AN14332

MCXC444 Power Mode Switch Application

Rev. 1 — 9 July 2024 Application note

Document information

Information	Content
Keywords	AN14332, MCXC444, power modes, PMC, WFE
Abstract	This application note provides detailed information about each power mode and includes case examples in the SDK power mode switch example demo.

1 Introduction

The MCXC444 microcontroller family provides an ultra-low-power feature for the power sensitive market. This MCU family implements several low-power modes to meet this requirement. The application note provides detailed information about each power mode and includes case examples in the SDK power mode switch example demo. It also offers tips for using each of the power modes.

The MCUXpresso SDK provides users with robust peripheral drivers, stacks, middleware, and example applications designed to simplify and accelerate application development on any NXP MCU. The MCUXpresso SDK is complimentary and includes full source code under a permissive open source license for all hardware abstraction and peripheral driver software.

This application note focuses on the power management controller (PMC), system mode controller (SMC), Multipurpose Clock Generator Lite (MCG-Lite), and Low-Leakage Wakeup Unit (LLWU).

2 Power modes on MCXC444 MCU

This section provides details on the Arm Cortex-M architecture and MCXC444 MCU power modes.

2.1 Basic power modes in Arm Cortex-M0+ core

The Arm Cortex-M0+ uses the following basic power modes of the Arm Cortex-M architecture:

- RUN
- SLEEP mode: It stops the processor clock
- DEEP SLEEP mode: It stops the system clock and switches the PLL and flash memory

Note: *The Arm Cortex-M0+ processor SLEEP modes reduce power consumption.*

The system can generate spurious wakeup events. For example, a debug operation wakes up the processor. Therefore, software must be able to put the processor back into the SLEEP mode after such an event. A program can have an idle loop to put the processor back into SLEEP mode.

To enter the low-power modes (SLEEP/DEEP SLEEP), inform the processor using the following three instructions:

- *Wait For Interrupt (WFI)*: The WFI instruction causes immediate entry to the SLEEP mode. When the processor executes a WFI instruction, it stops executing instructions and enters the SLEEP mode.
- *Wait For Event (WFE)*: The WFE instruction causes entry to the SLEEP mode conditional on the value of a 1-bit event register (set by the SEV instruction). When the processor executes a WFE instruction, it checks the value of the event register as follows:
  - 0 = The processor stops executing instructions and enters the SLEEP mode.
  - 1 = The processor sets the register to 0 and continues executing instructions without entering the SLEEP mode.
- *Send Event (SEV)*: The SEV instruction causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register.

In the Arm Cortex-M0+ core, the SCB register controls the behavior of entering low-power modes after the WFI/WFE instruction.

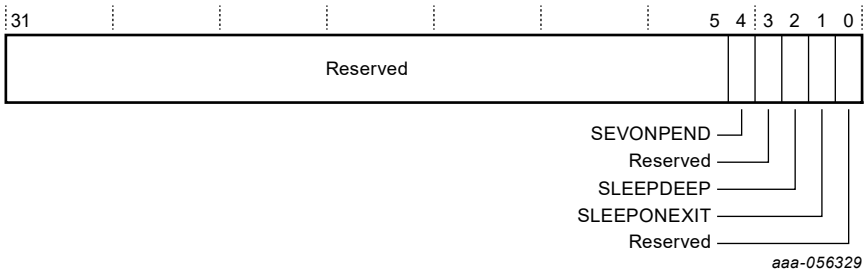

Figure 1. SCB register in Arm core

Figure 1 implies the following:

- `SCB[SLEEPDEEP]`: This bit controls whether the processor uses SLEEP mode or DEEP SLEEP mode as its low-power mode:
  - 0 = SLEEP
  - 1 = DEEP SLEEP
- `SCB[SLEEPONEXIT]`: This bit indicates sleep-on-exit when returning from Handler mode (interrupt handler) to Thread mode (the `main()` function):
  - 0 = Do not sleep when returning to Thread mode, go back to the `main()` function directly.
  - 1 = Enter the SLEEP or DEEP SLEEP again upon returning from an ISR to Thread mode, without going back to the `main()` function. Setting this bit to **1** enables an interrupt-driven application to avoid returning to an empty main application.
- `SCB[SEVONPEND]`: This bit send event on pending bit:
  - 0 = Only enabled interrupts or events can wake up the processor and disabled interrupts are excluded.
  - 1 = Enabled events and all interrupts, including disabled interrupts, can wake up the processor. When an event or interrupt becomes pending, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.

Here WFE, as a lightweight version of WFI, delays CPU execution without the need to restore and recover context, saving cycles during the low-power mode transitions.

The WFE instruction causes entries to SLEEP mode conditional on the value of a 1-bit event register. When the processor executes a WFE instruction, it checks the value of the event register:

- 0 = The processor stops executing instructions and enters the SLEEP mode.
- 1 = The processor sets the register to 0 and continues executing instructions without entering the SLEEP mode.

If the event register is "0", WFE suspends execution until one of the following events occurs:

- An exception, unless masked by the exception mask registers or the current priority level.
- An exception enters the pending state on setting the SEVONPEND in the System Control Register.
- A Debug Entry request event occurs on enabling the debug.
- An event signaled by a peripheral or another processor in a multiprocessor system using the SEV instruction.

2.2 Extend power modes in MCXC444

In the MCXC444, this core uses the WFI instruction to invoke SLEEP and DEEP SLEEP modes. It also extends the power modes and their relationship, as shown in Table 1.

Table 1. Power modes on MCXC444 MCU

Arm CM0+ power modes	MCXC444 MCU power modes	Wakeup module	Reset
RUN	RUN, VLPR	—	—
RUN	CPO	AWIC/NVIC	No
SLEEP	WAIT, VLPW	NVIC	No
DEEP SLEEP	STOP, VLPS	WIC	No
DEEP SLEEP	PSTOP1	AWIC	No
DEEP SLEEP	PSTOP2	AWIC/NVIC	No
DEEP SLEEP	LLS	LLWU	No
DEEP SLEEP	VLLSx (x=0/1/3)	LLWU	Yes

Wakeup modules in Table 1 imply the following:

- NVIC: Any interrupted source can wake up an MCU from WAIT/VLPW mode.
- AWIC: Only the AWIC wakeup source in the reference manual can wake up the MCU from STOP/VLPS mode.
- LLWU: Only the LLWU wakeup source in the reference manual can wake up the MCU from LLS/VLLSx modes. To wake up from VLLSx mode, go through a reset flow and call LLWU reset.
- For Compute Operation (CPO) mode, the Arm core is in the RUN mode. Any asynchronous interrupt and Arm core synchronous interrupt can wake up the MCU in the RUN mode.

Table 2 shows the detailed descriptions about each power mode.

Table 2. Power mode description

Mode	Description
RUN	The MCU can be run at full speed and the internal supply is fully regulated, that is, in run regulation. This mode is also referred to as normal RUN mode.
WAIT	• The core clock is gated off. • The system clock continues to operate. • Bus clocks, if enabled, continue to operate. • The run regulation is maintained.
STOP	• The core clock is gated off. • System clocks to other masters and bus clocks are gated off, after all stop acknowledge signals from supporting peripherals are valid.
VLPR	The core, system, bus, and flash clock maximum frequencies are restricted in this mode. For details about the maximum allowable frequencies, see the "Power Management" chapter in the *MCXC444 Sub-Family Reference Manual* (document MCXC444RM).
VLPW	• The core clock is gated off. • The system, bus, and flash clocks continue to operate, although their maximum frequency is restricted. For details about the maximum allowable frequencies, see the "Power Management" chapter in the *MCXC444 Sub-Family Reference Manual* (document MCXC444RM).
VLPS	• The core clock is gated off. • System clocks to other masters and bus clocks are gated off, after all stop acknowledge signals from supporting peripherals are valid.
LLS	• The core clock is gated off. • System clocks to other masters and bus clocks are gated off, after all stop acknowledge signals from supporting peripherals are valid.

Table 2. Power mode description*...continued*

Mode	Description
	• The MCU is placed in a low-leakage mode by reducing the voltage to internal logic. • All system RAM contents, internal logic, and I/O states are retained.
VLLS3	• The core clock is gated off. • System clocks to other masters and bus clocks are gated off, after all stop acknowledge signals from supporting peripherals are valid. • The MCU is placed in a low-leakage mode by powering down the internal logic. • All system RAM contents are retained and I/O states are held. • Internal logic states are not retained.
VLLS1	• The core clock is gated off. • System clocks to other masters and bus clocks are gated off, after all stop acknowledge signals from supporting peripherals are valid. • The MCU is placed in a low-leakage mode by powering down the internal logic and all system RAM. • I/O states are held. • Internal logic states are not retained.
VLLS0	• The core clock is gated off. • System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. • The MCU is placed in a low-leakage mode by powering down the internal logic and all system RAM. • I/O states are held. • Internal logic states are not retained. • The 1 kHz LPO clock is disabled and the power-on reset (POR) circuit can be optionally enabled using `STOPCTRL[PORPO]`.

For MCXC444 family devices, the NMI pin can wake up all power modes. If the bus clock does not filter the reset pin, the reset pin resets MCU power mode into default RUN mode.

Figure 2 shows the power mode state transitions available on the chip. Any reset always brings the MCU back to the normal RUN state.

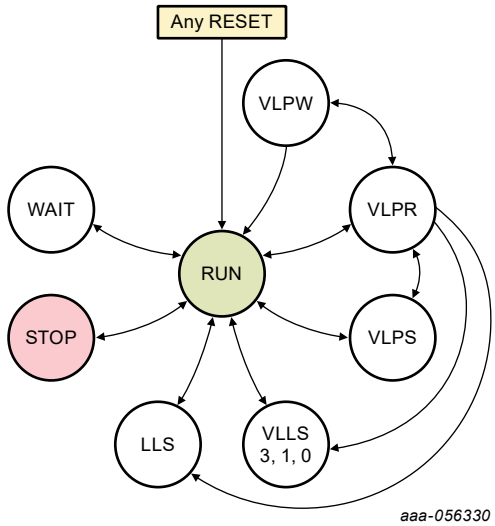

Figure 2. Power mode state transitions

To enter the target power mode from RUN/VLPR mode, perform the following steps:

1. To ignore the warning when the LDO reduces the power supply in ultra-low-power modes, disable the Low Voltage Detection (LVD) functions in the power management controller (PMC) module.
2. To save power in target low-power mode, disable the unnecessary modules/pins and to trigger the low-power exit event, set up the clock wakeup source module.
3. To ensure that the surviving module is still working with an available clock source, set up the clock source for target mode.
4. Unlock indicated power modes in the `SMC -> PMPROT` (Power Mode Protection) register, so that the target power mode can be entered.
5. Set up the target mode in the `SMC -> PMCTRL` (Power Mode Control) register and the `SMC > STOPCTRL` (Stop Control).
6. To invoke into the low-power mode, call the WFI or WFE.

To exit from the low-power mode, perform the following steps:

1. Wait for the wakeup event to trigger the pre-setup wakeup source.
2. For the VLLSx modes, LLWU is specially designed as a wakeup module to collect all available wakeup source.

For some wakeup routine from reset, clear the `PMC -> REGSC[ACKISO]` bit to unlock the port pins, which is locked and kept stable in some ultra-low-power modes.

Table 3 shows the module operation in the low-power modes (LLS and VLLSx).

Table 3. Module operation in low-power modes

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
Core modules						
NVIC	FF	FF	Static	Static	Static	OFF
System modules						
Mode controller	FF	FF	FF	FF	FF	FF
LLWU	Static	Static	Static	Static	FF	FF
Regulator	Low power	Low power	ON	Low power	Low power	• Low power in VLLS3 • OFF in VLLS0/1
Brownout detection	ON	ON	ON	ON	ON	• ON in VLLS1/3 • Optionally disabled in VLLS0
Clocks						
1 kHz LPO	ON	ON	ON	ON	ON	• ON in VLLS1/3 • OFF in VLLS0
System oscillator (OSC)	• OSCERCLK • Max of 16 MHz crystal	• OSCERCLK • Max of 16 MHz crystal	• OSCERCLK • Optional	• OSCERCLK • Max of 16 MHz crystal	OSCERCLK Max of 16 MHz crystal	• OSCERCLK • Max of 16 MHz crystal in VLLS1/3 • OFF in VLLS0
Memory and memory interfaces system modules						

Table 3. Module operation in low-power modes...*continued*

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
SRAM_U and SRAM_L	Low power	Low power	Low power	Low power	Low power	• Low power in VLLS3 • OFF in VLLS0/1
System register file	Powered	Powered	Powered	Powered	Powered	Powered
Timers						
LPTMR	FF	FF	• Async operation • FF in PSTOP2	Async operation	Async operation	Async operation
RTC	• FF • Async operation in CPO	FF	• Async operation • FF in PSTOP2	Async operation	Async operation	Async operation
Human-machine interfaces						
Segment LCD	• FF • Async operation in CPO	FF	• Async operation • FF in PSTOP2	Async operation	Async operation	• Async operation • OFF in VLLS0

3 Measuring the current in various power modes

In this document, an application software is designed to measure the current of the MCXC444 MCU while operating in various power modes. The FRDM-MCXC444 board is used as the main hardware platform. The two buttons on the board are used to switch the target power mode selection on the SLCD screen.

3.1 Board settings

FRDM-MCXC444 board has the measuring socket available in application.

JP1 is the expected measurement socket, as shown in Figure 3 and Figure 4.

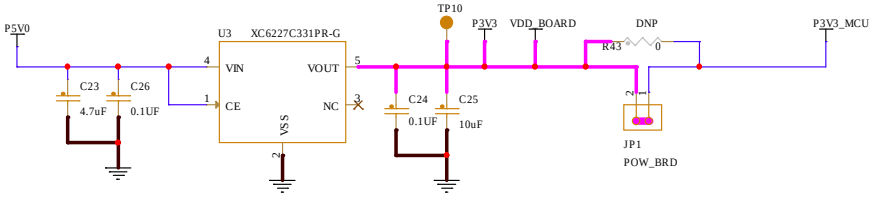

Figure 3. Schematic of measuring socket on FRDM-MCXC444 board

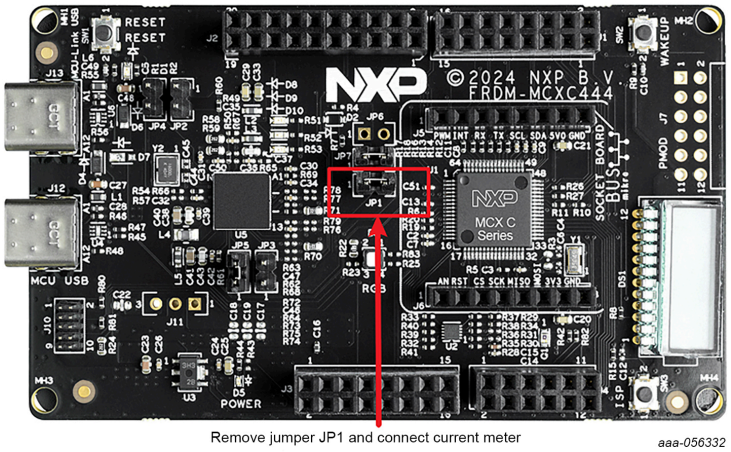

Figure 4. FRDM-MCXC444 measuring socket

The SLCD displays the current power mode during operation, as shown in Figure 5.

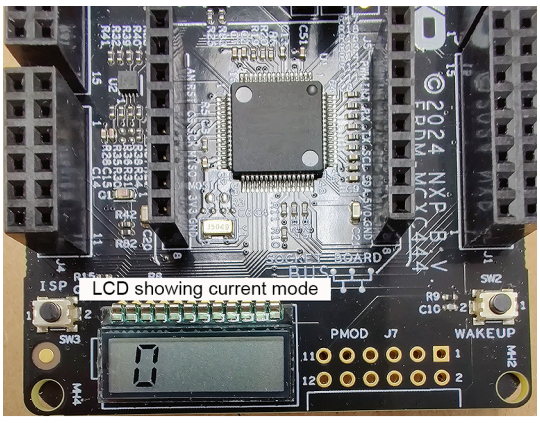

Figure 5. SLCD power mode display

3.2 Software design

MCUXpresso SDK software package provides the driver for SMC, PMC, LLWU, and clock modules. In the application software, these driver APIs can be used to operate the power modes with other peripheral drivers.

3.2.1 Switch power modes with software

In this application demo, a total of 10 power modes are covered as shown below:

```
/* Power mode definition used in application. */
typedef enum _app_power_mode
{
    kAPP_PowerModeRun, /* Normal RUN mode */
    kAPP_PowerModeWait, /* WAIT mode. */
    kAPP_PowerModeStop, /* STOP mode. */
    kAPP_PowerModeVlpr, /* VLPR mode. */
    kAPP_PowerModeVlpw, /* VLPW mode. */
    kAPP_PowerModeVlps, /* VLPS mode. */
    kAPP_PowerModeLls, /* LLS mode. */
    kAPP_PowerModeVlls0, /* VLLS0 mode. */
```

```
    kAPP_PowerModeVlls1, /* VLLS1 mode. */
    kAPP_PowerModeVlls3, /* VLLS3 mode. */
    kAPP_PowerModeMax
} app_power_mode_t;
```

- The `APP_PowerModeSwitch()` function is the most important function to execute the power mode switch.
- To control the target power mode, it uses the SMC API of the driver:

```
void APP_PowerModeSwitch(smc_power_state_t curPowerState, app_power_mode_t
targetPowerMode)
{
    smc_power_mode_vlls_config_t vlls_config;
    vlls_config.enablePorDetectInVlls0 = true;
    switch (targetPowerMode)
    {
        case kAPP_PowerModeVlpr:
            APP_SetClockVlpr(); /* setup the lower clock source for VLPR. */
            SMC_SetPowerModeVlpr(SMC);
            while (kSMC_PowerStateVlpr != SMC_GetPowerModeState(SMC))
            {
            }
            break;
        case kAPP_PowerModeRun:
            /* Power mode change. */
            SMC_SetPowerModeRun(SMC);
            while (kSMC_PowerStateRun != SMC_GetPowerModeState(SMC))
            {
            }
            /* If enter RUN from VLPR, change clock after the power mode
 change.
*/
            if (kSMC_PowerStateVlpr == curPowerState)
            {
               APP_SetClockRunFromVlpr(); /* setup the higher clock source for
RUN. */
             }
             break;
        case kAPP_PowerModeWait:
            SMC_PreEnterWaitModes();
            SMC_SetPowerModeWait(SMC);
            SMC_PostExitWaitModes();
            break;
        case kAPP_PowerModeStop:
            SMC_PreEnterStopModes();
            SMC_SetPowerModeStop(SMC, kSMC_PartialStop);
            SMC_PostExitStopModes();
            break;
        case kAPP_PowerModeVlpw:
            SMC_PreEnterWaitModes();
            SMC_SetPowerModeVlpw(SMC);
            SMC_PostExitWaitModes();
            break;
        case kAPP_PowerModeVlps:
            SMC_PreEnterStopModes();
            SMC_SetPowerModeVlps(SMC);
            SMC_PostExitStopModes();
            break;
        case kAPP_PowerModeLls:
            SMC_PreEnterStopModes();
```

```
            SMC_SetPowerModeLls(SMC);
            SMC_PostExitStopModes();
            break;
        case kAPP_PowerModeVlls0:
            vlls_config.subMode = kSMC_StopSub0;
            SMC_PreEnterStopModes();
            SMC_SetPowerModeVlls(SMC, &vlls_config);
            SMC_PostExitStopModes();
            break;
        case kAPP_PowerModeVlls1:
            vlls_config.subMode = kSMC_StopSub1;
            SMC_PreEnterStopModes();
            SMC_SetPowerModeVlls(SMC, &vlls_config);
            SMC_PostExitStopModes();
            break;
        case kAPP_PowerModeVlls3:
            vlls_config.subMode = kSMC_StopSub3;
            SMC_PreEnterStopModes();
            SMC_SetPowerModeVlls(SMC, &vlls_config);
            SMC_PostExitStopModes();
            break;
        default:
            break;
    }
}
```

- Entering a new power mode is similar to switching to a new task with a new working condition in the OS.
- To close the current mode and prepare for the new mode, perform the required operations before entering. In the new mode, perform the required initial work.
- Therefore, in the application demo code, the functions of `APP_PowerPreSwitchHook()` and `APP_PowerPostSwitchHook()` are created to pack these operations.
- To minimize power consumption in the MCU, the unnecessary peripherals are disabled before entering the WAIT/STOP modes, and recovered after wakeup.
- In the application demo, the UART peripheral for terminal interaction, and the output pins are disabled in low-power modes.
- During the MCU in SLEEP mode, only the SLCD driven by the OSC32 clock and the NVIC/AWIC are still alive.

3.2.2 Preserve SRAM contents in low-power modes

To show if SRAM contents can be preserved in VLLSx modes, a variable with software token written inside indicates whether the content is lost or not.

1. In the application demo, write a token `APP_SRAM_PRESERVE_FLAG` to the variable of `s_app_persist_flag` before entering VLLSx modes.
2. Read it after the MCU wakes up again.
3. When the MCU is wakened up from reset, the software reads the token variable and compares it with the expected value.
4. The user is informed via the display on the SLCD screen and UART terminal.

```
#define APP_SRAM_PRESERVE_FLAG 0x55555555

static volatile uint32_t s_app_persist_flag __attribute__((section(".noinit")));

...

/*!
```

```
 * @brief main demo function.
 */
int main(void) {

...

    if (kRCM_SourceWakeup & RCM_GetPreviousResetSources(RCM)) /* Wakeup from
 VLLS. */
    {
        PRINTF("\r\nMCU wakeup from VLLS modes...\r\n");

        if (s_app_persist_flag == APP_SRAM_PRESERVE_FLAG) {
            PRINTF("SRAM content preserved...\r\n");
            SLCD_Engine_Show_Num(&s_lcdEngine, 1, 2, true);
        } else {
            PRINTF("SRAM content not preserved...\r\n");
            SLCD_Engine_Show_Num(&s_lcdEngine, 0, 2, true);
        }
    }
...
```

3.3 Run application demo project

After building the project and downloading the image to the FRDM-MCXC444 board, run the application demo to measure the working current in different power modes. The UART terminal can output the log information.

1. Consider the multimeter. As mentioned previously, put the multimeter (in current measurement mode) into the series connection of the J20.
2. Connect the onboard debugger to the PC and open the terminal tool for UART communication with the configuration shown in Figure 6:
   - Baud rate = 9600
   - Data = 8 bit
   - Parity = None

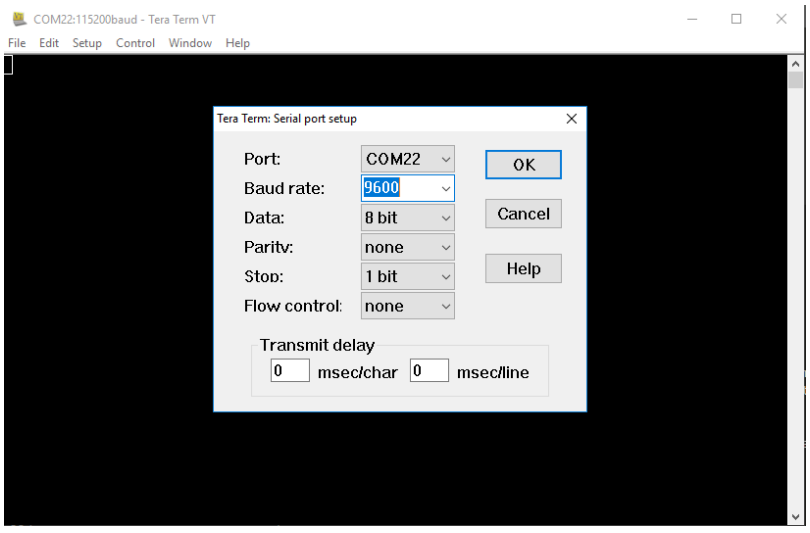

Figure 6. UART terminal settings

3. Now, the application demo is running.
4. The initial power mode is RUN. For the RUN mode, the SLCD shows "0", as shown in Figure 7.

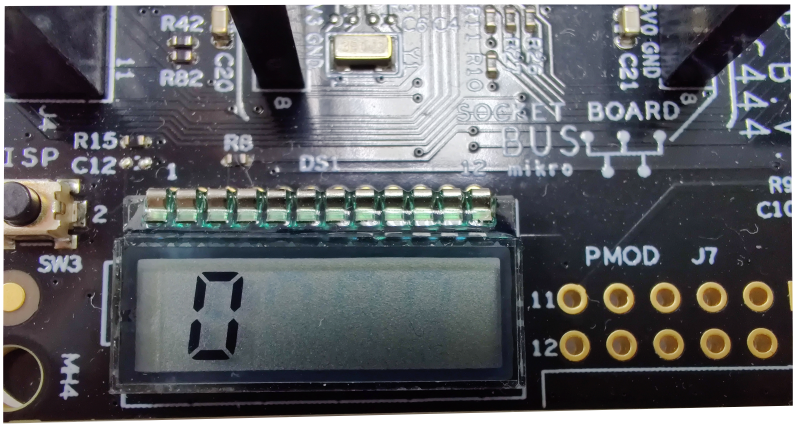

Figure 7. RUN mode

5. The UART terminal also shows the menu of the power mode selection, as shown in Figure 8.

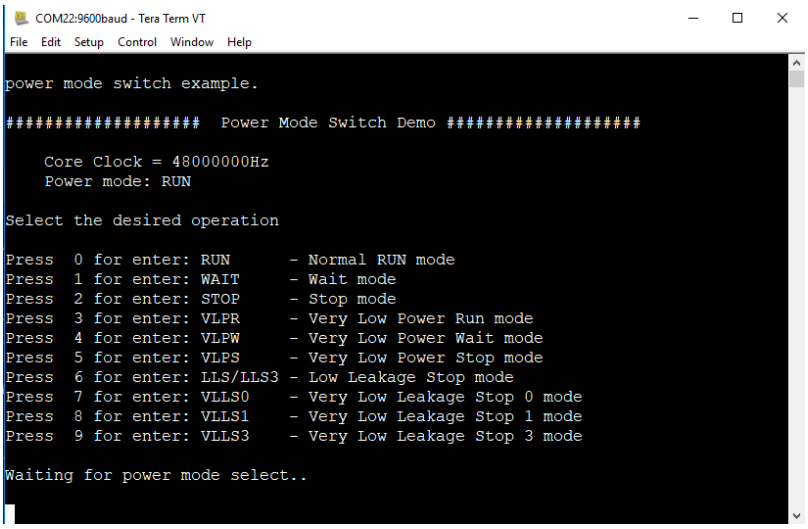

Figure 8. Power mode selection on UART terminal

6. Select the target power mode by typing "0" to "9" in the UART terminal. Most power modes support waking up either by using the LPTMR or **SW2** button event. The target power mode is activated after selecting the desired wakeup source. Figure 9 shows the SLCD displaying the current power mode.

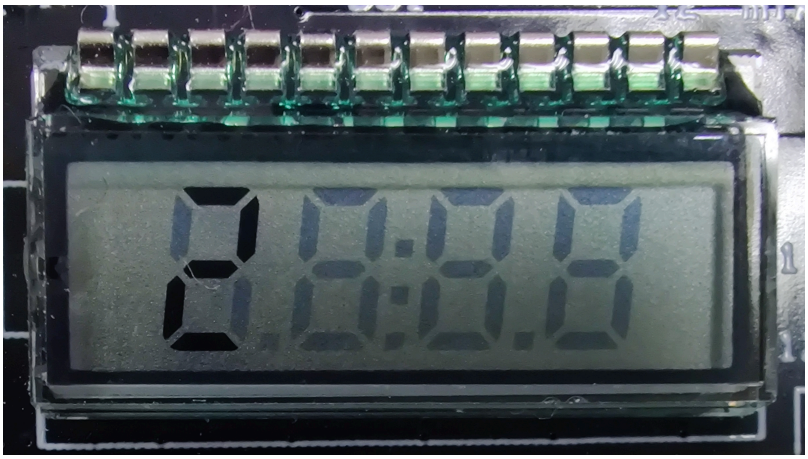

Figure 9. STOP mode

7. To wake up the MCU, press **SW2** or wait until LPTMR expires. Then the number on the SLCD changes to "0" or "3", depending on the previous power mode, as shown in Figure 10.

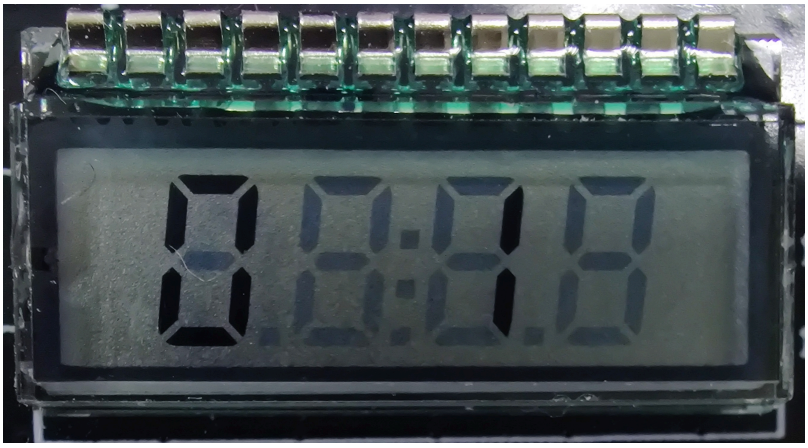

Figure 10. RUN mode with software token available

Figure 10 implies the following:

- "0" on the left means that the MCU returns to the RUN mode. It is "3" when the MCU wakes up from VLPS/VLPW to VLPR.
- "1" on the right means that the SRAM content is preserved. It is "0" when the SRAM is powered down and content is lost.
- In VLLS0 mode, the SLCD peripheral is powered down, therefore no LCD display in this mode. The LCD resumes functioning after wakeup.

8. The MCU returns to the RUN mode or VLPR mode.
9. Under VLLSx modes, the MCU wakes up from the reset handler.
10. To check whether the SRAM content is preserved in these modes, the software token is checked:
    - If the SRAM content is preserved (the memory token is valid), the number "1" is displayed next to the current power mode.
    - If the SRAM content is lost, the number "0" is displayed.

During the operations to switch power modes, users can read the current value on the multimeter, showing the real-time power consumption.

4 Conclusion

When running this application demo, the power consumption condition of MCXC444 on the FRDM-MCXC444 board is measured. Table 4 displays all the measuring values.

Note: *Even in the low-power modes, the LLWU, SLCD with OSC32 clock source are still active, as they are designed for low-power usage.*

Table 4. Power consumption in various power modes of MCXC444

Power Mode	VDD_I	Memory kept	Comment
RUN	8.36 mA	Yes	• 48 MHz CORE clock • UART enabled
WAIT	3.26 mA	Yes	• CORE SLEEP • UART disabled • NVIC wakeup
STOP	0.16 mA	Yes	• CORE DEEP SLEEP • UART disabled • AWIC wakeup
VLPR	0.91 mA	Yes	• 2 MHz CORE clock • UART enabled
VLPW	0.09 mA	Yes	• CORE SLEEP • UART disabled • NVIC wakeup
VLPS	3.9 µA	Yes	• CORE DEEP SLEEP • UART disabled • AWIC wakeup
LLS	3.4 µA	Yes	• CORE DEEP SLEEP • UART disabled • LLWU wakeup • SLCD and OSC32 enabled
VLLS0	0.5 µA	No	• CORE DEEP SLEEP • UART disabled • LLWU wakeup • SLCD and OSC32 disabled
VLLS1	1.9 µA	No	• CORE DEEP SLEEP • UART disabled • LLWU wakeup • SLCD and OSC32 enabled
VLLS3	2.9 µA	Yes	• CORE DEEP SLEEP • UART disabled • LLWU wakeup • SLCD and OSC32 enabled

Note: *The board power consumption results are informative only. For MCU power consumption data, refer to the device datasheet.*

5 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6 Revision history

Table 5 summarizes the revisions to this document.

Table 5. Revision history

Document ID	Release date	Description
AN14332 v.1.0	9 July 2024	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents