCXN947

2 probes found. Select the probe to use:

Available attached probes

Name	Serial number / ID / Nickname	Type	Manufacturer	IDE Debug Mode
MCU-LINK Pro (r0CF) CMSIS-DAP V3.128	L2KGO40SNA2DY	LinkServer	NXP Semiconductors	Non-Stop
MCU-LINK on-board (r0E7) CMSIS-DAP V3.128	EGWA04JJSNWHM	LinkServer	NXP Semiconductors	Non-Stop

Supported Probes (tick/untick to enable/disable)
- ☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- ☑ P&E Micro probes

Probe search options

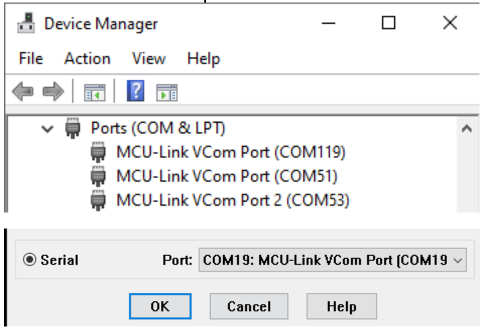
☑ Remember my selection (for this Launch configuration)

1.14. Wait for the application to download and stop at the beginning of main() function. Click "OK" on any windows that pop up during this process.

1.15. Open the terminal like Tera Term, or use the integrated terminal in the IDE.

1.16. Note that there are multiple COM ports available when both MCU-Links are connected to the PC. You can disconnect the MCU-Link Pro to find the COM port number associated with the on-board MCU-Link on the FRDM board.

Select the serial port for the MCU-LINK (your COM number might be different):



Device Manager

Ports (COM & LPT)
- MCU-Link VCom Port (COM119)
- MCU-Link VCom Port (COM51)
- MCU-Link VCom Port 2 (COM53)

● Serial Port: COM19: MCU-Link VCom Port (COM19

1.17. Press Resume (F8) to start the application. You should see:

POR Main Menu

```
#######    MCX Nx4x Power Manager component demo    #######

---------------------- Normal Boot ----------------------

Main Menu, press a key below:

      p: Power Modes Menu
      a: Analog Peripherals Menu
      c: Clocks Menu
      s: System SRAM Menu
      e: Enter Selected Mode
```

2. Lab 2: Measure Current with MCUXpresso IDE

The Energy Measurement view enables current measurements within MCUXpresso IDE. This lab will show how to collect current measurements for the FRDM using MCU-Link Pro.

NOTE: The energy Measurement functionality does not require an active debug session to capture data, but it can be linked to one if such a session exists.

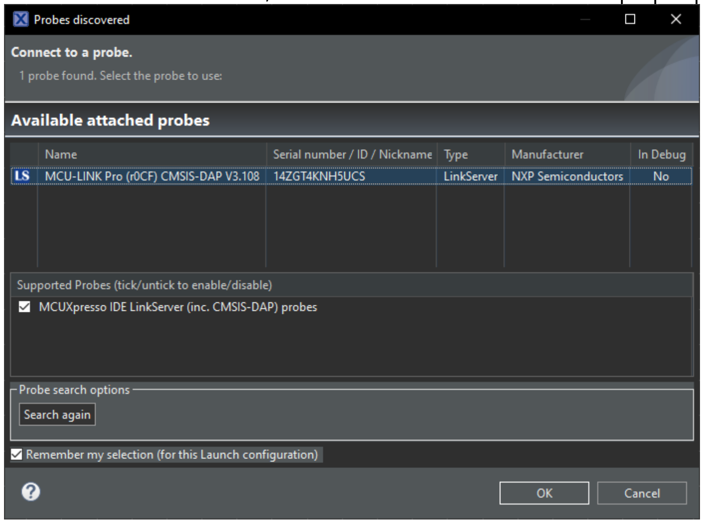
2.1. Make sure that your debug session is closed, by clicking over the Terminate button

2.2. Cycle power to the target board. When measuring current in the low-power modes, we should cycle power after the debug connection. Otherwise, the debug circuit in the MCU can remain powered.

2.3. In MCUXpresso IDE, open the Energy Measurement view by using the menus: **Analysis → Energy Measurement**

2.4. If the external MCU-Link Pro is not already connected, please connect it to the FRDM board and the PC.

2.5.Start an analysis session ~~by~~ using the **Identify probe for out-of-debug analysis** button. Select the attached MCU-Link Pro probe then click **OK**. Note, the on-board probe does not support the energy measurement feature, so it will not be listed in this pop-up. Select the MCU-Link Pro.



Probes discovered

Connect to a probe.

1 probe found. Select the probe to use:

Available attached probes

	Name	Serial number / ID / Nickname	Type	Manufacturer	In Debug
LS	MCU-LINK Pro (r0CF) CMSIS-DAP V3.108	14ZGT4KNH5UCS	LinkServer	NXP Semiconductors	No

Supported Probes (tick/untick to enable/disable)

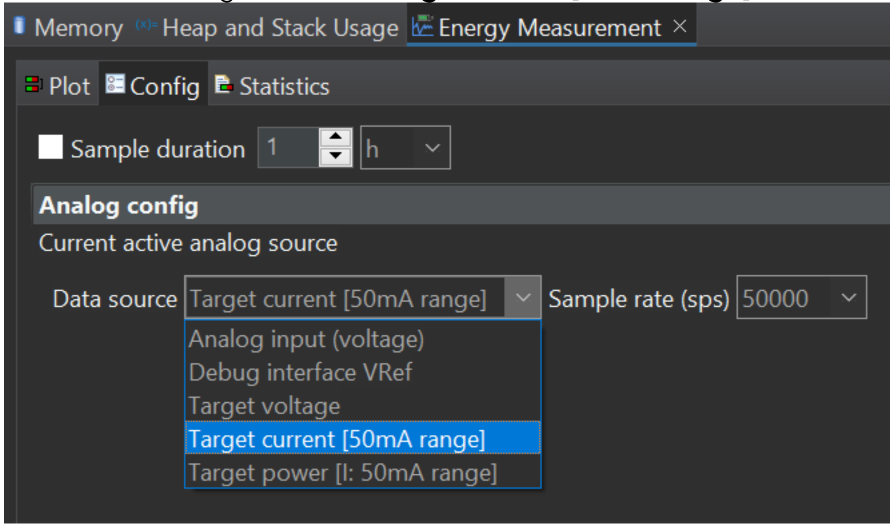
☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes

Probe search options

Search again

☑ Remember my selection (for this Launch configuration)

2.6. From the Config Tab, select **Target current [50mA range]** Data source in the Analog config section



Sample duration 1 h

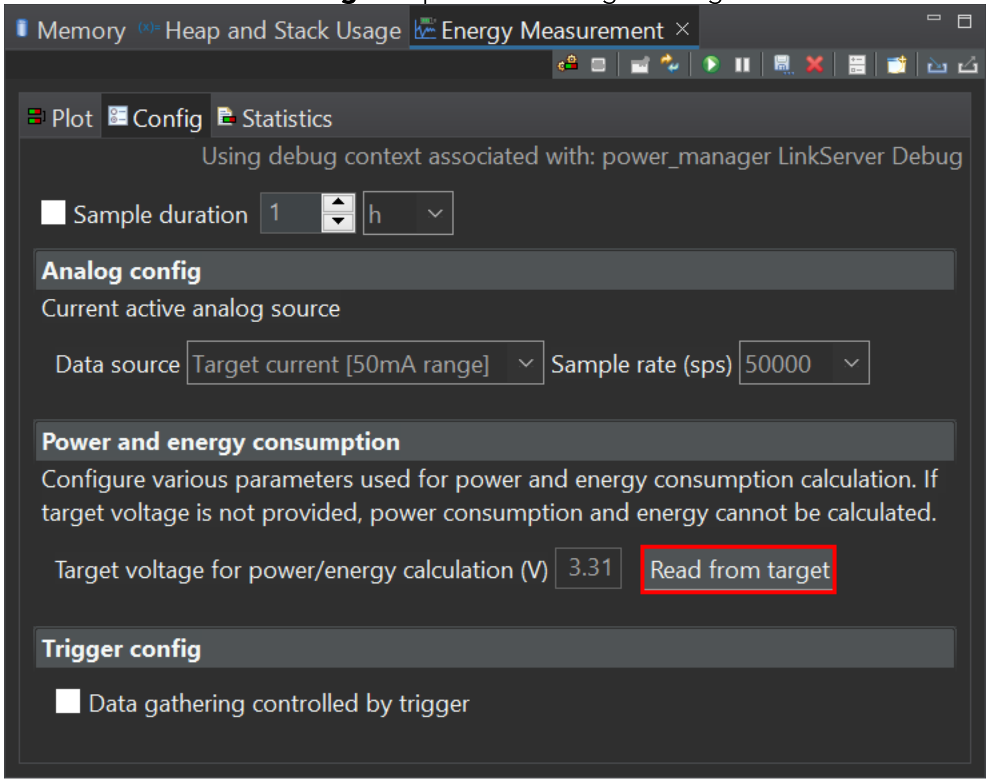
Analog config

Current active analog source

Data source: Target current [50mA range]   Sample rate (sps): 50000

- Analog input (voltage)
- Debug interface VRef
- Target voltage
- Target current [50mA range]
- Target power [I: 50mA range]

2.7. Click over **Read from target** to provide the target voltage



Memory | Heap and Stack Usage | Energy Measurement

Plot | Config | Statistics

Using debug context associated with: power_manager LinkServer Debug

☐ Sample duration 1 h

Analog config

Current active analog source

Data source Target current [50mA range] Sample rate (sps) 50000

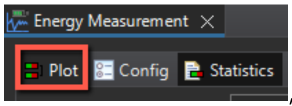
Power and energy consumption

Configure various parameters used for power and energy consumption calculation. If target voltage is not provided, power consumption and energy cannot be calculated.

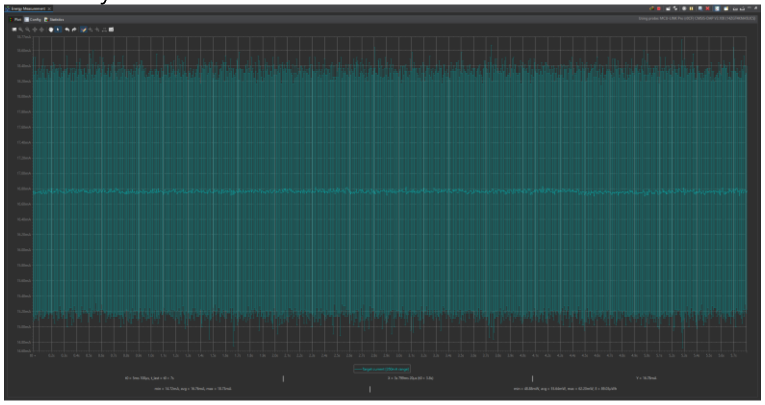
Target voltage for power/energy calculation (V) 3.31 Read from target

Trigger config

☐ Data gathering controlled by trigger

2.8. Go to the Plot tab  , then press **Resume data display** button ▶.

2.9. Now you should be able to measure current in the Plot tab



3. Lab 3: Static Power Modes

This lab focuses on entering different MCU low-power modes from the menu, and measuring the baseline current in each mode. You will want to record these baseline currents, to reference in later labs.

3.1. Reset or power cycle the board to start fresh.

Follow

3.2. Lab 2: Measure Current with MCUXpresso IDE to **start recording the current** in the IDE with the MCU-Link Pro

3.3. Using the terminal, from the main menu, press the **'P' key** to load the Power menu

```
Power Menu
---------------------------------------------
0. Choose a power mode
e. Enter selected power mode
w. Toggle CORE_WAKE domain constraint

b. Back to main menu
```

3.4. From the *Power Menu*, press the **'0' key (zero)** to choose the power mode to enter

```
Power Mode
---------------------------------------------
Select a Power Mode:
        0. Active
        1. Sleep
        2. Deep Sleep
        3. Power Down
        4. Deep Power Down
```

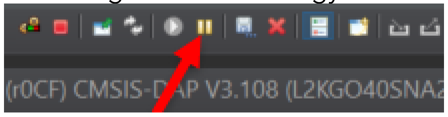
3.5. Press the **'1' key** to choose Sleep Mode.

```
Power Menu
---------------------------------------------
Power Menu, press a key below:
        Current Selection = Sleep

0. Choose a power mode
e. Enter selected power mode
w. Toggle CORE_WAKE domain constraint

b. Back to main menu
```

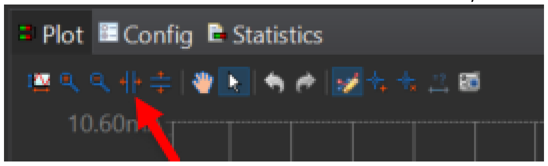
3.6. From the Power Menu, press the **'E' key** to enter the selected power mode.

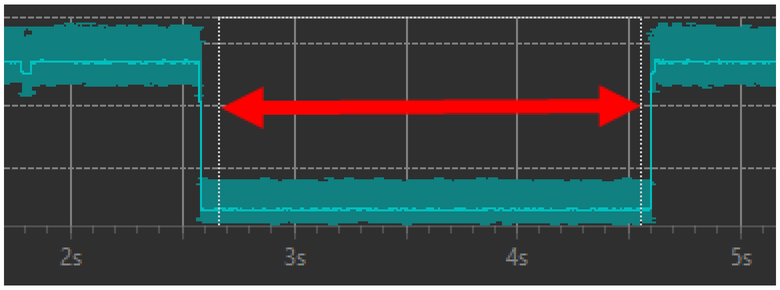
```
Power Mode Entry
---------------------------------------------
Calculated power mode is Sleep      , because a power-mode constraint limits to this mode.
```

3.7. The example uses the LPTMR to wake after 2 seconds. After the MCU wakes pause the measurement recording in the IDE Energy Measurement viewing.

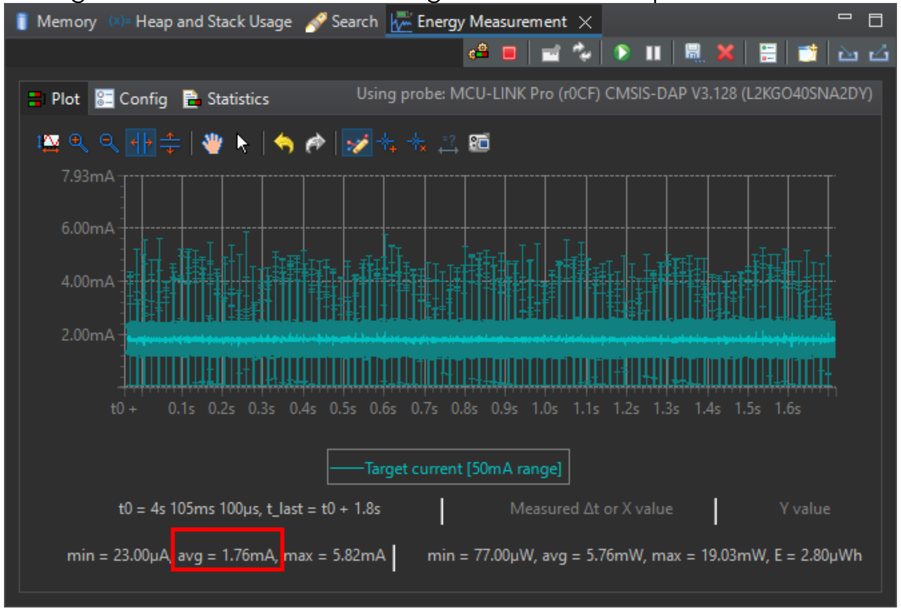


3.8. Click the **Horizontal Zoom button**, and zoom in to measure the current during Sleep mode.





3.9. The lower-left corner of the Energy Measurement view shows the current measurements, including the average current. Record this average current for Sleep Mode.

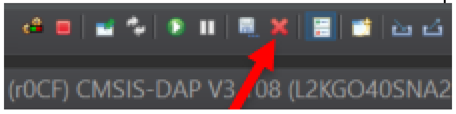


Using probe: MCU-LINK Pro (r0CF) CMSIS-DAP V3.128 (L2KGO40SNA2DY)

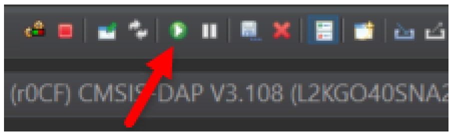
t0 = 4s 105ms 100µs, t_last = t0 + 1.8s | Measured Δt or X value | Y value

min = 23.00µA, avg = 1.76mA, max = 5.82mA | min = 77.00µW, avg = 5.76mW, max = 19.03mW, E = 2.80µWh

3.10. Click the **Clear Data button** to prepare for another measurement



3.11. Click the **Resume Data button** to start recording current again.



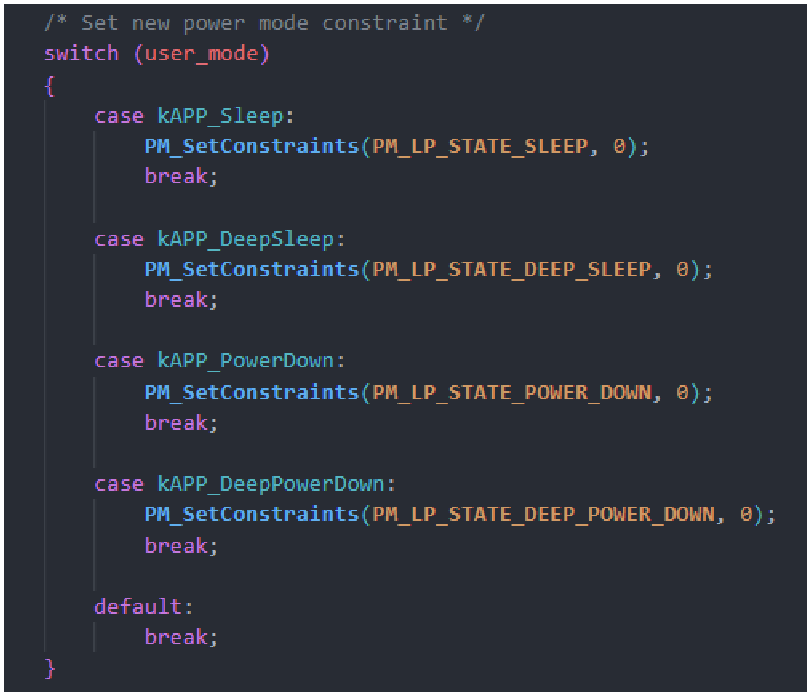
3.12. Repeat steps 3.3 to 3.11 to measure and record the current for each of these power modes:

- Deep Sleep
- Power Down
- Deep Power Down

Power Mode	Average Current
Sleep	
Deep Sleep	
Power Down	
Deep Power Down	

Software walk-through

The power menu leverages the Power Manager (**PM**) component included in the MCUXpresso SDK. When you use the menu to choose the power mode, the application sets a PM power-mode constraint, see below.



```c
/* Set new power mode constraint */
switch (user_mode)
{
    case kAPP_Sleep:
        PM_SetConstraints(PM_LP_STATE_SLEEP, 0);
        break;

    case kAPP_DeepSleep:
        PM_SetConstraints(PM_LP_STATE_DEEP_SLEEP, 0);
        break;

    case kAPP_PowerDown:
        PM_SetConstraints(PM_LP_STATE_POWER_DOWN, 0);
        break;

    case kAPP_DeepPowerDown:
        PM_SetConstraints(PM_LP_STATE_DEEP_POWER_DOWN, 0);
        break;

    default:
        break;
}
```

Figure 2 power-mode constraint

The later step with the 'E' key to enter the power mode calls `PM_EnterLowPower(DURATION_SECONDS(g_pmDuration));`

The PM component then enters the deepest power mode allowed given the current constraints. The PM component can also return details to the application when calling `PM_findDeepestState(duration, &results);`

The results include

- → `results.deepestState` = which power mode is the deepest allowed, and will be entered
- → `results.reason` = the reason this power mode is the deepest allowed
- → `results.resc_num` = which constraint prevented a deeper power mode, if the reason was a constraint

Power Mode Entry
Calculated power mode is Sleep   , because a power-mode constraint limits to this mode.

SPOILER ALERT: the next page has the expected current measurements! Complete your measurements before viewing the next page.

Low-Power Modes Currents

Power Mode	Average Current
Sleep	1.76 mA
Deep Sleep	133 uA
Power Down	2.82 uA
Deep Power Down	1.38 uA

Table 2 Baseline average current in low-power modes

4. Lab 4: Active Mode, Clock Frequencies

This lab focuses on configuring different active voltage/clock settings. You will want to record these baseline currents, to reference in later labs

4.1. Reset or power cycle the board to start fresh.

4.2. Follow

4.3. Lab 2: Measure Current with MCUXpresso IDE to start recording the current in the IDE with the MCU-Link Pro

4.4. Using the terminal, from the main menu, press the **'c'** key to load the Clocks menu

```
Clocks Menu
Clocks Menu, press a key below:

        0. Main clock
        b. Back to main menu
```

4.5. From the Clocks Menu, press the **'0'** key to choose the CORE0 clock frequency

```
CORE0 Clock Source
Select a CORE0 Clock Source:
        0. FRO LF  @ 12 MHz
        1. FRO HF  @ 48 MHz
        2. PLL     @100 MHz
        3. FRO HF  @144 MHz
        4. PLL     @150 MHz
```

4.6. Press the **'4'** key to choose PLL running at 150MHz.

```
Clocks Menu
Clocks Menu, press a key below:

        0. Main clock
        b. Back to main menu
```

4.7. From the Clocks Menu, press the **‘b’** key to go Back to the main menu

Main Menu

```
Main Menu, press a key below:

    p: Power Modes Menu
    a: Analog Peripherals Menu
    c: Clocks Menu
    s: System SRAM Menu
    e: Enter Selected Mode
```

4.8. From the Main Menu, press the **‘p’** key to select Active Mode, double check that the Current Selection = **Active**

Power Menu

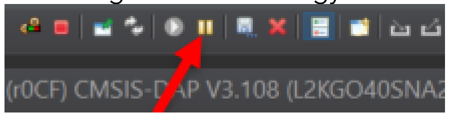
```
Power Menu, press a key below:
    Current Selection = Active

    0. Choose a power mode
    e. Enter selected power mode
    w. Toggle CORE_WAKE domain constraint, currently disabled
        Only applies in Power Down mode. If set, WAKE domain will
        be in Deep Sleep mode. Otherwise, WAKE domain will be in
        Power Down mode


    b. Back to main menu
```

4.9. From the Power Menu, press the **‘e’** key to enter the selected power mode.

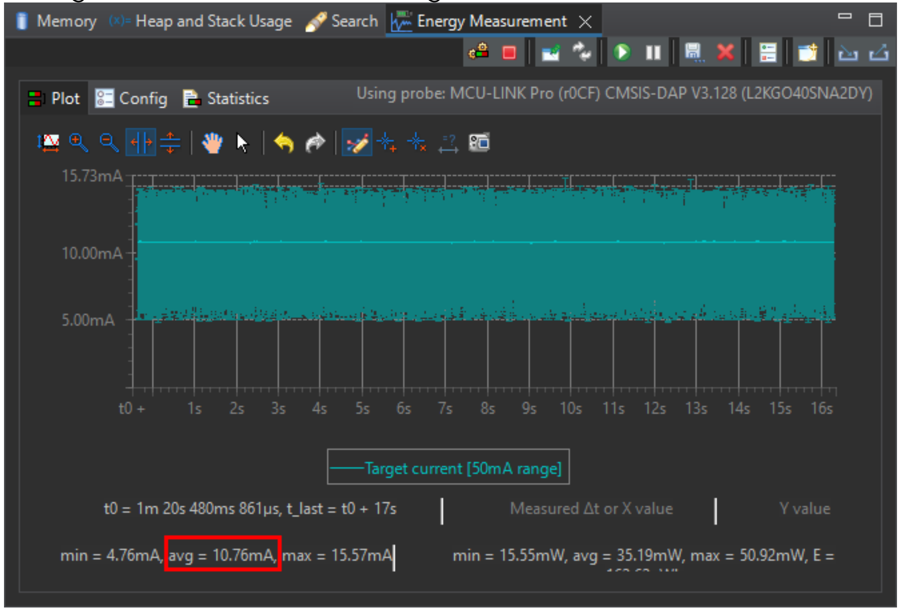
Active Mode Entry

```
Main Clock Source is PLL    @150 MHz
  Core Clock = 150000000Hz
  VDD_CORE = 1.2V
```

4.10. After initiating the selected clock source, the example code runs in a while(1). Pause the measurement recording in the IDE Energy Measurement viewing.



4.11. The lower-left corner of the Energy Measurement view shows the current measurements, including the average current. Record this average current for PLL @ 150MHz.



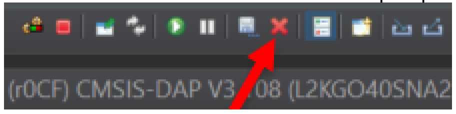
Using probe: MCU-LINK Pro (r0CF) CMSIS-DAP V3.128 (L2KGO40SNA2DY)

t0 = 1m 20s 480ms 861µs, t_last = t0 + 17s

min = 4.76mA, avg = 10.76mA, max = 15.57mA

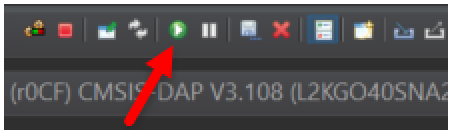
min = 15.55mW, avg = 35.19mW, max = 50.92mW, E =

4.12. Click the **Clear Data** button to prepare for another measurement



(r0CF) CMSIS-DAP V3.108 (L2KGO40SNA2

4.13. Click the **Resume Data** button to start recording the current again.



(r0CF) CMSIS-DAP V3.108 (L2KGO40SNA2

4.14. Repeat steps 4.1 to 4.12 to measure and record the current for these CORE0 clock sources:

CORE 0 Clock	I_{DD_CORE}	V_{DD_CORE}
FRO 12MHz		
FRO 48MHz		
PLL 100MHz		
FRO 148MHz		
PLL 150MHz		

Table 3 Baseline average current active mode

SPOILER ALERT: the next page has the expected current measurements! Complete your measurements before viewing the next page.

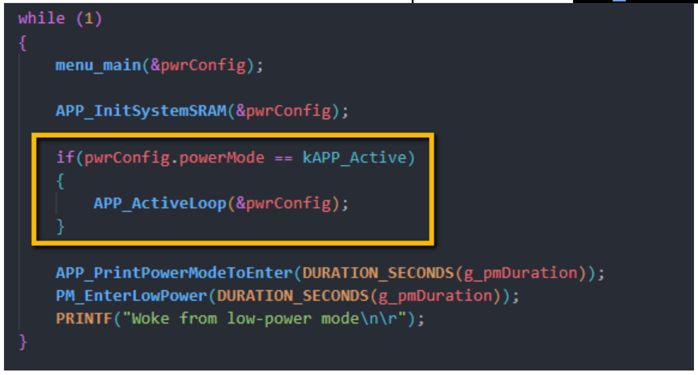
Active Mode Currents

CORE0 Clock	Average Current	V_{DD_CORE}
FRO 12MHz	1.53 mA	1.0V
FRO 48MHz	3.07 mA	1.0V
PLL 100MHz	6.95 mA	1.1V
FRO 148MHz	5.94 mA	1.2V
PLL 150MHz	10.76 mA	1.2V

Table 4 Baseline average current active mode results

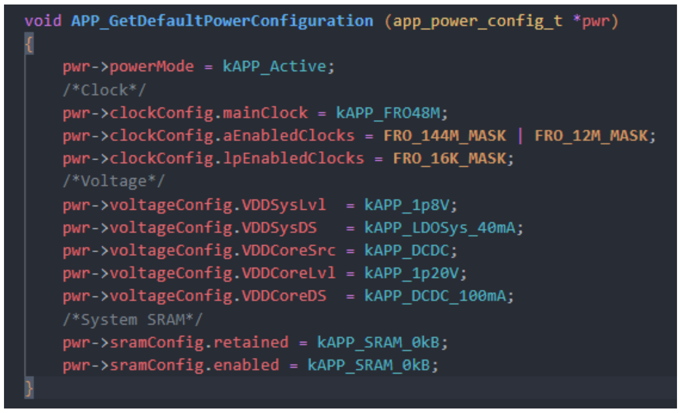
Active Mode software walk-through

When Active Mode is the chosen power mode `APP_ActiveLoop(&pwrConfig);` function gets called.



```
while (1)
{
    menu_main(&pwrConfig);

    APP_InitSystemSRAM(&pwrConfig);

    if(pwrConfig.powerMode == kAPP_Active)
    {
        APP_ActiveLoop(&pwrConfig);
    }

    APP_PrintPowerModeToEnter(DURATION_SECONDS(g_pmDuration));
    PM_EnterLowPower(DURATION_SECONDS(g_pmDuration));
    PRINTF("Woke from low-power mode\n\r");
}
```

Before showing what `APP_ActiveLoop` does, we need to explain what information the `pwrConfig` structure contains. This structure has most of the active mode user configurations like: power mode, clock, voltage, and system SRAM
`pwrConfig` gets its default values in the initialization code `APP_GetDefaultPowerConfiguration (&pwrConfig);`

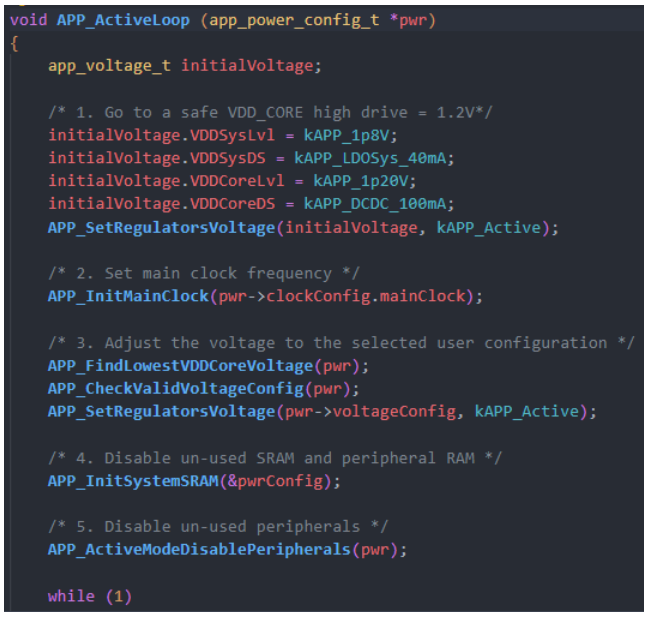


```
void APP_GetDefaultPowerConfiguration (app_power_config_t *pwr)
{
    pwr->powerMode = kAPP_Active;
    /*Clock*/
    pwr->clockConfig.mainClock = kAPP_FRO48M;
    pwr->clockConfig.aEnabledClocks = FRO_144M_MASK | FRO_12M_MASK;
    pwr->clockConfig.lpEnabledClocks = FRO_16K_MASK;
    /*Voltage*/
    pwr->voltageConfig.VDDSysLvl  = kAPP_1p8V;
    pwr->voltageConfig.VDDSysDS   = kAPP_LDOSys_40mA;
    pwr->voltageConfig.VDDCoreSrc = kAPP_DCDC;
    pwr->voltageConfig.VDDCoreLvl = kAPP_1p20V;
    pwr->voltageConfig.VDDCoreDS  = kAPP_DCDC_100mA;
    /*System SRAM*/
    pwr->sramConfig.retained = kAPP_SRAM_0kB;
    pwr->sramConfig.enabled = kAPP_SRAM_0kB;
}
```

The power menu updates `->powerMode`. In this lab, using the clock menu updates are, added to the `->clockConfig`.

After all the user input is received and it is time to configure the desired settings, the application calls `APP_ActiveLoop(&pwrConfig);` inside this function, these steps occur:

1. Go to a safe V_{DD_CORE} = 1.2V
2. `APP_InitMainClock` configures CORE0 main clock source and frequency
3. `APP_FindLowestVDDCoreVoltage` looks for the lowest V_{DD_CORE} possible depending on CORE0 clock frequency. If that voltage is lower than the current setup (1.2V), it will change the V_{DD_CORE} voltage level.
4. Disable un-used peripherals i.e. UART, LDO
5. while(1)



```
void APP_ActiveLoop (app_power_config_t *pwr)
{
    app_voltage_t initialVoltage;

    /* 1. Go to a safe VDD_CORE high drive = 1.2V*/
    initialVoltage.VDDSysLvl = kAPP_1p8V;
    initialVoltage.VDDSysDS = kAPP_LDOSys_40mA;
    initialVoltage.VDDCoreLvl = kAPP_1p20V;
    initialVoltage.VDDCoreDS = kAPP_DCDC_100mA;
    APP_SetRegulatorsVoltage(initialVoltage, kAPP_Active);

    /* 2. Set main clock frequency */
    APP_InitMainClock(pwr->clockConfig.mainClock);

    /* 3. Adjust the voltage to the selected user configuration */
    APP_FindLowestVDDCoreVoltage(pwr);
    APP_CheckValidVoltageConfig(pwr);
    APP_SetRegulatorsVoltage(pwr->voltageConfig, kAPP_Active);

    /* 4. Disable un-used SRAM and peripheral RAM */
    APP_InitSystemSRAM(&pwrConfig);

    /* 5. Disable un-used peripherals */
    APP_ActiveModeDisablePeripherals(pwr);

    while (1)
```

5. Lab 5: Analog Peripherals Menu

This lab uses a menu to enable the analog peripherals controlled by the ACTIVE_CFG1/LP_CFG1 registers. The application leverages the PM resource constraints to keep the peripherals enabled in the low-power mode, which manages the LP_CFG1 register. The PM will also limit the power mode entered based on these constraints.

The PM component does not manage the peripheral configuration in Active mode, or the ACTIVE_CFG1 register. Instead, the example application uses the MCUXpresso SDK drivers for these peripherals, which also set the ACTIVE_CFG1 bits.

This lab will also measure the current in low-power modes with analog peripherals enabled. And will compare the current against the power-mode baseline current from Lab 3: Static Power Modes, to calculate the peripheral adder current.

Please note, the configuration of these analog peripherals is using the default SDK driver and driver example settings. These peripherals may have configuration options that consume less current. An application will want to optimize these peripherals for the required use case.

5.1. Reset or power cycle the board to start fresh.

Follow

5.2. Lab 2: Measure Current with MCUXpresso IDE to **start recording the current** in the IDE with the MCU-Link Pro

5.3. Using the terminal, from the main menu, press the **'A' key** to load the Analog Peripheral menu. Notice this menu prints the register values for ACTIVE_CFG1 and LP_CFG1.

Analog Menu

```
Analog peripherals
Current register settings:
        ACTIVE_CFG1 = 0x0
        LP_CFG1     = 0x0
        0: Change peripherals enabled in ACTIVE    mode
        1: Change peripherals enabled in Low-Power modes

        b. Back to main menu
```

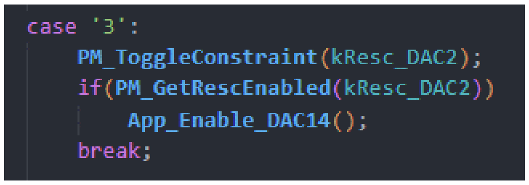
5.4. Press the **'1' key** to enable a peripheral in low-power mode

```
Analog in LP
Analog peripherals in low-power modes
Current register settings:
        ACTIVE_CFG1 = 0x0
        LP_CFG1     = 0x0
Listing current status.  Press a key to toggle that status:
        0. Toggle VREF,         currently disabled
        1. Toggle DAC0,         currently disabled
        2. Toggle DAC1,         currently disabled
        3. Toggle DAC2,         currently disabled
        4. Toggle OPAMP0,       currently disabled
        5. Toggle OPAMP1,       currently disabled
        6. Toggle OPAMP2,       currently disabled
        7. Toggle CMP0,         currently disabled
        8. Toggle CMP1,         currently disabled
        9. Toggle CMP2,         currently disabled
        b. Back to main menu
```

5.5. Press the **'3' key** to enable DAC2. Notice the update to both registers, and the menu now shows DAC2 is enabled.

```
Analog in LP
Current register settings:
        ACTIVE_CFG1 = 0x40
        LP_CFG1     = 0x40
Listing current status.  Press a key to toggle that status:
        0. Toggle VREF,         currently disabled
        1. Toggle DAC0,         currently disabled
        2. Toggle DAC1,         currently disabled
        3. Toggle DAC2,         currently enabled
        4. Toggle OPAMP0,       currently disabled
        5. Toggle OPAMP1,       currently disabled
        6. Toggle OPAMP2,       currently disabled
        7. Toggle CMP0,         currently disabled
        8. Toggle CMP1,         currently disabled
        9. Toggle CMP2,         currently disabled
        b. Back to main menu
```

Enabling a peripheral for low-power mode also means the peripheral must be enabled now in Active mode. You should see the board current increased when DAC2 was enabled. This menu toggles the PM resource constraint for the peripheral, see the code below. After toggling, if the constraint is enabled, the app also enables the peripheral. This uses the MCUXpresso SDK driver to initialize that peripheral, which also sets the ACTIVE_CFG1 bit.



```
case '3':
    PM_ToggleConstraint(kResc_DAC2);
    if(PM_GetRescEnabled(kResc_DAC2))
        App_Enable_DAC14();
    break;
```

5.6. The next steps will use the Power menu again to enter a low-power mode. Press the **'B' key** to return to the Main menu

```
Main Menu
Main Menu, press a key below:

        p: Power Modes Menu
        a: Analog Peripherals Menu
        c: Clocks Menu
        s: System SRAM Menu
        e: Enter Selected Mode
```

5.7. Press the **'P' key** to load the Power menu

```
Power Menu
Power Menu, press a key below:
        Current Selection = Active

        0. Choose a power mode
        e. Enter selected power mode
        w. Toggle CORE_WAKE domain constraint

        b. Back to main menu
```

5.8. Press the **'0' key (zero)** to choose a power mode

```
Power Mode Menu
Select a Power Mode:
        0. Active
        1. Sleep
        2. Deep Sleep
        3. Power Down
        4. Deep Power Down
```

5.9. Press the **'4' key** to select Deep Power Down mode. The menu now shows the selection is Deep Power Down mode. Consider the PM resource constraints that are currently set. Will the PM component enter Deep Power Down mode now, or choose a different mode?

```
Power Menu
Power Menu, press a key below:
        Current Selection = Deep Power Down

        0. Choose a power mode
        e. Enter selected power mode
        w. Toggle CORE_WAKE domain constraint, currently disabled
                Only applies in Power Down mode. If set, WAKE domain will
                be in Deep Sleep mode. Otherwise, WAKE domain will be in
                Power Down mode


        b. Back to main menu
```

5.10. Press the **‘E’ key** to enter the power mode. Notice the PM does not enter Deep Power Down mode. It enters Deep Sleep mode instead, and also reports the DAC2 resource constraint is the reason for this.

Power Mode Entry

```
Power Menu, press a key below:
        Current Selection = Deep Power Down

        0. Choose a power mode
        e. Enter selected power mode
        w. Toggle CORE_WAKE domain constraint

        b. Back to main menu

Calculated power mode is Deep Sleep    , because resc DAC2 peripheral is set, and limits to this mode.
```

5.11. Use the IDE to measure the current in Deep Sleep. How does this current compare to the Deep Sleep baseline measurement you recorded in Lab 3: Static Power Modes? Record the current adder when DAC2 is enabled.

The MCX N947 has 3 DACs on-chip. Do you think the DACs consume the same current? Let’s find out. The next steps release the PM constraint for DAC2. And then enable a different DAC to compare the current adder.

5.12. From the Main menu, press the **‘A’ key** to load the Analog Peripheral menu.

Main Menu

```
Main Menu, press a key below:

        p: Power Modes Menu
        a: Analog Peripherals Menu
        c: Clocks Menu
        s: System SRAM Menu
        e: Enter Selected Mode
```

5.13. Press the **‘0’ key (zero)** to change the peripheral in Active mode

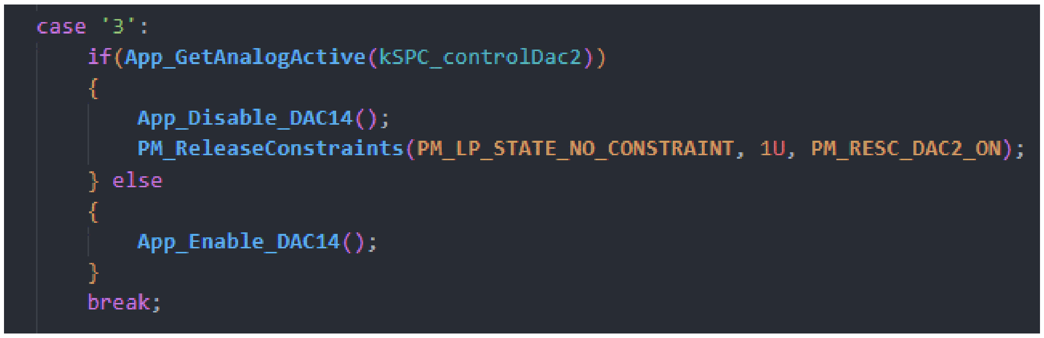
Analog in Active

```
Analog peripherals in Active mode
Current register settings:
        ACTIVE_CFG1 = 0x40
        LP_CFG1     = 0x40
Listing active status.  Press a key to toggle that status:

        0. Toggle VREF,         currently disabled
        1. Toggle DAC0,         currently disabled
        2. Toggle DAC1,         currently disabled
        3. Toggle DAC2,         currently enabled
        4. Toggle OPAMP0,       currently disabled
        5. Toggle OPAMP1,       currently disabled
        6. Toggle OPAMP2,       currently disabled
        7. Toggle CMP0,         currently disabled
        8. Toggle CMP1,         currently disabled
        9. Toggle CMP2,         currently disabled
        b. Back to main menu
```

5.14. Press the **‘3’ key** to disable DAC2. Notice the DAC2 bit is now cleared in both registers.

```
Analog in Active
Analog peripherals in Active mode
Current register settings:
            ACTIVE_CFG1 = 0x0
            LP_CFG1     = 0x0
Listing active status.  Press a key to toggle that status:

            0. Toggle VREF,              currently disabled
            1. Toggle DAC0,              currently disabled
            2. Toggle DAC1,              currently disabled
            3. Toggle DAC2,              currently disabled
            4. Toggle OPAMP0,            currently disabled
            5. Toggle OPAMP1,            currently disabled
            6. Toggle OPAMP2,            currently disabled
            7. Toggle CMP0,              currently disabled
            8. Toggle CMP1,              currently disabled
            9. Toggle CMP2,              currently disabled
            b. Back to main menu
```

When the Active peripheral menu disables a peripheral, it uses the MCUXpresso SDK driver to Deinit the driver, and clears the ACTIVE_CFG1 bit. You should see the Active current decreased when DAC2 was disabled. If the peripheral is disabled in Active mode, the application cannot use the peripheral in low-power mode either. This menu also releases the PM resource constraint for that peripheral.



```
case '3':
    if(App_GetAnalogActive(kSPC_controlDac2))
    {
        App_Disable_DAC14();
        PM_ReleaseConstraints(PM_LP_STATE_NO_CONSTRAINT, 1U, PM_RESC_DAC2_ON);
    } else
    {
        App_Enable_DAC14();
    }
    break;
```

5.15. Pick another DAC, and repeat these steps to measure and record the current adder for that DAC. How does the adder current compare to DAC2?

The next steps focus on enabling CMP2 and measuring the current. This resource constraint also depends on the CORE_WAKE domain.

5.16. Now that you are an expert with the Analog menu, use the menus to set CMP2 as the only low-power PM constraint. Note that the comparators set 2 bits in each register: 1 bit for the DAC included in the CMP, and 1 bit for the rest of the CMP.

```
Analog in Low Power

Analog peripherals in low-power modes
Current register settings:
    ACTIVE_CFG1 = 0x440000
    LP_CFG1     = 0x440000
Listing current status.  Press a key to toggle that status:
    0. Toggle VREF,       currently disabled
    1. Toggle DAC0,       currently disabled
    2. Toggle DAC1,       currently disabled
    3. Toggle DAC2,       currently disabled
    4. Toggle OPAMP0,     currently disabled
    5. Toggle OPAMP1,     currently disabled
    6. Toggle OPAMP2,     currently disabled
    7. Toggle CMP0,       currently disabled
    8. Toggle CMP1,       currently disabled
    9. Toggle CMP2,       currently enabled
    b. Back to main menu
```

5.17. Like the earlier steps, use the menus to attempt to enter Deep Power Down mode again. But it's no surprise to you now that the PM does not enter Deep Power Down mode. Notice that the CORE_WAKE domain is in Deep Sleep mode while the CORE_MAIN domain is in Power Down mode. CMP2 is a peripheral in the CORE_WAKE domain, so this PM resource constraint limits the CORE_WAKE domain to Deep Sleep mode.

```
Low Power Entry

Calculated power mode is Power Down with WAKE domain in Deep Sleep, because resc CMP2 peripheral is set, and
limits to this mode.
```

5.18. Like the previous steps, measure the current in this state: CORE_MAIN in Power Down, CORE_WAKE in Deep Sleep, and CMP2 enabled. Record this current.

To calculate the CMP2 adder current, we need a baseline current without CMP2 enabled. The baseline current you recorded earlier for Power Down mode is not usable for this, because that also had the CORE_WAKE domain in Power Down mode. The PM component has a resource constraint for the CORE_WAKE domain. The next steps will use the menus to enter the state with CORE_MAIN in Power Down and CORE_WAKE in Deep Sleep, with no other constraints set. Then this baseline current can be measured.

5.19. Use the IDE to start recording the current.

5.20. Use the Analog menus to release all Analog constraints. (Or cheat, and press the reset button on the board to start fresh.)

5.21. From the Main menu, press the **'P'** key to load the Power menu.

5.22. Press the **'W'** key to set the CORE_WAKE domain constraint.

```
CORE_WAKE
CORE_WAKE constraint enabled
```

5.23. Press the **'0'** key (zero), and then **'4'** key to select Deep Power Down mode.

5.24. Press the **'E'** key to enter the low-power mode

```
Power Mode Entry
Calculated power mode is Power Down with WAKE domain in Deep Sleep, because resc CORE_WAKE Domain is set,
and limits to this mode.
```

5.25. Measure the current, and record for this baseline. How does this current compare to your recorded baseline for Power Down mode (when the CORE_WAKE domain was in Power Down mode)? With this new baseline, calculate the current adder for CMP2.

5.26. Pick a different CMP, and repeat the steps to measure the current when this CMP is active. How does that current compare to when CMP2 is active? Why is it different?

5.27. Feel free to enable the remaining peripherals available in the Analog menu, and measure their current adders.

Module	Power Mode	ACTIVE_CFG1 = 0b0
		LP_CFG1 = 0b0
			ACTIVE_CFG1 = 0b1
			LP_CFG1 = 0b1
				Adder

VREF				
DAC0				
DAC1				
DAC2	Deep Sleep	133 uA	900 uA	767 µA
OpAmp0				
OpAmp1				
OpAmp2				
CMP0				
CMP1				
CMP2				

Table 5 Analog Current Adders Results LP_CFG1, ACTIVE_CFG1

The runtime menu gives the ability to disable an analog peripheral in the low-power mode by clearing the bit in LP_CFG1, but set the bit in ACTIVE_CFG1. For some analog peripherals, this configuration increases the current in the low-power mode. As shown below, the DAC2 adds 776 uA to Deep Power Down mode if configured like this. Optionally, you can make these measurements to see which peripherals should have the ACTIVE_CFG1 bits cleared before entering a low-power mode.

Module	Power Mode	ACTIVE_CFG1 = 0b0
		LP_CFG1 = 0b0
			ACTIVE_CFG1 = 0b1
			LP_CFG1 = 0b0
				Adder

VREF				
DAC0				
DAC1				
DAC2	DPD	1.38 uA	777 uA	776 µA
OpAmp0				
OpAmp1				
OpAmp2				
CMP0				
CMP1				
CMP2				

Table 6 Analog Current Adders Results ACTIVE_CFG1 only

SPOILER ALERT: the next page has the expected current measurements! Complete your measurements before viewing the next page.

Analog Peripherals Current Adders

Module	Power Mode	ACTIVE_CFG1 = 0b0 LP_CFG1 = 0b0	ACTIVE_CFG1 = 0b1 LP_CFG1 = 0b1	Adder
VREF	Deep Sleep	133 uA	497 uA	364 μA
DAC0	Deep Sleep	133 uA	150 uA	17 μA
DAC1	Deep Sleep	133 uA	151 uA	18 μA
DAC2	Deep Sleep	133 uA	900 uA	777 μA
OpAmp0	Deep Sleep	133 uA	462 uA	329 μA
OpAmp1	Deep Sleep	133 uA	471 uA	338 μA
OpAmp2	Deep Sleep	133 uA	481 uA	348 μA
CMP0	Deep Power Down	1.38 uA	8.42 uA	7.04 μA
CMP1	Deep Power Down	1.38 uA	10.26 uA	8.88 μA
CMP2	Power Down - Deep Sleep	25.86 uA	28.65 uA	2.79 uA

Table 7 Analog Current Adders Results LP_CFG1, ACTIVE_CFG1

Module	Power Mode	ACTIVE_CFG1 = 0b0 LP_CFG1 = 0b0	ACTIVE_CFG1 = 0b1 LP_CFG1 = 0b0	Adder
VREF	Deep Power Down	1.38 uA	19.66 uA	18.28 μA
DAC0	Deep Power Down	1.38 uA	19.93 uA	18.55 μA
DAC1	Deep Power Down	1.38 uA	20.33 uA	18.95 μA
DAC2	Deep Power Down	1.38 uA	777 uA	776 μA
OpAmp0	Deep Power Down	1.38 uA	1.38 uA	0
OpAmp1	Deep Power Down	1.38 uA	1.38 uA	0
OpAmp2	Deep Power Down	1.38 uA	1.38 uA	0
CMP0	Deep Power Down	1.38 uA	1.38 uA	0
CMP1	Deep Power Down	1.38 uA	1.38 uA	0
CMP2	Deep Power Down	1.38 uA	1.38 uA	0

Table 8 Analog Current Adders Results ACTIVE_CFG1 only

6. Lab 6: System SRAM

The MCX N947 has 512 KB of SRAM. RAMA is a special retention SRAM of 32 KB, because it can be retained in all power modes on the VBAT domain. The remaining 480 KB is referenced in this lab as System SRAM, and is the focus on this lab. The System SRAM is divided into 15 arrays with individual control to power optimize for the application. The SRAM menu enables different configuration options to measure their impact on the power consumption.

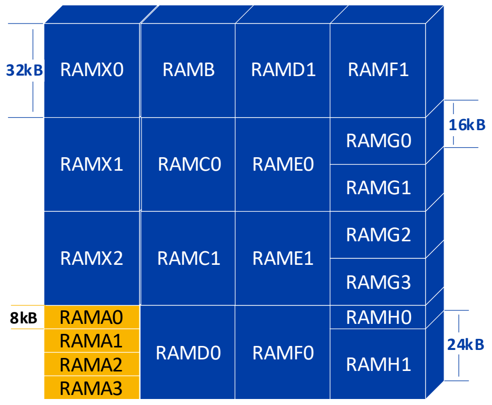


Figure 3 SRAM Memory Blocks

6.1. Reset or power cycle the board to start fresh.

Follow

6.2. Lab 2: Measure Current with MCUXpresso IDE to **start recording the current** in the IDE with the MCU-Link Pro

6.3. Using the terminal, from the main menu, press the **'S' key** to load the SRAM menu. Where you can find information on how this menu configures the memory arrays.

SRAM Menu

```
System SRAM Menu, press a key below:

    0. ENABLE System SRAM:
        SRAM array is enabled; array bit in SRAMDIS cleared.
        Enabled arrays are retained down to Deep Sleep mode
        Disabled arrays are not usable or retained in any power mode.

    1. RETAIN System SRAM:
        Retained arrays are also retained in Power Down mode.
        Clears array bit in SRAMRET register.
        Retaining the array will also enable the array.

    a. Apply System SRAM Configuration
    b. Back to main menu
```

6.4. To see this project's default values press **'a'** key. Notice this menu prints the register values for **SRAMDIS** and **SRAMRET**.

```
SRAM Registers
Current register settings:
    SRAMDIS = 0x3e007fff
    SRAMRET = 0x3e007fff
```

SRAMDIS[i] = 0b1 application does not use SRAM block at all (forces shutdown)
SRAMRET[i] = 0b1 application does not need SRAM block retained in Power Down

The default setting configures SRAMDIS, SRAMRET = 0x7fff, which sets both SRAMDIS, and SRAMRET bits. Making all SRAM shut down with power-gated, and data loss.

6.5. Press the **'S' key** and then **'1' key** to retain some SRAM. The menu provides different size options from 0kB to 480 kB.

```
SRAM Retain Sizes
RETAIN the SRAM in A, S, DS, and PD
    0.  0kB System SRAM
    1.  32kB System SRAM (RAMX0)
    2.  64kB System SRAM (RAMX0, RAMX1)
    3.  256kB System SRAM (~half)
    4.  480kB System SRAM (ALL)
```

6.6. Press the **'1' key** to retain 32KB of SRAM.

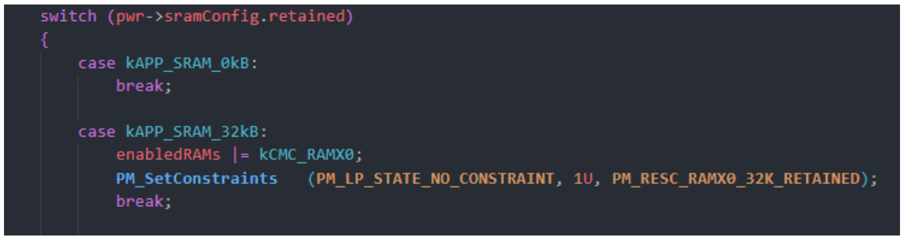
6.7. Notice that the Current register settings have not changed.

```
SRAM Registers
Current register settings:
    SRAMDIS = 0x3e007fff
    SRAMRET = 0x3e007fff

System SRAM Menu, press a key below:

    0. ENABLE System SRAM:
        SRAM array is enabled; array bit in SRAMDIS cleared.
        Enabled arrays are retained down to Deep Sleep mode
        Disabled arrays are not usable or retained in any power mode.

    1. RETAIN System SRAM:
        Retained arrays are also retained in Power Down mode.
        Clears array bit in SRAMRET register.
        Retaining the array will also enable the array.

    a. Apply System SRAM Configuration
    b. Back to main menu
```

6.8. Press the **'a' key** to Apply the System SRAM Configuration

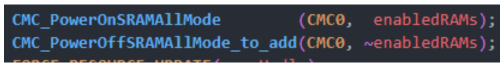
```
Apply SRAM
Current register settings:
      SRAMDIS = 0x3e007ffe
      SRAMRET = 0x3e007ffe

Main Menu, press a key below:

      p: Power Modes Menu
      a: Analog Peripherals Menu
      c: Clocks Menu
      s: System SRAM Menu
      e: Enter Selected Mode
```

Now the SRAMDIS and SRAMRET registers reflect 32 KB retention in Power Down with bit0 cleared in both registers. These two bits configure the RAMX0 array. Retaining SRAM is managed by the PM component as resource constraints. The code below shows how the menu sets a constraint for the RAMX0 SRAM array. With this constraint set, the PM component will clear the SRAMRET[RAMX0] bit.



```
    switch (pwr->sramConfig.retained)
    {
        case kAPP_SRAM_0kB:
            break;

        case kAPP_SRAM_32kB:
            enabledRAMs |= kCMC_RAMX0;
            PM_SetConstraints   (PM_LP_STATE_NO_CONSTRAINT, 1U, PM_RESC_RAMX0_32K_RETAINED);
            break;
```

To retain an SRAM array, it must also be enabled, with the SRAMDIS bit cleared. Enabling/Disabling SRAM arrays is not managed by the PM component. Instead, the application should configure the SRAM arrays it will be using. When the menu configures SRAM arrays to be retained, it also enables the arrays, with the code below:



```
CMC_PowerOnSRAMAllMode        (CMC0,  enabledRAMs);
CMC_PowerOffSRAMAllMode_to_add(CMC0, ~enabledRAMs);
```

6.9. Like you have done before on Lab 3: Static Power Modes, use the Power menu to attempt to enter Deep Power Down mode. Press the keys 'P', '0' (zero), '4', and 'E'. The PM component enters **Power Down**, because this is the deepest power mode that will retain the System SRAMs. Record the current measured

```
Power Entry
Calculated power mode is Power Down      with WAKE domain in Power Down, because
 resc RAMX0 SRAM is set, and limits to this mode
```

6.10. Record the current measured for the different RAM retention sizes.

7. Size	Power Mode	SRAMRET	I_{DD}	SRAM adder
0				
32 kB				
64 kB				
256 kB				
480 kB				

Table 9 SRAM retention current adders

The next steps will release the PM resource constraint (SRAMRET= 0b1) from the RAMX0 array but will keep the memory array enabled (SRAMDIS = 0b0) .

7.1. Press the **'S' key** to return to the SRAM menu.

7.2. Press the **'1' key** for the retention settings.

7.3. Press the **'0' (zero) key** to retain no SRAM.

7.4. Press the **'A' key** to apply your changes. The SRAMRET register should be updated like shown in the following terminal capture

```
Apply SRAM
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Current register settings:
    SRAMDIS = 0x3e007ffe
    SRAMRET = 0x3e007fff
```

7.5. Feel free to follow these steps and use the SRAM menu to configure other SRAM sizes. See how the SRAMDIS and SRAMRET registers are configured.

SPOILER ALERT: the next page has the SRAM current measurements! Complete your measurements before viewing the next page.

System SRAM Current Adders

Size	Power Mode	SRAMRET	I_{DD}	SRAM adder
0	Power Down	0x3E000_7FFF	2.82 µA	
32 kB	Power Down	0x3E000_7FFE	2.87 µA	50 nA
64 kB	Power Down	0x3E000_7FFC	2.99 µA	170 nA
256 kB	Power Down	0x3E000_7F00	3.56 µA	740 nA
480 kB	Power Down	0x3E000_0000	4.17 µA	1.35 µA

Table 10 SRAM retention current adders

8. Lab 7: Creating a Custom Use-Case

This Power Manager example provides runtime menus to quickly evaluate and learn about the power management features. But a true end application would not use these menus. This lab will test your knowledge of the previous labs, and emulate writing a custom application for a specific use-case. The goal is to configure the MCU for a use-case to measure the current consumption, without using the menus.

Here are your requirements for your modified application:

- All 480 KB of System SRAM should be retained
- All 3 analog comparators should be enabled
- Confirm the board current consumption in this use-case

The example firmware already has a function `APP_LabCustomUseCase();` to help with this use-case. This function is called in `main()`, after the MCU and PM components are initialized, and before the runtime menus. The steps in this lab will guide you to modify that function for this use-case, and enable you to evaluate it. You will need to write a few lines of code to complete the lab. In addition to these steps below, here are other resources you can use to complete this lab:

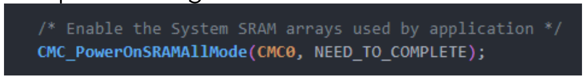
i. Revisit the previous labs. They include code snippets and details that will help with needed code changes in these steps.
ii. Use the example menus as a reference. You can use the menus to configure the same use-case, and can even step through the code to see how the menus work.
iii. Ask the lab instructors for help or suggestions.
iv. As a last resort, the example includes the completed function in the file lab_use-case_Final.c. If needed, you can refer to it, or copy that file.

8.1. Open the source file **lab_use-case.c**, and find the only function `APP_LabCustomUseCase`. Inside `APP_LabCustomUseCase()`, change the line `#if 0` to `#if 1`.

This step includes the code inside this function in the build. Note that the application will no longer build until you complete the steps. If you want to revert back to run the menus instead, just change this line back to `#if 0`.

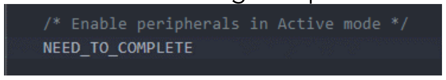
`APP_LabCustomUseCase` already has some code written for you, so you can focus on the power management use-case. Lines of code for you to complete are marked with **NEED_TO_COMPLETE**.

8.2. First enable the System SRAMs. The CMC driver API is already called in this function. But you need to complete an argument. HINT: search for this API and see how the example is using it.



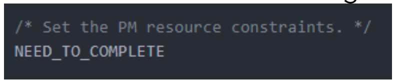
```
/* Enable the System SRAM arrays used by application */
CMC_PowerOnSRAMAllMode(CMC0, NEED_TO_COMPLETE);
```

8.3. Enable the 3 analog comparators. HINT: review how the analog menu does this.



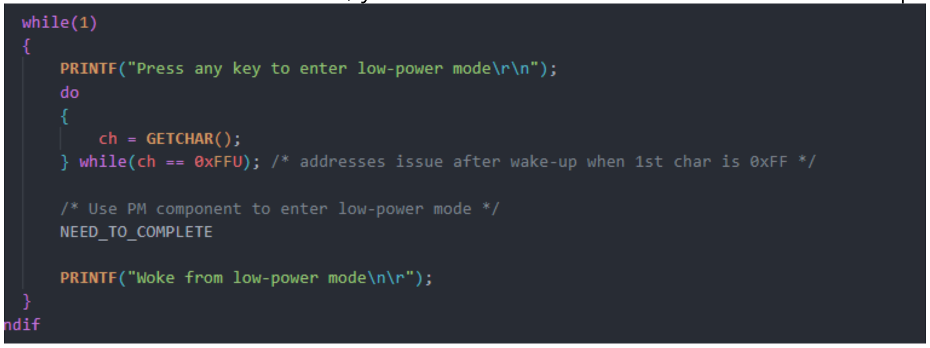
```
/* Enable peripherals in Active mode */
NEED_TO_COMPLETE
```

8.4. Now the active configuration is complete. But you need to set some constraints for the PM component. HINT: review how the Analog menus do this.



```
/* Set the PM resource constraints. */
NEED_TO_COMPLETE
```

8.5. Near the end of the function, you need to call the PM API to enter the low-power mode.



```
  while(1)
  {
      PRINTF("Press any key to enter low-power mode\r\n");
      do
      {
          ch = GETCHAR();
      } while(ch == 0xFFU); /* addresses issue after wake-up when 1st char is 0xFF */

      /* Use PM component to enter low-power mode */
      NEED_TO_COMPLETE

      PRINTF("Woke from low-power mode\n\r");
  }
ndif
```

8.6. Build the application and flash to your board.

8.7. Use the IDE to measure the current .

8.8. Work on answering these questions:

→ What is the deepest power mode for this use case?

→ What should be the values in the SRAMDIS and SRAMRET registers?

→ What should be the values in the ACTIVE_CFG1 and LP_CFG1 registers?

→ What is the board current for this use-case?

SPOILER ALERT: the next page has the results of this use-case! Complete the lab before viewing the next page.

→ What is the deepest power mode for this use case?
Power Down (MAIN domain) – Deep Sleep (WAKE domain)

→ What should be the values in the SRAMDIS and SRAMRET registers?
0x3E00_0000

→ What should be the values in the ACTIVE_CFG1 and LP_CFG1 registers?
0x0077_0000

→ What is the board current for this use-case?
56 uA

Custom Use-Case

```
Sticky RESET sources:
Warm,   PIN,
#######    MCX Nx4x Power Manager component demo    #######

---------------------- Normal Boot ----------------------
Calculated power mode is Power Down      with WAKE domain in Deep Sleep, because
 resc CMP2 peripheral is set, and limits to this mode.
SRAMDIS     =0x3e000000
SRAMRET     =0x3e000000
ACTIVE_CFG1 =0x770000
LP_CFG1     =0x770000
Press any key to enter low-power mode
Woke from low-power mode
Press any key to enter low-power mode
```