Table of Contents

Objectives

In this lab, you will learn:

- Utilize the Powerlib for PMIC
- Low power modes
- Operating modes

Hardware

- NXP MIMXRT685-EVK Rev E
- Micro-USB cable
- Digital Multimeter
- Jumper wire (Female-to-female)
- One pin header (Male-to-male)

Software

- Download the latest MCUXpresso IDE version 11.4.0 and above at (https://mcuxpresso.nxp.com)
- Download the latest SDK 2.10.0 and above for EVK-MIMXRT685 at (https://mcuxpresso.nxp.com)
- Power management labs
- Serial Terminal Console such as Putty, minicom, TeraTerm, etc.

Lab description

This lab shows user how to use the RT685 powerlib API functions to achieve desired operating mode and low power modes.

Power operating

Power rails

There are four power rails:

- 1.8V Always On - VDD_AO1V8
- 3.3V for IOs USB, and System - VDDIO0, VDDIO1, VDDIO2, USB1_VDD3V3, VDDA_BIAS
- 1.8V for IOS and System - VDDIO0, VDDIO1, VDDIO2, VDD1V8, VDDA_ADC1V8, VDDA_BIAS
- VDDCORE.

Only RTC wakeup timers, LDO_ENABLE, PMC_PMIC (interrupt and mode pins) and Reset Pin are on VDD_AO1V8 "Always On" domain when voltage (1.71 – 1.89V) is supplied.

LDO_ENABLE

LDO_ENABLE pin is used to select whether VDDCore is using internal LDO or from external. When LDO_ENALBE pin is pulled high, VDDCore will be supplied from internal LDO and the LDO is powered from the VDD1V8. When LDO_ENABLE pin is pulled low, VDDCore will be supplied from external supply.

External power supplies using Power Management IC (PMIC)

Using external PMIC provides flexibility to configure the power supply rails as application needs. For example, uSDHC may change the VDDIO rail from 3.3V to 1.8V, or deep sleep power rail configuration.

RT685 has a dedicated I^{2}C pins to communicate with PMIC, this allows different power configuration for low power modes or different operating mode at certain core voltage. RT685 allows PMIC to wake CM33 from PMIC_IRQ interrupt pin. The two PMIC_MODE pins allows CM33 to enter pre-configured PMIC power configuration without going through setup every time via I^{2}C.

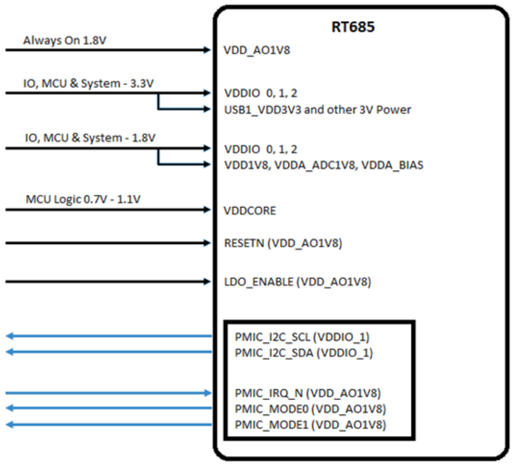


Figure 1 Power rails, LDO_ENABLE and PMIC connections

Power configuration

There are four type of power modes in RT6xx: Active, Sleep, Deep Sleep, Deep Power Down and Full Deep Power Down. The external PMIC referencing the PMIC_MODE pins from RT685 to provide appropriate power configurations.

Table 1 PMIC Mode Pins and Low Power modes

PMIC_MODE[1:0]	00	01	10	11
Low Power Modes	Active / Sleep	Deep Sleep	Deep Power Down	Full Deep Power Down

Active mode

The power consumption in Active mode is determined by PSCCTL, PDRUNCFG, clock sources, peripheral clocks and/or CPU operating modes.

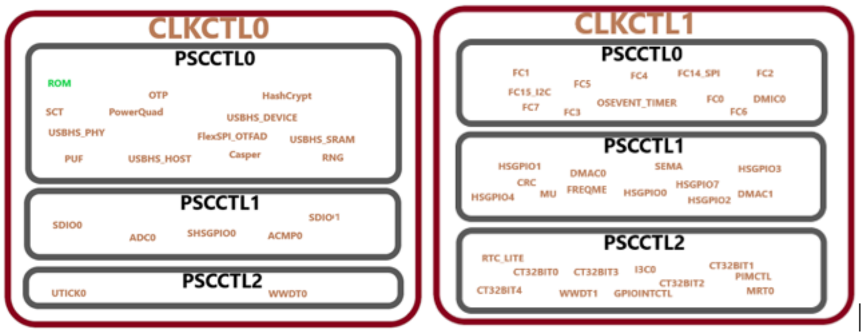


Figure 2 PSCCTL

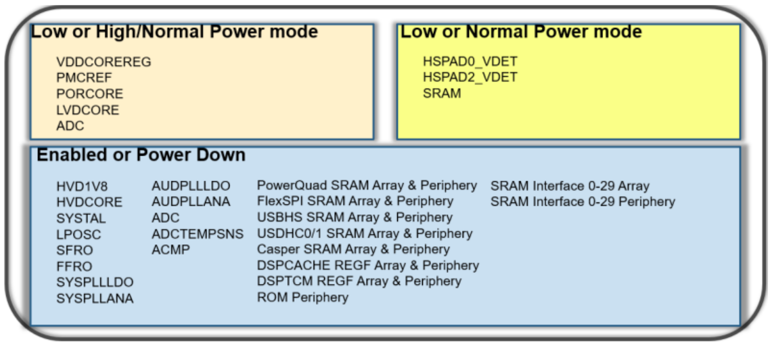

Low or High/Normal Power mode
VDDCOREREG
PMCREF
PORCORE
LVDCORE
ADC
Low or Normal Power mode
HSPAD0_VDET
HSPAD2_VDET
SRAM
Enabled or Power Down
HVD1V8
HVDCORE
SYSTAL
LPOSC
SFRO
FFRO
SYSPLLLDO
SYSPLLANA
AUDPLLLDO
AUDPLLANA
ADC
ADCTEMPSNS
ACMP
PowerQuad SRAM Array & Periphery
FlexSPI SRAM Array & Periphery
USBHS SRAM Array & Periphery
USDHC0/1 SRAM Array & Periphery
Casper SRAM Array & Periphery
DSPCACHE REGF Array & Periphery
DSPTCM REGF Array & Periphery
ROM Periphery
SRAM Interface 0-29 Array
SRAM Interface 0-29 Periphery

Figure 3 Power Control RUNCFG or SLEEPCFG

Sleep mode

Clock to the CM33 is gated, execution of instruction is suspended until interrupt occurs. Memories, clocks, and peripherals can be operation, and there are determined by PSCCTL, PDRUNCFG, clock sources, peripheral clocks. Any interrupt can wake up the CM33.

Deep Sleep mode

main_clk and CM33 will be gated to be in Deep Sleep. Memories, certain clocks and certain peripherals can be active are determined by PDSLEEPCFG.

Deep Power Down mode

VDDCore, memories, clocks and peripherals are shutoff except RTC "Always on" Domain. RTC wake-up timers, reset or PMIC interrupt can be used to wake up the system.

Full Deep Power Down mode

VDDCore, VDD1V8, 3V3 power rail, memories, clocks and peripherals are shutoff except RTC on Always on Domain. RTC wake timers, reset or PMIC interrupt can be used to wake up the system.

Wake up source

There are many peripherals that can be served as the wake-up function for the deep sleep low power mode. (Refer to STARTEN0 and STARTEN1 of SYSCTL0 in RT685 User Manual)

Timers function during deep sleep and deep power down

Table 2 shows timers' comparison in deep sleep, it also shows which clock source it requires to stay active. RTC is the only timer active during deep power down or full deep power down.

Table 2 Timers comparison in low power modes

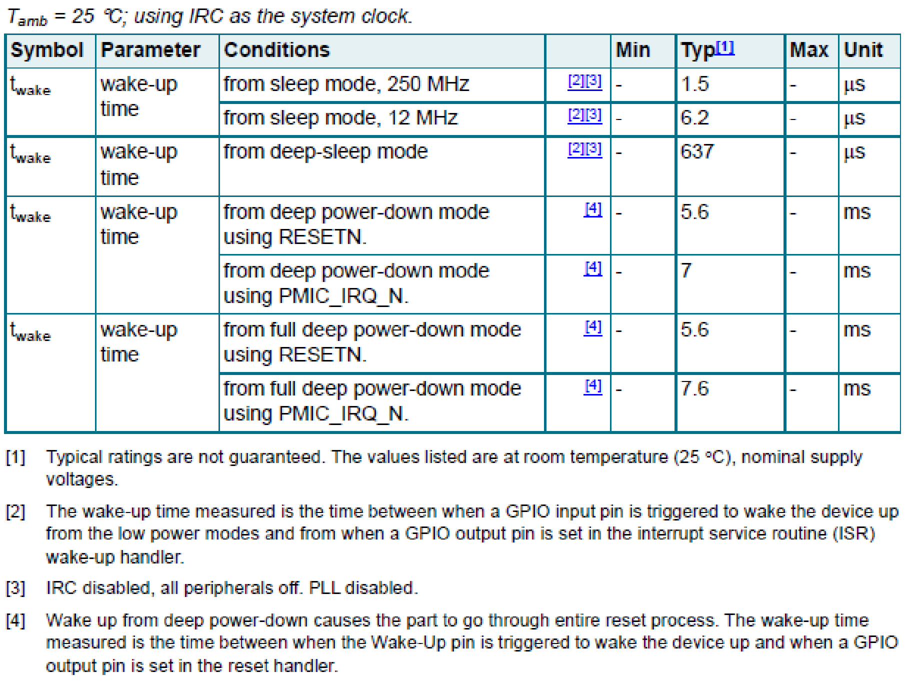
Timer			Deep Sleep												Deep Power Down
	Counter Resolution	Read back counter (Elapsed time)	main_clk	main_pll	aux0_pll	aux1_pll	audio_pll	hclk	APB	48/60m_irc	16m_irc	mclk	lposc	32khz	
Watchdog Timer (WDT)	24-bit	Yes	✓										✓		No
Micro-tick Timer (uTick)	31-bit	No	✓										✓		No
Multi-Rate Timer (MRT)	24-bit	Yes							✓						No
Counter Timer (Ctimer)	32-bit	Yes	✓	✓	✓	✓	✓			✓					No
SCTimer	16- or 32-bit	Yes	✓				✓			✓	✓	✓			No
Real_Time Clock (RTC)	15-(subsec) or 32-bit (sec)	Yes												✓	RTC Timer is on "Always ON" power domain.
OS Event Timer (OSTimer)	64-bit	Yes						✓					✓	✓	No

1. lposc has ±10% accuracy
2. 16m_irc has ±3% accuracy
3. 48/60m_irc has ±1% accuracy
4. 32khz accuracy depends on crystal setup
5. High frequency crystal that input to main_pll usually is between 10 – 50ppm

Peripherals to wake from deep sleep

The following peripherals can be used as a wake up source for deep sleep: DMA, GPIOs, Flexcomm (SPI, I^{2}C, I^{2}S, USART), ADC, DMIC, Hypervisor, Secureviolation, HW VAD, RNG, MU, FlexSPI, SDIO, I^{3}C, USB, PUF, PowerQuad, Casper, PMIC, and SHA.

Wake up process



T_{amb} = 25 °C; using IRC as the system clock.

Symbol	Parameter	Conditions		Min	Typ[1]	Max	Unit
t_{wake}	wake-up time	from sleep mode, 250 MHz	[2][3]	-	1.5	-	µs
		from sleep mode, 12 MHz	[2][3]	-	6.2	-	µs
t_{wake}	wake-up time	from deep-sleep mode	[2][3]	-	637	-	µs
t_{wake}	wake-up time	from deep power-down mode using RESETN.	[4]	-	5.6	-	ms
		from deep power-down mode using PMIC_IRQ_N.	[4]	-	7	-	ms
t_{wake}	wake-up time	from full deep power-down mode using RESETN.	[4]	-	5.6	-	ms
		from full deep power-down mode using PMIC_IRQ_N.	[4]	-	7.6	-	ms

[1] Typical ratings are not guaranteed. The values listed are at room temperature (25 °C), nominal supply voltages.

[2] The wake-up time measured is the time between when a GPIO input pin is triggered to wake the device up from the low power modes and from when a GPIO output pin is set in the interrupt service routine (ISR) wake-up handler.

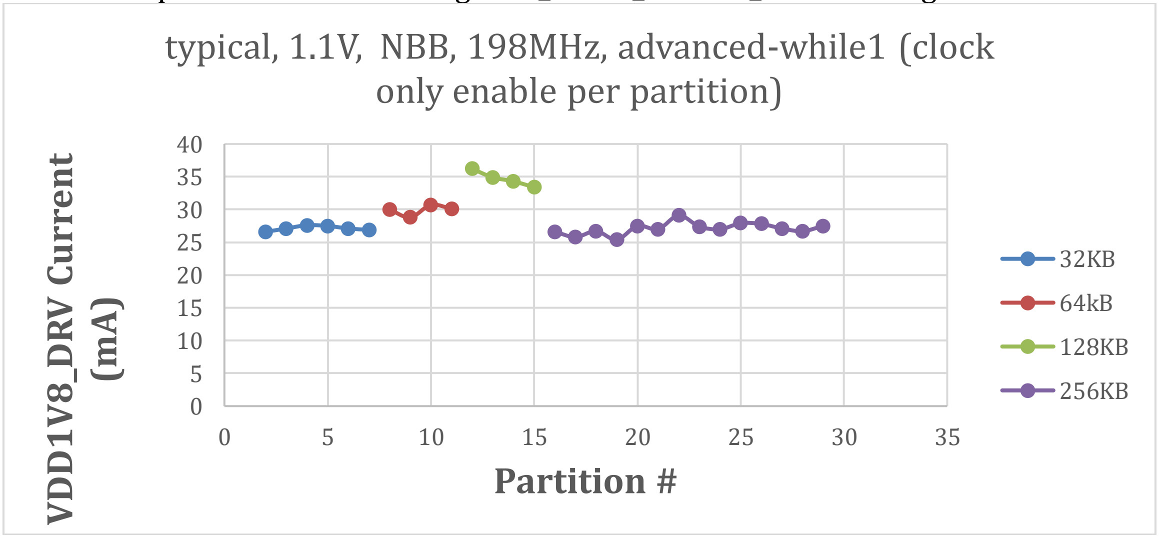
[3] IRC disabled, all peripherals off. PLL disabled.

[4] Wake up from deep power-down causes the part to go through entire reset process. The wake-up time measured is the time between when the Wake-Up pin is triggered to wake the device up and when a GPIO output pin is set in the reset handler.

Figure 4 Wake up time from low power modes

RAM partitions and power consumption

- Running advanced-while1 in different partition (2 to 29)
- Typical, 1.1V, 25, NBB, 198 MHz(PLL)
- All SRAM array/periphery are ON (set in PDRUNCFG2 and PDRUNCFG3) and one partition clocked using AHB_SRAM_ACCESS_DISABLE register.



Core operation in Active Mode

Maximum core operation frequencies and minimum core voltage in Active Mode

Some power consumption can be achieved by lowing the core voltage and core frequency. Below are the table showing the maximum operation frequencies and minimum voltage. The CPU frequencies show below must have SYSCPUAHBCLKDIV set to 0 (divided by 1).

Table 3 Maximum Core frequency and Core Voltage

Maximum Core Frequency (-20 °C to +70 °C)	Maximum Core Frequency (0 °C to +70 °C)	Minimum VDD Core Voltage
65 MHz	75 MHz	0.7V
140 MHz	155 MHz	0.8V
210 MHz	220 MHz	0.9V
270 MHz	275 MHz	1.0V
300 MHz	300 MHz	1.13V

Body Bias

CMOS transistors usually have three terminals – a source, gate and drain, it is also common to have a fourth terminal to the body or substrate. The threshold voltage can be affected by voltage source and voltage body.

RT685 has three type of body bias: Forward Body Bias (FBB), Reverse Body Bias (RBB) and Normal Body Bias (NBB).

Table 4 Body Bias

Body Bias	Description
FBB	Lower the threshold voltage and allows the device to turn on quickly for high performance, but has higher current leakage
RBB	Increase the threshold voltage and allows the device to turn on slowly, and it reduces current leakage
NBB	Source voltage and body voltage are matching

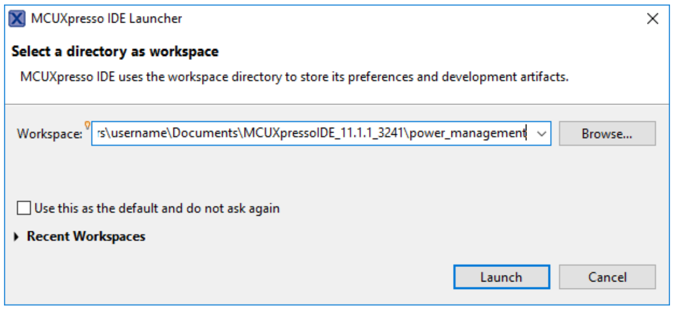
Power Lab 1 – Cortex M33 Operating frequency at 0.7V VDDCore

MCUXpresso SDK power_manager example

1. Open the MCUXpresso IDE 11.1.1 or later.

Figure 5 MCUXpresso IDE

2. Creating MCUXpresso new workspace.



MCUXpresso IDE Launcher

Select a directory as workspace

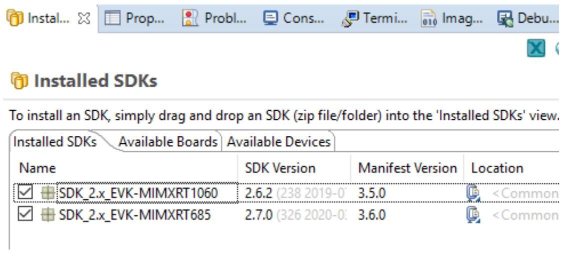
MCUXpresso IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: rs\username\Documents\MCUXpressoIDE_11.1.1_3241\power_management

Figure 6 workspace

Create a project name and click Launch to continue.

3. Skip this step if SDK 2.7.0 and above is installed.
   Drag and drop downloaded MCUXpresso SDK to Installed SDKs panel at the bottom of MCUXpresso IDE.



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

Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_EVK-MIMXRT1060	2.6.2 (238 2019-0	3.5.0	<Common
☑ SDK_2.x_EVK-MIMXRT685	2.7.0 (326 2020-0	3.6.0	<Common

Figure 7 Installed SDK

4. Import project.
   Select and click Import project(s) from file system... link at the bottom left corner panel of IDE.

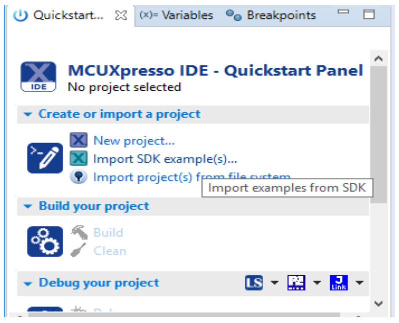
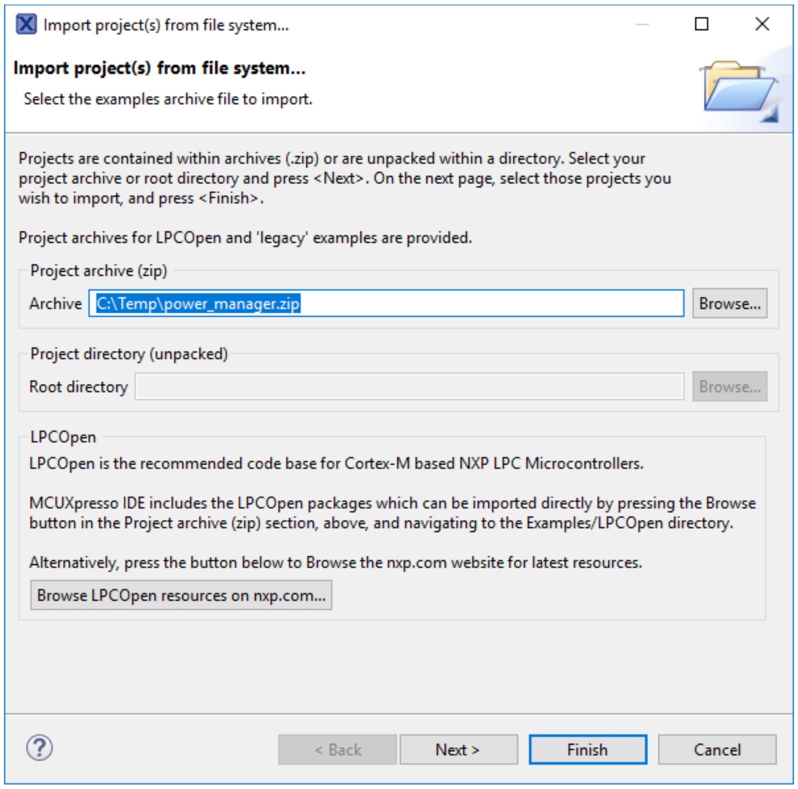


Figure 8 Import project

5. Import project from file system.
   Click Browse... to select power_manager.zip project. Select Next > to go to next step.



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 C:\Temp\power_manager.zip Browse...

Project directory (unpacked)

Root directory Browse...

LPCOpen

LPCOpen is the recommended code base for Cortex-M based NXP LPC Microcontrollers.

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.

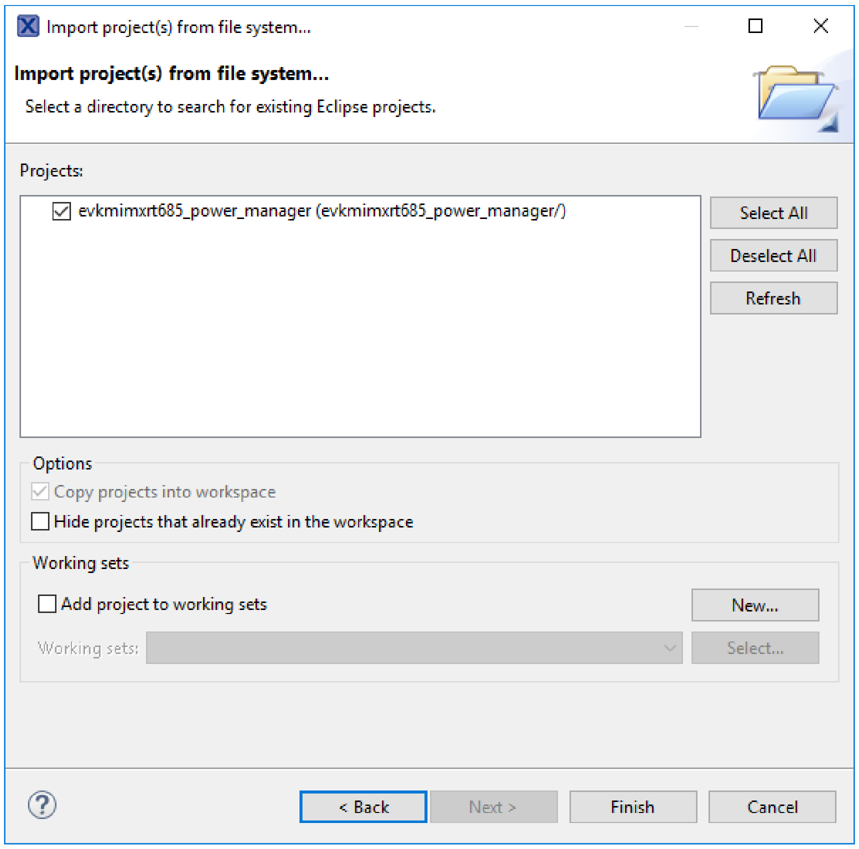
Alternatively, press the button below to Browse the nxp.com website for latest resources.

Browse LPCOpen resources on nxp.com...

< Back Next > Finish Cancel

Figure 9 Board selection

6. Select project to complete the project import.
   Make sure ☑ evkmimxrt685_power_manager (evkmimxrt685_power_manager/) is selected, then press Finish to complete the import.


Import project(s) from file system...

Import project(s) from file system...

Select a directory to search for existing Eclipse projects.

Projects:

- [✓] evkmimxrt685_power_manager (evkmimxrt685_power_manager/)

Options

- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

Figure 10 Select project

PCA9420 and powerlib API

7. SW1_OUT of PCA9420 and VDDCore

   VDDCore voltage is connected to SW1_OUT of PCA9420. Current SW1_OUT is programmed to output at 1.00V.

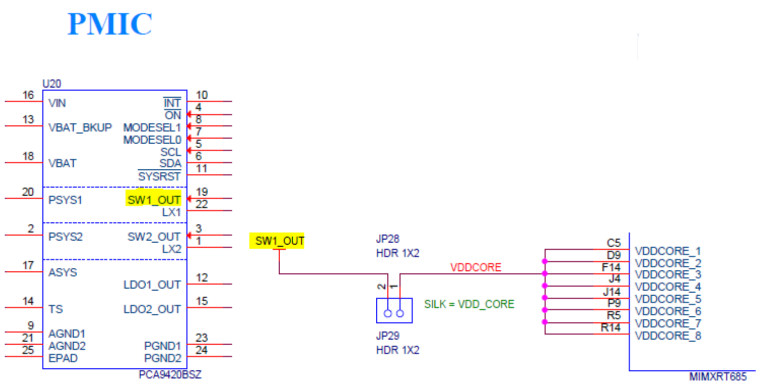


Figure 11 PCA9420 and VDDCore

8. Understanding PCA9420 MODECFG register and using powerlib API
   There are 6-bit settings for SW1_OUT in MODECFG_0_0 registers (MODECFG_0_0, address 22h).

Table 5 PCA9420 MODECFG register

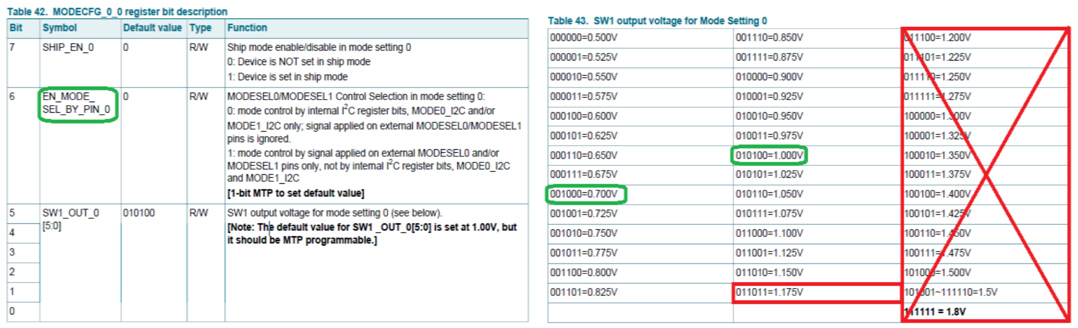


Table 42. MODECFG_0_0 register bit description

Bit	Symbol	Default value	Type	Function
7	SHIP_EN_0	0	R/W	Ship mode enable/disable in mode setting 0 0: Device is NOT set in ship mode 1: Device is set in ship mode
6	EN_MODE_SEL_BY_PIN_0	0	R/W	MODESEL0/MODESEL1 Control Selection in mode setting 0: 0: mode control by internal I^2C register bits, MODE0_I2C and/or MODE1_I2C only; signal applied on external MODESEL0/MODESEL1 pins is ignored. 1: mode control by signal applied on external MODESEL0 and/or MODESEL1 pins only, not by internal I^2C register bits, MODE0_I2C and MODE1_I2C **[1-bit MTP to set default value]**
5	SW1_OUT_0 [5:0]	010100	R/W	SW1 output voltage for mode setting 0 (see below). **[Note: The default value for SW1 _OUT_0[5:0] is set at 1.00V, but it should be MTP programmable.]**
4				
3				
2				
1				
0				

Table 43. SW1 output voltage for Mode Setting 0

000000=0.500V	001110=0.850V	~~011100=1.200V~~
000001=0.525V	001111=0.875V	~~011101=1.225V~~
000010=0.550V	010000=0.900V	~~011110=1.250V~~
000011=0.575V	010001=0.925V	~~011111=1.275V~~
000100=0.600V	010010=0.950V	~~100000=1.300V~~
000101=0.625V	010011=0.975V	~~100001=1.325V~~
000110=0.650V	010100=1.000V	~~100010=1.350V~~
000111=0.675V	010101=1.025V	~~100011=1.375V~~
001000=0.700V	010110=1.050V	~~100100=1.400V~~
001001=0.725V	010111=1.075V	~~100101=1.425V~~
001010=0.750V	011000=1.100V	~~100110=1.450V~~
001011=0.775V	011001=1.125V	~~100111=1.475V~~
001100=0.800V	011010=1.150V	~~101000=1.500V~~
001101=0.825V	011011=1.175V	~~101001~111110=1.5V~~
		~~**111111 = 1.8V**~~

The `BOARD_SetPmicVoltageForFreq()` power API in pmic_support.c setups desired VDDCore voltage without user provides value in above table. Simply passing Core and DSP frequency (Table 2), the power API will assign appropriate voltage for the VDDCore.

```
void BOARD_SetPmicVoltageForFreq(power_part_temp_range temp_range, uint32_t main_clk_freq,
uint32_t dsp_main_clk_freq)

power_part_temp_range = { kPartTemp_0C_P70C = 0U or kPartTemp_N20C_P70C = 1U }
main_clk_freq or dsp_main_clk_freq range up to 600,000,000
```

(NOTE: Do NOT use `PCA9420_WriteModeConfigs()` to write value over 011010b!!! This will cause permanent damage to the core.)

9. PCA9420 comes with voltage comparator on each rail (LDO1_OUT, LDO2_OUT, SW1_OUT and SW2_OUT), it compares the actual output voltage against 90% and 110% of its target value. When power-good is enabled, the voltage comparator reports “power-good” status if the nominal voltage is within 90% to 110%. When power-good is disabled, the nominal voltage is set outside the range of 90% to 110%.

   Current power_manager example has power-good disabled, when calling `BOARD_SetPmicVoltageForFreq()` to set VDDCore at 0.70 V. Actual measurement on VDDCore is approx. 0.74 V. Change the *`kPCA9420_PgoodDisabled`* to *`kPCA9420_PgoodEnabled`* in `BOARD_InitPmic()`. This will put the VDDCore close to 0.70 V.

```
pca9420Config.powerGoodEnable = kPCA9420_PGoodEnabled;
```

Core Frequency

10. Changing Core frequency

    Before calling `BOARD_BootClockRUN()`, the default 48 MHz (FFRO or 48/60m_irC) CPU frequency is derived from 40/60m_irc (MAINCLKSELA = 3, MAINCLKSELB = 0, and SYSCPUAHBCLKDIV = 0, see green highlight in Figure 12).

    Code executes after `BOARD_BootClockRUN()` sets the CPU to run at 250 MHz from main_pll_clk (MAINCLKSELB = 2, and SYSCPUAHBCLKDIV = 1, see orange highlight in Figure 12). The current value from VCO to Main PLL is 528 MHz, the PFD output frequency formula is FPFDn = FSYSPLL_OUT * (18 / PFDn). In `BOARD_BootClockRUN()`, PFD0 is set to 19. main_clk is 528,000,000 * (18 / 19) = 500,210,526 Hz. MAINPLLCLKDIV is 0 (divided by 1). However, SYSCPUAHBCLKDIV is divided by 2; therefore, the CPU clock is 500,210,526 / 2 = 250,105,263 Hz.

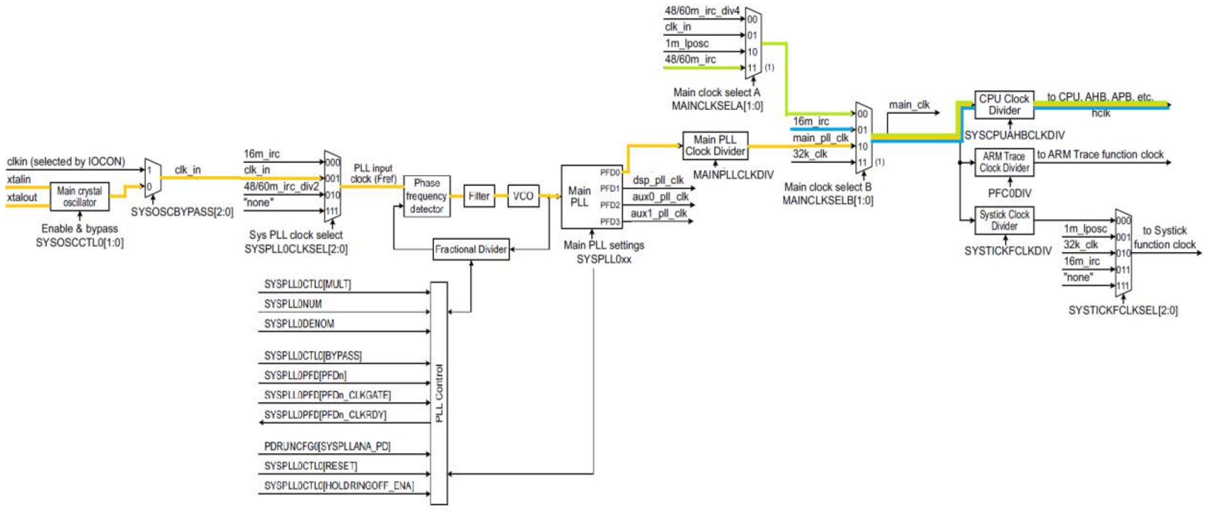


Figure 12 CPU Clock Source Overview

We will be focused more on the clock setup at main_clk (after MAINCLKSELA and MAINCLKSELB). To setup CPU to run less than 65 MHz, the closer frequency it can output is 60 MHz from 48/60m_IRC.

If the CPU clock source from the same clock source tries to change, the CPU clock must switch to another clock source before lowering the CPU clock. For example, the CPU is now using FFRO 48 MHz as clock source, and will be changed to FFRO 60 MHz. The CPU clock must be on another clock source such as SFRO or PLL before the change can be taken place.

In this lab example, the FFRO is set to run at 60 MHz from 48 MHz after the CPU is set to use the clock source from main_clk, then change the CPU clock source to 16m_irc (SFRO see blue highlight in Figure 12). The CPU frequency is now running at 8 MHz, the SYSCPUAHBCLKDIV is still divided by 2. The flash clock needs to adjust not to exceed core frequency. The flash is using main_pll_clk, so 500 MHz divided by 5 will generate 100 MHz flash clock. Switch the flash clock to FFRO with divided by 10 before switch the CPU clock source, this put the flash clock at 6 MHz. Set the SYSCPUAHBCLKDIV to 0 (divided by 1), the CPU frequency is at 16 MHz. Call `BOARD_SetPmicVoltageForFreq()` to set VDDCore to 0.70 V. Attach CPU clock source from SFRO to FFRO, the CPU is now running at 60 MHz. Adjust the flash clock again to run from FFRO at 30 MHz.

```
CLOCK_EnableFfroClk(kCLOCK_Ffro60M);

BOARD_SetFlexspiClock(3U, 10U);
CLOCK_AttachClk(kSFRO_to_MAIN_CLK); /* Let CPU run on ffro before configure SYS PLL. */
CLOCK_SetClkDiv(kCLOCK_DivSysCpuAhbClk, 1U); /* Set SYSCPUAHBCLKDIV divider to value 1 */

BOARD_SetPmicVoltageForFreq(kPartTemp_0C_P70C, 60000000, 0);
CLOCK_AttachClk(kFFRO_to_MAIN_CLK); /* Let CPU run on ffro before configure SYS PLL. */
BOARD_SetFlexspiClock(3U, 2U);
SystemCoreClock = CLOCK_GetFreq(kCLOCK_CoreSysClk);
```

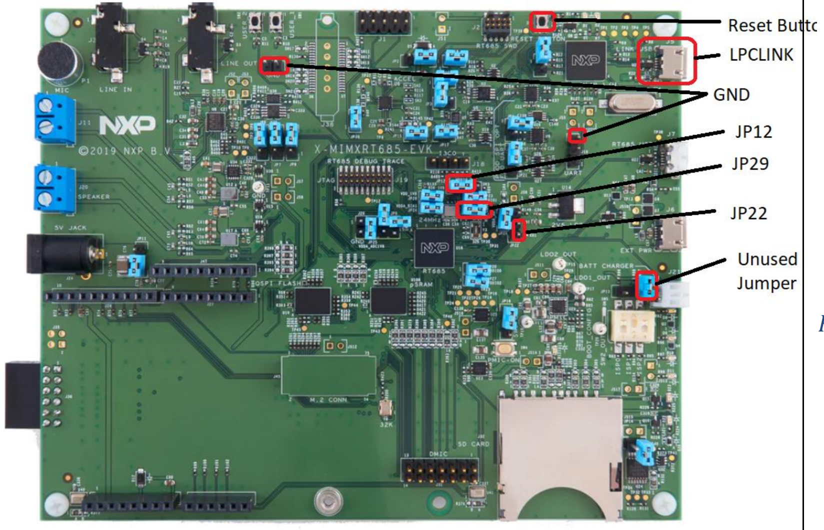
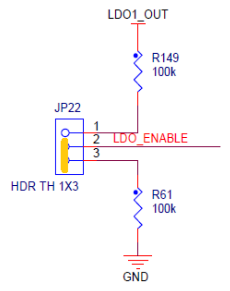
When using internal LDO, replace `BOARD_SetPmicVoltageForFreq()` with `POWER_SetLdoVoltageForFreq()`, the parameters are the same as PMIC. Remove **JP29** and change **JP22** to **1-2** as shown in Figure 13 VDDCore and LDO_Enable locationFigure 13.

NOTE: **JP12 1-2 MUST** be shunt when using internal LDO, make sure SW2_OUT 1.8V is supplied to VDDIO1 for the internal core voltage. **DO NOT** use internal LDO if **JP12 2-3** is shunt.

MCUXpresso SDK power_manager example

11. VDDCore and LDO_ENABLE location

Take an unused jumper as indicated in the image below and place in on 2-3 of JP22. Place multimeter to JP29 to measure VDDCore or place amp meter to JP29 (remove jumper) to	

measure current.  Reset Butto LPCLINK GND JP12 JP29 JP22 Unused Jumper *Figure 13 VDDCore and LDO_Enable location*	 LDO1_OUT R149 100k JP22 1 2 LDO_ENABLE 3 HDR TH 1X3 R61 100k GND *Figure 14 LDO_ENABLE selection*

NOTE: **JP12 1-2 MUST** be shunt when using internal LDO, make sure SW2_OUT 1.8V is supplied to VDDIO1 for the internal core voltage. **DO NOT** use internal LDO if **JP12 2-3** is shunt.

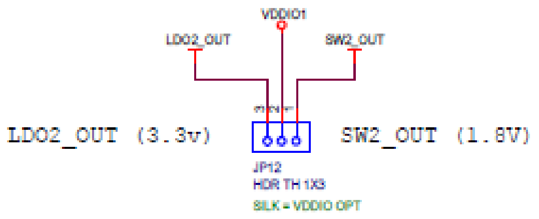


Figure 15 JP12 VDDIO1 power source selection

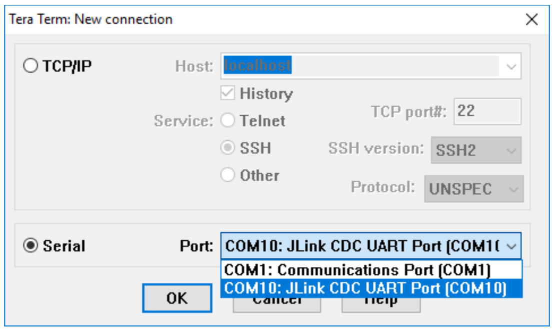
12. Power the board

    Connects USB micro cable from PC to J5 (LPCLink as indicated in Figure 13).

Serial Terminal

13. Open Serial Terminal program such as TeraTerm as shown below.

    Select serial port listed as Jlink CDC UART Port, and press to OK continue.


Tera Term: New connection

TCP/IP Host: localhost

History

Service: Telnet — TCP port#: 22

SSH — SSH version: SSH2

Other — Protocol: UNSPEC

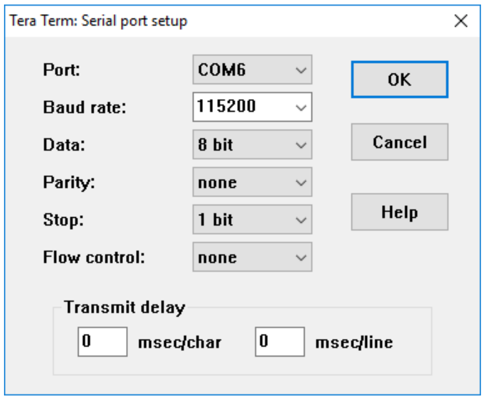
Serial [✓] Port: COM10: JLink CDC UART Port (COM1(

COM1: Communications Port (COM1)

COM10: JLink CDC UART Port (COM10)

Figure 16 Select LPC-LINKII serial port

Setup the serial port settings, and press OK to continue.


Tera Term: Serial port setup

Port:	COM6
Baud rate:	115200
Data:	8 bit
Parity:	none
Stop:	1 bit
Flow control:	none

Transmit delay

0 msec/char 0 msec/line

Figure 17 Serial port setup

Build the project

14. Build the example

Under the Quickstart Panel at the bottom left, click the Build under the Build your project.

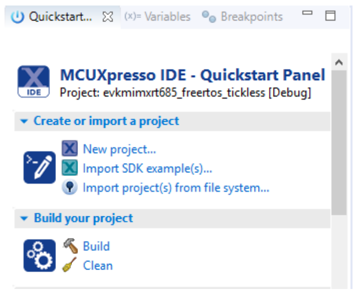
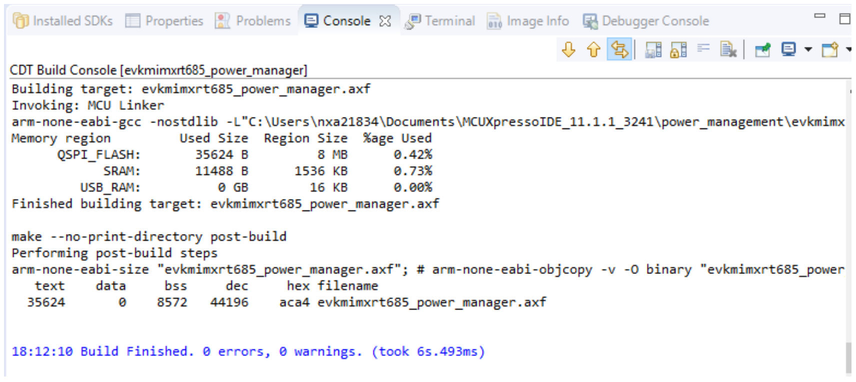

Quickstart...
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: evkmimxrt685_freertos_tickless [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean

Figure 18 Quickstart Panel - Build

At the middle bottom of IDE, the console will show the project built successfully.



```
CDT Build Console [evkmimxrt685_power_manager]
Building target: evkmimxrt685_power_manager.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\Users\nxa21834\Documents\MCUXpressoIDE_11.1.1_3241\power_management\evkmimx
Memory region         Used Size  Region Size  %age Used
      QSPI_FLASH:       35624 B         8 MB      0.42%
            SRAM:       11488 B      1536 KB      0.73%
         USB_RAM:           0 GB        16 KB      0.00%
Finished building target: evkmimxrt685_power_manager.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "evkmimxrt685_power_manager.axf"; # arm-none-eabi-objcopy -v -O binary "evkmimxrt685_power
   text    data     bss     dec     hex filename
  35624       0    8572   44196    aca4 evkmimxrt685_power_manager.axf

18:12:10 Build Finished. 0 errors, 0 warnings. (took 6s.493ms)
```

Figure 19 CDT Build Console

Run the project

15. Program the application

Make sure the SW5 (ISPs) is ON, OFF, ON from left to right.

Under the Quickstart Panel at the bottom left, click the Debug under the Debug your project.

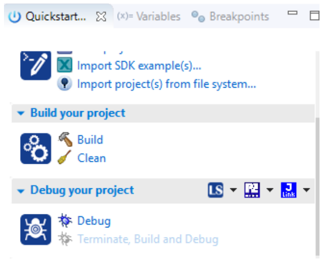
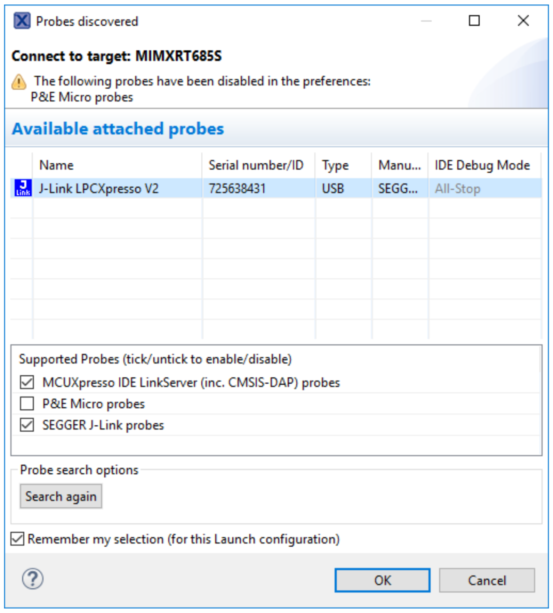


Figure 20 Quickstart Panel - Debug

16. A window will pop up to select debugger connects to the EVK. Select J-Link LPCXpresso V2 and press OK to continue.


Probes discovered

Connect to target: MIMXRT685S

⚠ The following probes have been disabled in the preferences:
P&E Micro probes

Available attached probes

Name	Serial number/ID	Type	Manu...	IDE Debug Mode
J-Link LPCXpresso V2	725638431	USB	SEGG...	All-Stop

Supported Probes (tick/untick to enable/disable)
- [x] MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- [ ] P&E Micro probes
- [x] SEGGER J-Link probes

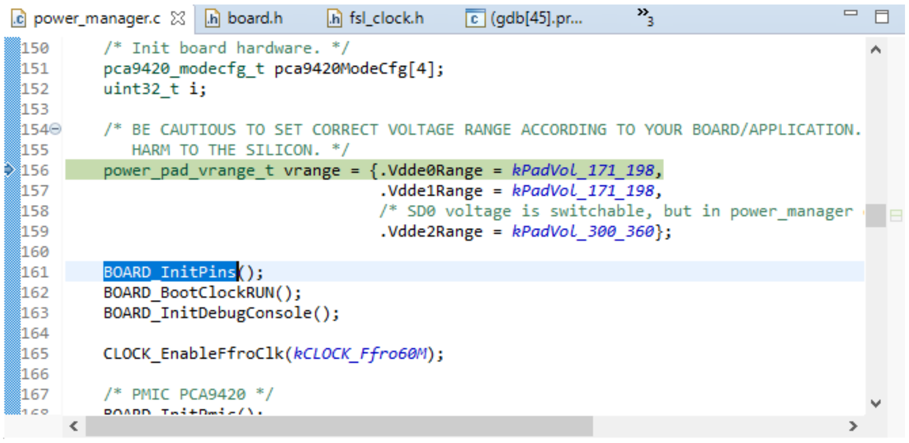
Probe search options
Search again

- [x] Remember my selection (for this Launch configuration)

OK Cancel

Figure 21 Selecting debugger

When it is successfully programmed, the program count will be halted at main() as shown in the source window.

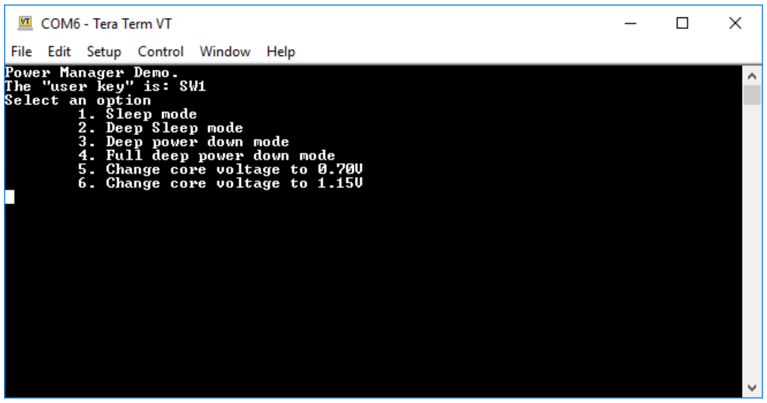


```
    /* Init board hardware. */
    pca9420_modecfg_t pca9420ModeCfg[4];
    uint32_t i;

    /* BE CAUTIOUS TO SET CORRECT VOLTAGE RANGE ACCORDING TO YOUR BOARD/APPLICATION.
       HARM TO THE SILICON. */
    power_pad_vrange_t vrange = {.Vdde0Range = kPadVol_171_198,
                                 .Vdde1Range = kPadVol_171_198,
                                 /* SD0 voltage is switchable, but in power_manager
                                 .Vdde2Range = kPadVol_300_360};

    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();

    CLOCK_EnableFroClk(kCLOCK_Ffro60M);

    /* PMIC PCA9420 */
```

Figure 22 Program counter stops at main()

You can continue to run the program by pressing (resume) . Or terminate the debug program by pressing , then press the reset button on the board (see Figure 13).

17. Either continue to run the program from debug or from reset, message will be shown as below.



```
Power Manager Demo.
The "user key" is: SW1
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
```

Figure 23 Serial output

18. Press 5 on the Terminal to change the VDDCore to 0.70 V, or press 6 to change the VDDCore to 1.15 V. Place a multimeter on JP29 to measure the voltage change when pressing number 5 or 6 key on the serial terminal.

Power Lab 2 – Wake from deep sleep using RTC, Pin Interrupt and USART

RTC wake (high resolution counter) as wake up source

1. Initialize RTC module and enable the RTC wake in STARTEN
   Enable the RTC output of the RTC oscillator 32,768 Hz. Failure to do so, the RTC counter will not function.

   After initialize the RTC by calling `RTC_Init()`, calls `RTC_StartTimer()` to enable the RTC. There are two type of wake-up features in RTC : WAKE and ALARM. WAKE is using 1 khz counter (milli seconds) up to 65.5s (0xFFFF) and ALARM is using RTC counter (seconds) up to 100+ years (0xFFFF_FFFF). WAKE interrupts are enabled in RTC by calling `RTC_EnableInterrupts()`. STARTEN and IRQ will be set when calling `EnableDeepSleepIRQ()`.

```
int main(void)
{
    …
    CLKCTL0->OSC32KHZCTL0 = 1;

    /* Initialize RTC timer */
    RTC_Init(RTC);
    RTC_StartTimer(RTC);

    /* enable RTC interrupt */
    RTC_EnableInterrupts(RTC, RTC_CTRL_WAKEDPD_EN_MASK);

    /* Enable RTC wake */
    EnableDeepSleepIRQ(RTC_IRQn);

    EnableIRQ(RTC_IRQn);
    BOARD_InitPins();
    BOARD_BootClockRUN();
    …
```

2. RTC IRQ Handle
   It is essential to have RTC IRQ to clear the flag after woke up from deep sleep. The flag needs to be cleared in order to avoid deadlock.

```
void RTC_IRQHandler(void)
{
    if (RTC_GetStatusFlags(RTC) & kRTC_WakeupFlag)
    {
        /* Clear wake flag */
        RTC_ClearStatusFlags(RTC, kRTC_WakeupFlag);
    }
    if (RTC_GetStatusFlags(RTC) & kRTC_AlarmFlag)
    {
        /* Clear alarm flag */
        RTC_ClearStatusFlags(RTC, kRTC_AlarmFlag);
    }

/* Add for ARM errata 838869, affects Cortex-M4, Cortex-M4F Store immediate overlapping
  exception return operation might vector to incorrect interrupt */
#if defined __CORTEX_M && (__CORTEX_M == 4U)
    __DSB();
#endif
}
```

3. Provides a counter value to RTC to wake from deep sleep. A write to this register will immediately start count down sequence in ms. In the example, 10000 (ms) is passed to `RTC_SetWakeupCount()`, the RT685 will wake from deep sleep after 10s.

```
            case kPmu_Deep_Sleep: /* Enter deep sleep mode. */
#if POWER_DOWN_PLL_BEFORE_DEEP_SLEEP
                /* Disable Pll before enter deep sleep mode */
                BOARD_DisablePll();
#endif
                RTC_SetWakeupCount(RTC, 10000);
                POWER_EnterDeepSleep(APP_EXCLUDE_FROM_DEEPSLEEP);
#if POWER_DOWN_PLL_BEFORE_DEEP_SLEEP
                /* Restore Pll before enter deep sleep mode */
                BOARD_RestorePll();
#endif
                break;
```

Note: Other timers that described in Timers function during deep sleep and deep power down use similar approach – setup timer IRQ Handle, setup timer clock source with the clock source excludes from deep sleep, set STARTEN and interrupt for the timer, and set time out on the timer before entering deep sleep.

```
static void UTickCallback(void)
{
}

CLOCK_AttachClk(kLPOSC_to_UTICK_CLK);
UTICK_Init(UTICK0);
UTICK_SetTick(UTICK0, kUTICK_Onetime, 1000000UL - 1, UTickCallback);
EnableDeepSleepIRQ(UTICK0_IRQn);
POWER_EnterDeepSleep(APP_EXCLUDE_FROM_DEEPSLEEP);
```

Pin Interrupt as wake up source

GPIO pins can be used as wake up source for deep sleep. Two type of interrupts for GPIO – Pin Interrupt (PINT) or GPIO interrupt. There are a total of 8 Pin Interrupts and 2 GPIO group interrupts in RT685. Current example is using Pin interrupt. During the deep sleep, user can press SW1 user button to wake from deep sleep.

```
static void APP_InitWakeupPin(void)
```

```
{
    gpio_pin_config_t gpioPinConfigStruct;

    /* Set SW pin as GPIO input. */
    gpioPinConfigStruct.pinDirection = kGPIO_DigitalInput;
    GPIO_PinInit(APP_USER_WAKEUP_KEY_GPIO, APP_USER_WAKEUP_KEY_PORT, APP_USER_WAKEUP_KEY_PIN,
&gpioPinConfigStruct);

    /* Configure the Input Mux block and connect the trigger source to PinInt channle. */
    INPUTMUX_Init(INPUTMUX);
    INPUTMUX_AttachSignal(INPUTMUX, kPINT_PinInt0, APP_USER_WAKEUP_KEY_INPUTMUX_SEL); /* Using
channel 0. */
    INPUTMUX_Deinit(INPUTMUX); /* Turnoff clock to inputmux to save power. Clock is only
needed to make changes */

    /* Configure the interrupt for SW pin. */
    PINT_Init(PINT);
    PINT_PinInterruptConfig(PINT, kPINT_PinInt0, kPINT_PinIntEnableFallEdge,
pint_intr_callback);
    PINT_EnableCallback(PINT); /* Enable callbacks for PINT */

    EnableDeepSleepIRQ(PIN_INT0_IRQn);
}
```

USART as another wake up source

There are two methods in deep sleep mode for USART. One is using 32 khz (RTC or lp_32k) runs in async mode, and another is sync mode that requires additional pin for clock to provide from master.

In this example, before RT685 goes to deep sleep, enabled the 32 khz clock source in USARTn->CFG. The `WAKECLK32KHZSEL` selection is assigned USART to use either RTC 32 khz or lp_32k (derived from 1m_osc then divided by 32, 1m_osc must be excluded from deep sleep).

This example uses RTC since it is turned on and its accuracy is better than lp_32k. Next step is to store the BRG and OSR value for baudrate restore before switching to 9600 baudrate. Enable USART Rx Interrupt(s), STARTEN and Flexcomm IRQ before going deep sleep.

Any data receive from USART Rx will wake up from deep sleep. Disable the 32k clock and quickly turn off the IRQ to prevent next Rx data to trigger interrupt since the current console output does not use interrupt for serial console. Restore the baud rate from 9600 to 115200.

```
void FLEXCOMM0_IRQHandler(void)
{
    uint8_t data;

    /* If new data arrived. */
    if ((kUSART_RxFifoNotEmptyFlag | kUSART_RxError) & USART_GetStatusFlags(USART0))
    {
        data = USART_ReadByte(USART0);
    }
```

```
/* Add for ARM errata 838869, affects Cortex-M4, Cortex-M4F Store immediate overlapping
  exception return operation might vector to incorrect interrupt */
#if defined __CORTEX_M && (__CORTEX_M == 4U)
    __DSB();
#endif
}

int main(void)
{
    ...

    uint32_t i, brg, osr;
    rtc_datetime_t date;
    ...

            case kPmu_Deep_Sleep: /* Enter deep sleep mode. */
#if POWER_DOWN_PLL_BEFORE_DEEP_SLEEP
                /* Disable Pll before enter deep sleep mode */
                BOARD_DisablePll();
#endif
                USART0->CFG |= USART_CFG_MODE32K_MASK;
                CLKCTL0->WAKECLK32KHZSEL = 0;

                brg = USART0->BRG;
                osr = USART0->OSR;
                USART0->BRG = 0;
                USART0->OSR = 13;

                USART_EnableInterrupts(USART0, kUSART_RxLevelInterruptEnable |
kUSART_RxErrorInterruptEnable);
                EnableDeepSleepIRQ(FLEXCOMM0_IRQn);
                RTC_SetWakeupCount(RTC, 10000);
                POWER_EnterDeepSleep(APP_EXCLUDE_FROM_DEEPSLEEP);
                USART0->CFG &= ~USART_CFG_MODE32K_MASK;
                DisableIRQ(FLEXCOMM0_IRQn);
                USART0->BRG = brg;
                USART0->OSR = osr;
#if POWER_DOWN_PLL_BEFORE_DEEP_SLEEP
                /* Restore Pll before enter deep sleep mode */
                BOARD_RestorePll();
#endif
                break;
```

Understanding deep sleep parameters that passes to deep sleep power API

The power API to enter deep sleep is POWER_EnterDeepSleep(), the API passes value of array.
Parameter 0: RUN/SLEEPCFG
Parameter 1: Peripherals RAM
Parameter 2: Array RAM partitions
Parameter 3: Periphery RAM partitions

When the bit is set in the parameter, it will be excluded from deep sleep. For example, FFRO needs to be active during deep sleep, set bit 16 of parameter 0. (SYSCTL0->PDSLEEPCFG0 in User Manual).

In the example, parameter 1 has FLEXSPI RAM excluded, parameter 2 has 21 array partitions excluded.

```
#define APP_DEEPSLEEP_RUNCFG0 (SYSCTL0_PDSLEEPCFG0_RBB_PD_MASK | SYSCTL0_PDSLEEPCFG0_FFRO_PD_MASK)
#define APP_DEEPSLEEP_RAM_APD 0xFFFFF8U
#define APP_DEEPSLEEP_RAM_PPD 0x0U
#define APP_EXCLUDE_FROM_DEEPSLEEP
\
    (((const uint32_t[]){APP_DEEPSLEEP_RUNCFG0,
\
                         (SYSCTL0_PDSLEEPCFG1_FLEXSPI_SRAM_APD_MASK |
SYSCTL0_PDSLEEPCFG1_FLEXSPI_SRAM_PPD_MASK), \
                         APP_DEEPSLEEP_RAM_APD, APP_DEEPSLEEP_RAM_PPD}))
```

Build the project

4. Build the example

   Under the Quickstart Panel at the bottom left, click the Build under the Build your project.

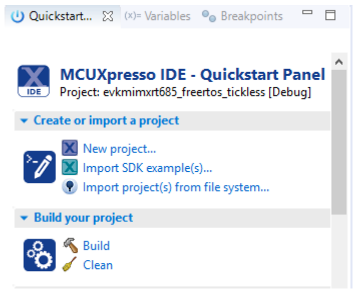
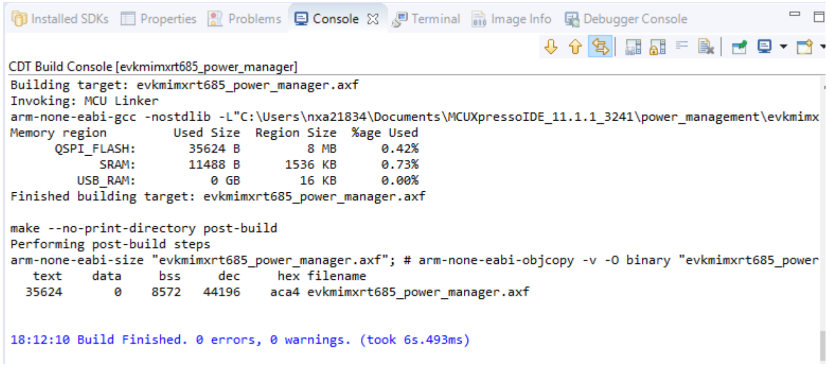

Quickstart...
(x)= Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: evkmimxrt685_freertos_tickless [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean

Figure 24 Quickstart Panel - Build

At the middle bottom of IDE, the console will show the project built successfully.



```
CDT Build Console [evkmimxrt685_power_manager]
Building target: evkmimxrt685_power_manager.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\Users\nxa21834\Documents\MCUXpressoIDE_11.1.1_3241\power_management\evkmimx
Memory region         Used Size  Region Size  %age Used
      QSPI_FLASH:       35624 B         8 MB      0.42%
            SRAM:       11488 B      1536 KB      0.73%
         USB_RAM:           0 GB        16 KB      0.00%
Finished building target: evkmimxrt685_power_manager.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "evkmimxrt685_power_manager.axf"; # arm-none-eabi-objcopy -v -O binary "evkmimxrt685_power
   text    data     bss     dec     hex filename
  35624       0    8572   44196    aca4 evkmimxrt685_power_manager.axf


18:12:10 Build Finished. 0 errors, 0 warnings. (took 6s.493ms)
```

Figure 25 CDT Build Console

Run the project

5. Program the application

Make sure the SW5 (ISPs) is ON, OFF, ON from left to right.

Under the Quickstart Panel at the bottom left, click the Debug under the Debug your project.

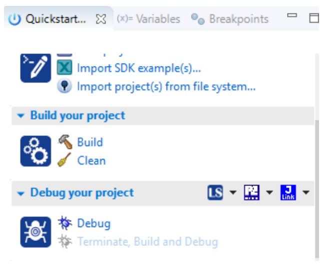
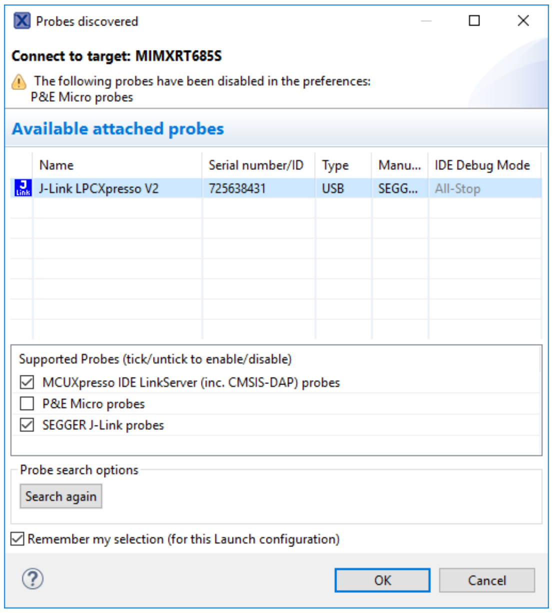


Figure 26 Quickstart Panel - Debug

6. A window will pop up to select debugger connects to the EVK. Select J-Link LPCXpresso V2 and press OK to continue.


Probes discovered

Connect to target: MIMXRT685S

The following probes have been disabled in the preferences:
P&E Micro probes

Available attached probes

Name	Serial number/ID	Type	Manu...	IDE Debug Mode
J-Link LPCXpresso V2	725638431	USB	SEGG...	All-Stop

Supported Probes (tick/untick to enable/disable)

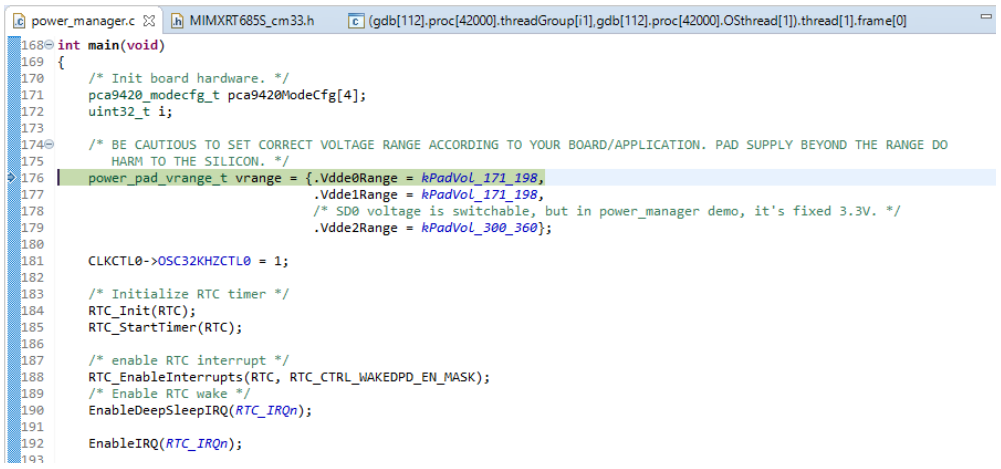
- ☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- ☐ P&E Micro probes
- ☑ SEGGER J-Link probes

Probe search options

☑ Remember my selection (for this Launch configuration)

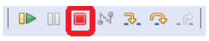
Figure 27 Selecting debugger

When it is successfully programmed, the program count will be halted at main() in the source window as shown below.


power_manager.c
MIMXRT685S_cm33.h
(gdb[112].proc[42000].threadGroup[i1],gdb[112].proc[42000].OSthread[1]).thread[1].frame[0]

```
int main(void)
{
    /* Init board hardware. */
    pca9420_modecfg_t pca9420ModeCfg[4];
    uint32_t i;

    /* BE CAUTIOUS TO SET CORRECT VOLTAGE RANGE ACCORDING TO YOUR BOARD/APPLICATION. PAD SUPPLY BEYOND THE RANGE DO
       HARM TO THE SILICON. */
    power_pad_vrange_t vrange = {.Vdde0Range = kPadVol_171_198,
                                 .Vdde1Range = kPadVol_171_198,
                                 /* SD0 voltage is switchable, but in power_manager demo, it's fixed 3.3V. */
                                 .Vdde2Range = kPadVol_300_360};

    CLKCTL0->OSC32KHZCTL0 = 1;

    /* Initialize RTC timer */
    RTC_Init(RTC);
    RTC_StartTimer(RTC);

    /* enable RTC interrupt */
    RTC_EnableInterrupts(RTC, RTC_CTRL_WAKEDPD_EN_MASK);
    /* Enable RTC wake */
    EnableDeepSleepIRQ(RTC_IRQn);

    EnableIRQ(RTC_IRQn);
```

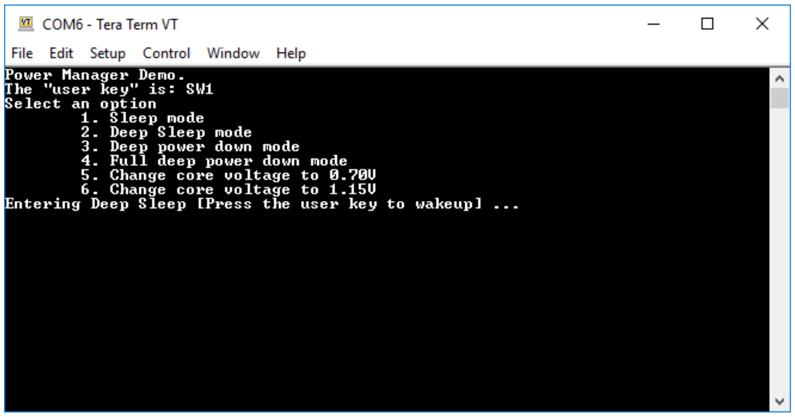
Figure 28 Program counter stops at main()

Terminate the debug program by pressing , then press the reset button on the board (see Figure 13).

NOTE: Do not use debugger when the code is going to perform low power modes. The debugger will shut off during low power modes, this will cause the IDE not response.

Deep sleep wake by RTC

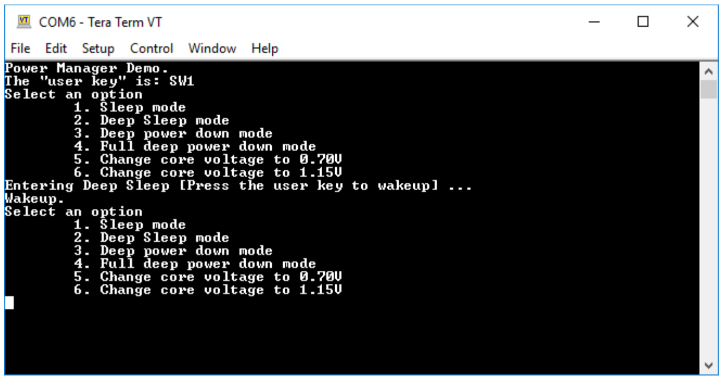
7. After reset, message will be shown as below. Press 2 on the serial terminal window to enter deep sleep.



```
Power Manager Demo.
The "user key" is: SW1
Select an option
          1. Sleep mode
          2. Deep Sleep mode
          3. Deep power down mode
          4. Full deep power down mode
          5. Change core voltage to 0.70V
          6. Change core voltage to 1.15V
Entering Deep Sleep [Press the user key to wakeup] ...
```

Figure 29 Deep sleep

8. After wake has elapsed, it will back to the selections.



```
Power Manager Demo.
The "user key" is: SW1
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
Entering Deep Sleep [Press the user key to wakeup] ...
Wakeup.
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
```

Figure 30 Deep sleep by RTC

Deep sleep wake by Pin

9. Press 2 again to enter deep sleep, this time using SW1 User button to wake from deep sleep prior to RTC wake (10s).

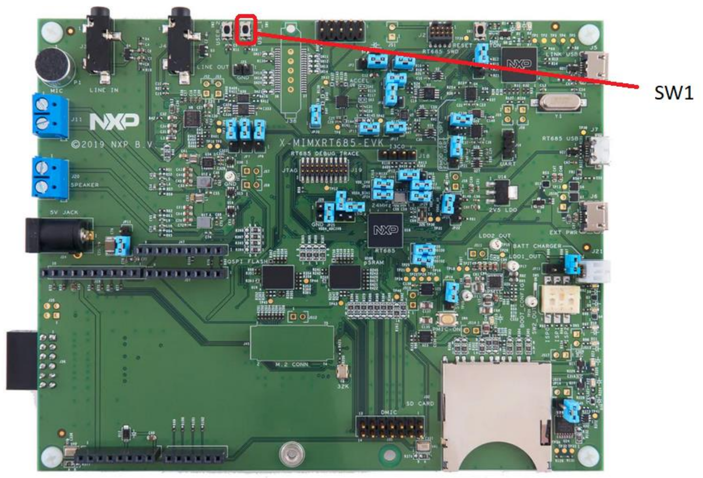
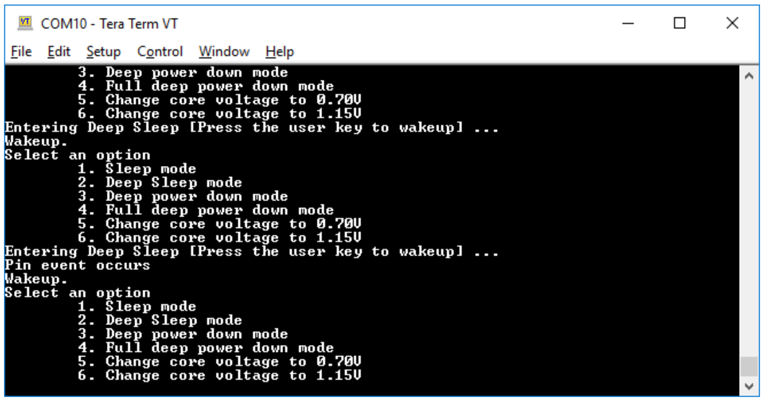


Figure 31 SW1 location

10. After wake, it will back to the selections.



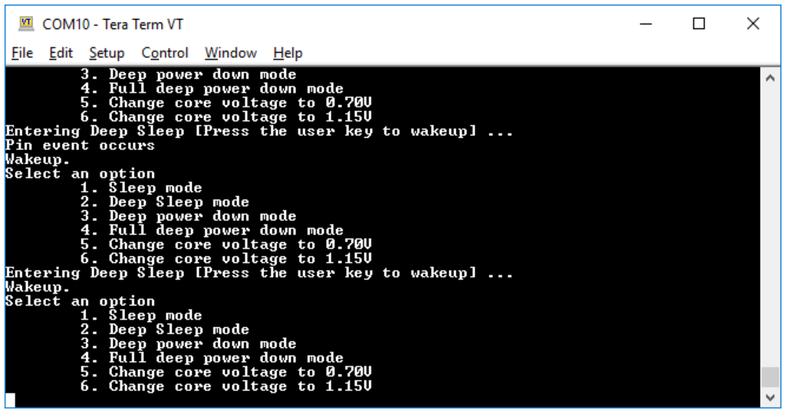
```
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
Entering Deep Sleep [Press the user key to wakeup] ...
Wakeup.
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
Entering Deep Sleep [Press the user key to wakeup] ...
Pin event occurs
Wakeup.
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
```

Figure 32 Deep sleep by Pin

Deep sleep wake by USART

11. Press 2 again to enter deep sleep, this time enter any key on the serial terminal window to wake from deep sleep prior to RTC wake (10s).

12. After wake, it will back to the selections.



```
          3. Deep power down mode
          4. Full deep power down mode
          5. Change core voltage to 0.70V
          6. Change core voltage to 1.15V
Entering Deep Sleep [Press the user key to wakeup] ...
Pin event occurs
Wakeup.
Select an option
          1. Sleep mode
          2. Deep Sleep mode
          3. Deep power down mode
          4. Full deep power down mode
          5. Change core voltage to 0.70V
          6. Change core voltage to 1.15V
Entering Deep Sleep [Press the user key to wakeup] ...
Wakeup.
Select an option
          1. Sleep mode
          2. Deep Sleep mode
          3. Deep power down mode
          4. Full deep power down mode
          5. Change core voltage to 0.70V
          6. Change core voltage to 1.15V
```

Figure 33 Deep sleep wake by USART

Power Lab 3 – Using RTC to wake from deep power down

1. Configure the RTC alarm to wake from deep power down. Append `RTC_CTRL_ALARMDPD_EN_MASK` to `RTC_EnableInterrupts()`.

```
    RTC_StartTimer(RTC);

    /* enable RTC interrupt */
    RTC_EnableInterrupts(RTC, RTC_CTRL_WAKEDPD_EN_MASK | RTC_CTRL_ALARMDPD_EN_MASK);
    /* Enable RTC wake */
    EnableDeepSleepIRQ(RTC_IRQn);
```

2. Setup RTC time and alarm
   Setup the date and convert to seconds using `RTC_SetDatetime()`. Add 20s to the date.second and pass it to `RTC_SetAlarm()`. Or, direct register write using RTC->MATCH = RTC->COUNT + 20;

```
int main(void)
{
    /* Init board hardware. */
    pca9420_modecfg_t pca9420ModeCfg[4];
    uint32_t i;
    rtc_datetime_t date;

    …

            case kPmu_Deep_PowerDown: /* Enter deep power down mode. */
                PRINTF(
                    "Press any key to confirm to enter the deep power down mode and wakeup the
device by "
                    "reset.\r\n\r\n");
                GETCHAR();

                date.year = 2020;
                date.month = 4;
                date.day = 6;
                date.hour = 9;
                date.minute = 0;
                date.second = 0;
                RTC_SetDatetime(RTC, &date);
                date.second += 20;
                RTC_SetAlarm(RTC, &date);
                POWER_EnterDeepPowerDown(APP_EXCLUDE_FROM_DEEP_POWERDOWN);
                break;
```

Build the project

3. Build the example
   Under the Quickstart Panel at the bottom left, click the Build under the Build your project.

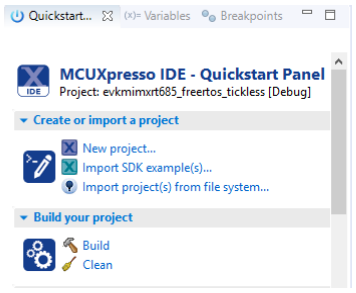
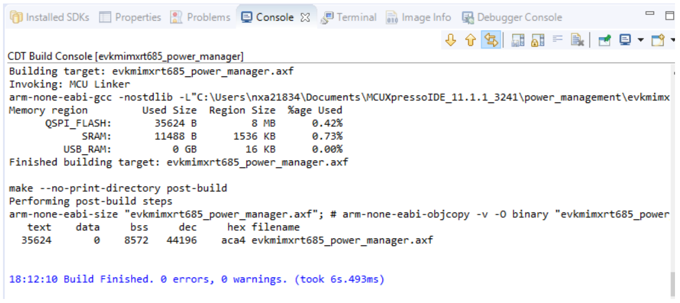

Quickstart...
(x)= Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: evkmimxrt685_freertos_tickless [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean

Figure 34 Quickstart Panel - Build

At the middle bottom of IDE, the console will show the project built successfully.



```
CDT Build Console [evkmimxrt685_power_manager]
Building target: evkmimxrt685_power_manager.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\Users\nxa21834\Documents\MCUXpressoIDE_11.1.1_3241\power_management\evkmimx
Memory region         Used Size  Region Size  %age Used
      QSPI_FLASH:       35624 B         8 MB      0.42%
            SRAM:       11488 B      1536 KB      0.73%
         USB_RAM:          0 GB        16 KB      0.00%
Finished building target: evkmimxrt685_power_manager.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "evkmimxrt685_power_manager.axf"; # arm-none-eabi-objcopy -v -O binary "evkmimxrt685_power
   text    data     bss     dec     hex filename
  35624       0    8572   44196    aca4 evkmimxrt685_power_manager.axf


18:12:10 Build Finished. 0 errors, 0 warnings. (took 6s.493ms)
```

Figure 35 CDT Build Console

Run the project

4. Program the application

   Make sure the SW5 (ISPs) is ON, OFF, ON from left to right.

   Under the Quickstart Panel at the bottom left, click the Debug under the **Debug your project**.

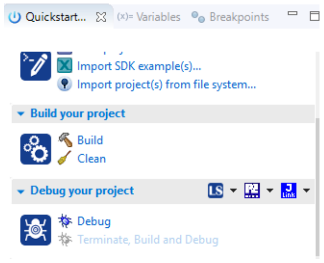
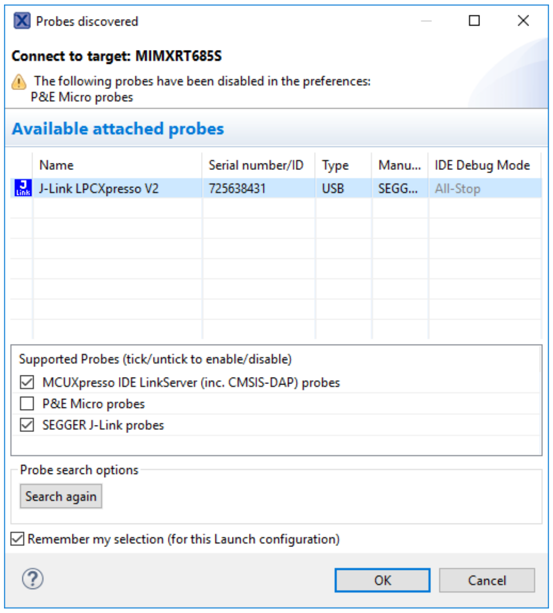


Figure 36 Quickstart Panel - Debug

5. A window will pop up to select debugger connects to the EVK. Select J-Link LPCXpresso V2 and press OK to continue.



Probes discovered

Connect to target: MIMXRT685S

The following probes have been disabled in the preferences:
P&E Micro probes

Available attached probes

Name	Serial number/ID	Type	Manu...	IDE Debug Mode
J-Link LPCXpresso V2	725638431	USB	SEGG...	All-Stop

Supported Probes (tick/untick to enable/disable)

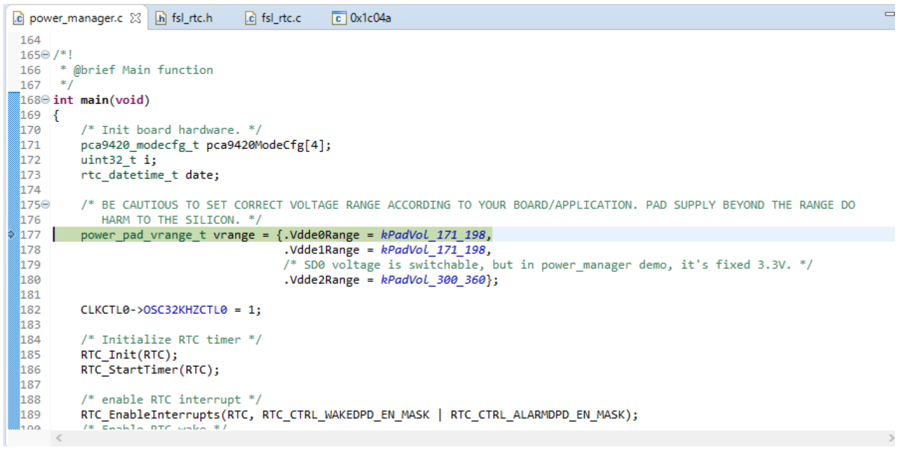
- [x] MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- [ ] P&E Micro probes
- [x] SEGGER J-Link probes

Probe search options

- [x] Remember my selection (for this Launch configuration)

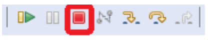
Figure 37 Selecting debugger

When it is successfully programmed, the program count will be halted at main() in the source window as shown below.



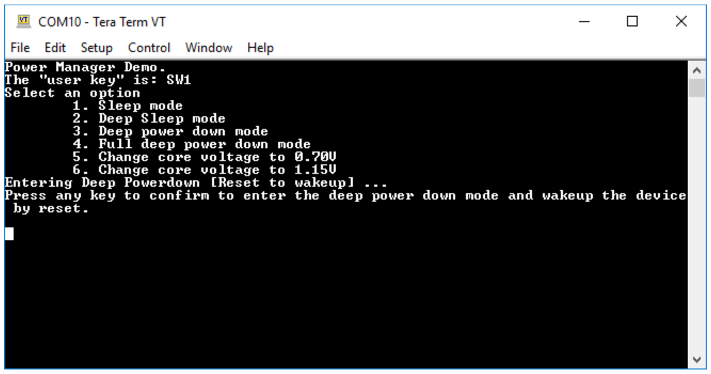
```
/*!
 * @brief Main function
 */
int main(void)
{
    /* Init board hardware. */
    pca9420_modecfg_t pca9420ModeCfg[4];
    uint32_t i;
    rtc_datetime_t date;

    /* BE CAUTIOUS TO SET CORRECT VOLTAGE RANGE ACCORDING TO YOUR BOARD/APPLICATION. PAD SUPPLY BEYOND THE RANGE DO
       HARM TO THE SILICON. */
    power_pad_vrange_t vrange = {.Vdde0Range = kPadVol_171_198,
                                 .Vdde1Range = kPadVol_171_198,
                                 /* SD0 voltage is switchable, but in power_manager demo, it's fixed 3.3V. */
                                 .Vdde2Range = kPadVol_300_360};

    CLKCTL0->OSC32KHZCTL0 = 1;

    /* Initialize RTC timer */
    RTC_Init(RTC);
    RTC_StartTimer(RTC);

    /* enable RTC interrupt */
    RTC_EnableInterrupts(RTC, RTC_CTRL_WAKEDPD_EN_MASK | RTC_CTRL_ALARMDPD_EN_MASK);
```

Figure 38 Program counter stops at main()

Terminate the debug program by pressing , then press the reset button on the board (see Figure 13).

Deep power down wake by deep sleep

6. After reset, selection message will show. Press 3 in the serial terminal to enter deep power down.



```
Power Manager Demo.
The "user key" is: SW1
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
Entering Deep Powerdown [Reset to wakeup] ...
Press any key to confirm to enter the deep power down mode and wakeup the device
 by reset.
```

Figure 39 Deep sleep

7. Manually reset the flash while in deep power down.
   Use a female to female jumper wire, one end connects to GND (see Figure 40), one end connects to male to male header.

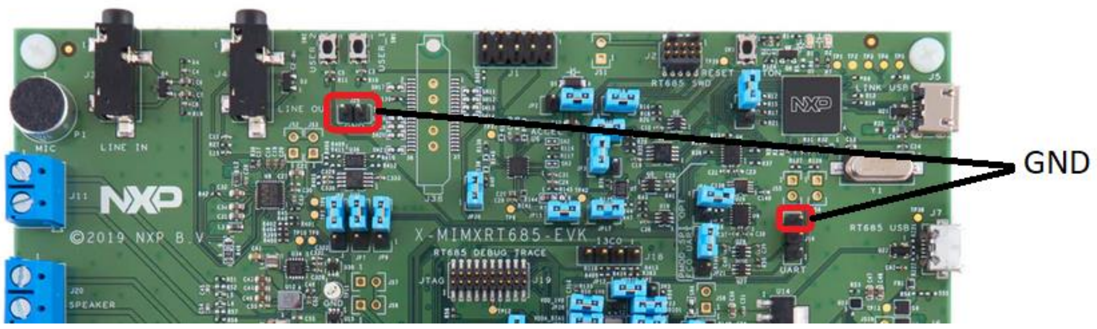


Figure 40 GND point

The reason is when RT685 wakes up by RTC from deep power down, the flash is still in previous mode (Quad or Octal). ROM will not be able to access flash content correctly; therefore, RT685 is not able to execute code from flash.

There is an GPIO pin connects to nRESET_OSPI of the reset pin on the SPI flash. User can configure the P2_12 to be asserted in the OTP. This lab chooses not to use OTP to configure the P2_12;incorrectly programming the OTP will cause unpredictable behavior.

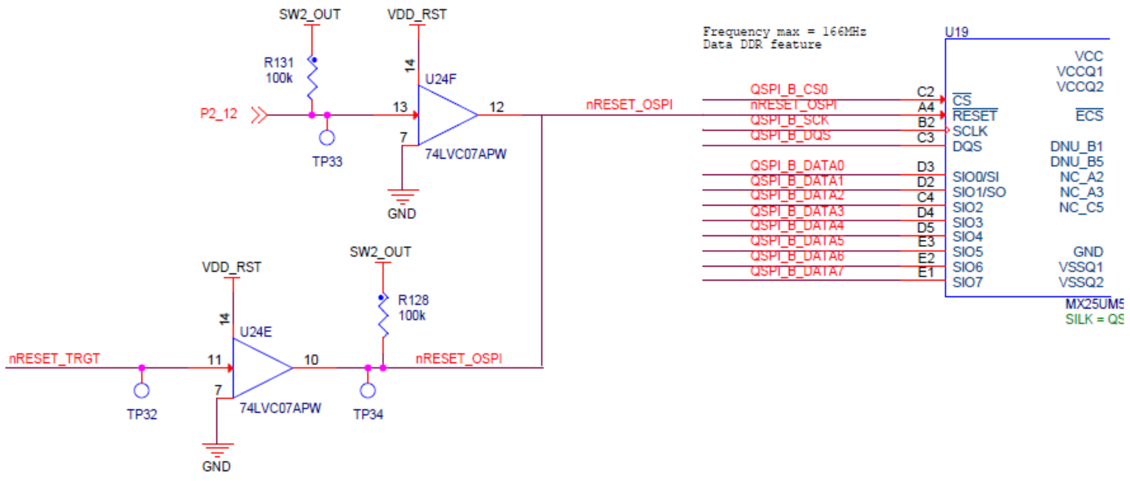


Figure 41 Flash reset pin

8. Locate TP34 or TP33 near the standard SD slot.

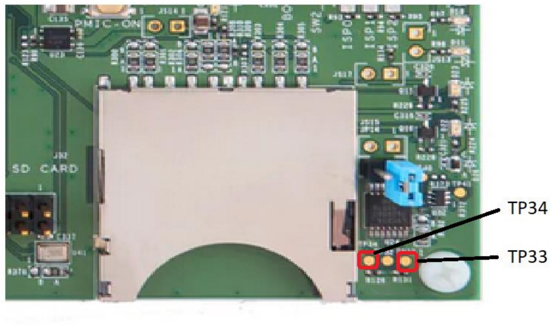
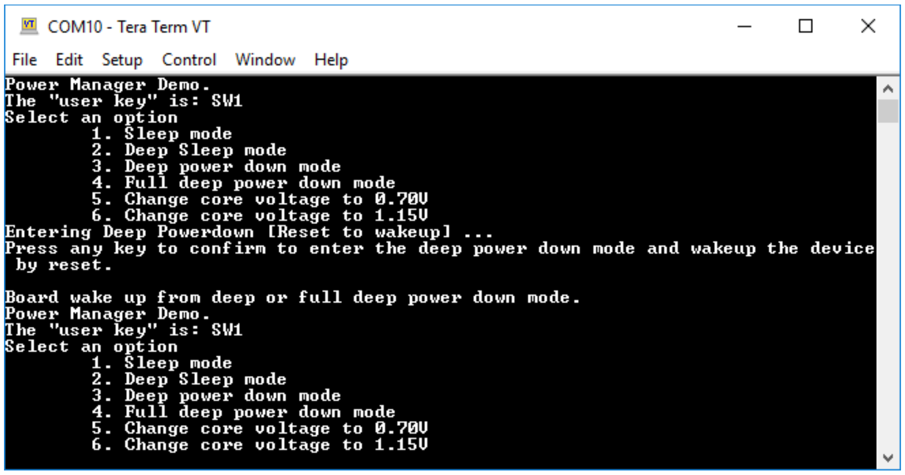


Figure 42 TP34

9. Press any key in the serial terminal to confirm to enter the deep power down mode.

10. Quickly use the female-to-female jumper wire with male header to touch TP33 or TP34, and untouch it.

11. When RT685 wake up by RTC and boot up, it will back to the selection.



```
Power Manager Demo.
The "user key" is: SW1
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
Entering Deep Powerdown [Reset to wakeup] ...
Press any key to confirm to enter the deep power down mode and wakeup the device
 by reset.

Board wake up from deep or full deep power down mode.
Power Manager Demo.
The "user key" is: SW1
Select an option
        1. Sleep mode
        2. Deep Sleep mode
        3. Deep power down mode
        4. Full deep power down mode
        5. Change core voltage to 0.70V
        6. Change core voltage to 1.15V
```

Figure 43 Wake and boot up

Deep power down wake by reset

12. Press 3 to enter deep power down.

13. Press any key to confirm in the serial terminal to enter the deep power down mode.

14. Press reset to wake from deep power down.

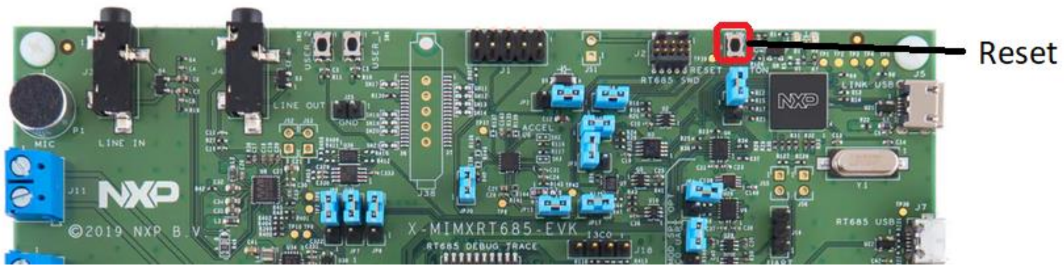


Figure 44 Reset button

Measuring power consumption on VDDCore JP29

Different multi-meter has different burden voltage when using the current measurement (mA/A and mA/uA). High end current measurement equipment provides wider range of burden voltage that will not draw current from the VDDCore. Handheld such as Fluke 83 is 0.03V in A, 1.8mV in mA and 100uV in uA, and the resistance between the input terminals is 0.03 ohm for A, 1.8 ohm for mA, and 100 ohm for uA. Assuming 40mA VDDCore is requirement in active mode, 40mA * 1.8mV = 74mV. The VDDCore is dropped below 1.0V (1.0V – 74mV = 926mV).

The solution is to raise the VDDCore voltage from 1.0V to 1.1V before changing the core frequency from FFRO to PLL. Enable SFRO after `BOARD_InitPins()` and before `BOARD_BootClockRUN()`. It is required for PMIC configuration before `BOARD_BootClockRUN()`.

```
void BOARD_ConfigPMICModes(pca9420_modecfg_t *cfg, uint32_t num)
{
    assert(cfg);

    /* Configuration PMIC mode to align with power lib like below:
     *  0b00    run mode, no special.
     *  0b01    deep sleep mode, vddcore 0.7V.
     *  0b10    deep powerdown mode, vddcore off.
     *  0b11    full deep powerdown mode vdd1v8 and vddcore off. */
    cfg[0].sw1OutVolt = kPCA9420_Sw1OutVolt1V100;
    …
}

int main(void)
{
    …

    BOARD_InitPins();
    /* Configure SFRO clock */
    POWER_DisablePD(kPDRUNCFG_PD_SFRO); /* Power on SFRO (16MHz) */
    CLOCK_EnableSfroClk();              /* Wait until SFRO stable */

    /* PMIC PCA9420 */
    BOARD_InitPmic();
    for (i = 0; i < ARRAY_SIZE(pca9420ModeCfg); i++)
    {
        PCA9420_GetDefaultModeConfig(&pca9420ModeCfg[i]);
    }
    BOARD_ConfigPMICModes(pca9420ModeCfg, ARRAY_SIZE(pca9420ModeCfg));
    PCA9420_WriteModeConfigs(&pca9420Handle, kPCA9420_Mode0, &pca9420ModeCfg[0],
ARRAY_SIZE(pca9420ModeCfg));

    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();

    CLOCK_EnableFfroClk(kCLOCK_Ffro60M);

    POWER_SetPadVolRange(&vrange);

    …

}
```