NXP Semiconductors

Power Management: Introduction to LLS and VLLS modes

By: Technical Information Center

The power consumption of devices and the implications around designing for low power are common topics nowadays. Kinetis microcontroller family includes internal power management features that can be used to control the microcontroller's power usage and assist reaching the targets of embedded designs.

This document is focused on lowest power modes in Kinetis Family: LLS and VLLS modes. Topics such as features, differences and examples are provided. It uses FRDM-KL26Z as its main target, however, same principle applies to any other Kinetis family such as K and L.

1 Introduction.

Typical power modes in legacy cores and other embedded systems are Run, Wait, and Stop. ARM® Cortex M0+ and M4 power modes are Run, Sleep and Deep Sleep. Kinetis MCUs have extended power modes that offer a variety of power management. Their relationship to typical embedded system power modes and Cortex M4 and M0+ is depicted in next figure.

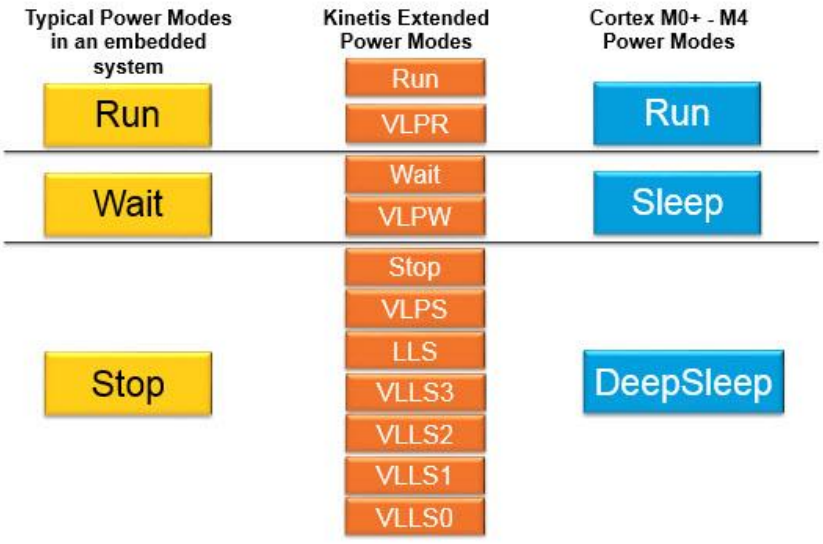


Figure 1 Power Modes comparison

On Kinetis, core uses the ***wake from interrupt (WFI)*** instruction to invoke Sleep and Deep Sleep modes. When entering sleep mode, Nested Vectored Interrupt Controller (NVIC) logic remains active and interrupts or a reset wake the core from sleep. When entering Deep Sleep mode, an Asynchronous Wakeup Interrupt Controller (AWIC) is used to wake the MCU from a selected group of sources. This Wakeup Interrupt Controller is enabled only when the DEEPSLEEP bit in the System Control Register (SCR) is set to 1. This SCR register is located in the System Control Block (SCB) of the cortex processor. Next figure illustrates the ARM Cortex Module structure for power management.

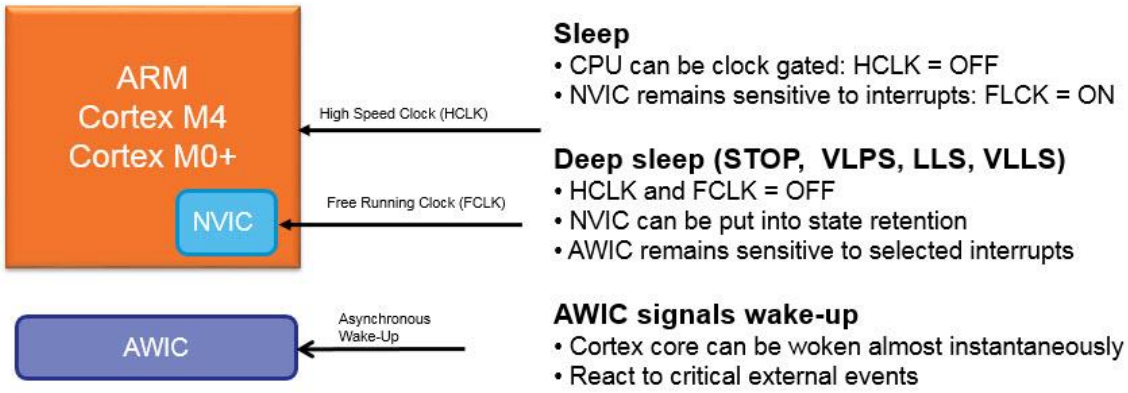


Figure 2 ARM Cortex M4 and M0+ Power modes

In Kinetis family, there is a special module that interacts with Cortex's AWIC module. It is called Low Leakage Wake-Up (LLWU) and it allows the user to select up to 16 eternal pin sources and up to 8 internal modules as a wake-up sources from low-leakage power modes. Next section introduces these low-leakage power modes and differences between each other.

2 Lowest power consumption modes: LLS and VLLS

Although Kinetis family has a wide variety of low power modes, this document is focused on lowest consumption modes: Low Leakage Stop (LLS) and Very Low Leakage Stop (VLLS). VLLS has some variants that mainly differ on which SRAM block is being powered.

- ***Low Leakage Stop (LLS)***: It is a power mode where most peripherals are in state retention mode (with clocks stopped) but OSC, LPTMR, RTC, CMP and TSI can be used[1]. As its core mode is Deep Sleep mode, NVIC is disabled (in static retention mode) and Low Leakage Wakeup Unit (LLWU) is used to wake-up. All SRAM is operating (Content is retained and I/O states are held).[2]
- ***Very Low Leakage Stop3 (VLLS3):*** Most peripherals are disabled (with clocks stopped) but OSC, LPTMR, RTC, CMP and TSI can be used. NVIC is disabled and LLWU is used to wake up. Upper SRAM (SRAM_U) and Lower SRAM (SRAM_L) remain powered on.
- ***Very Low Leakage Stop2 (VLLS2):*** Most peripherals are disabled (with clocks stopped) but LPTMR, RTC, CMP and TSI can be used. NVIC is disabled and LLWU is used to wake up. SRAM_L is powered off. A portion of SRAM_U remains powered on.
- ***Very Low Leakage Stop1 (VLLS1):*** Most peripherals are disabled (with clocks stopped) but LPTMR, RTC, CMP and TSI can be used. NVIC is disabled and LLWU is used to wake up. All SRAM is powered off. The 32-byte system register file and the 128-byte VBAT register file remain powered for customer-critical data.
- ***Very Low Leakage Stop0 (VLLS0):*** Most peripherals are disabled (with clocks stopped) but RTC can be used. NVIC is disabled and LLWU is used to wake up. All SRAM is powered off. The 32-byte system register file and the 128-byte VBAT register file remain powered for customer-critical data. The 1 kHz LPO clock is disabled and the Power-On Reset (POR) circuit can be optionally enabled.

Following figure shows the allowed power mode transitions. As LLS and VLLSx modes are the lowest power Stop modes, they should be selected based on the amount of logic or memory that is required to be retained by the application.

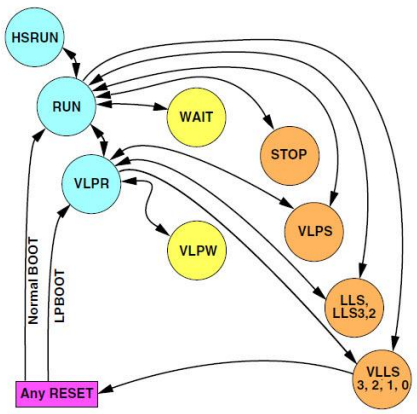


Figure 3 - Power Mode Transitions

[1] For detailed information, consult ***Module operation in low power modes*** table for specific chip's reference manual.
[2] Larger Memories M4 cores added LLS2 and LLS3 modes.

Next section illustrates the steps needed to enter LLS and VLLSx modes for basic bare metal implementation. Both examples use external wake-up sources and use on-board GPIO to differentiate between RUN and LLS/VLLSx modes.

For further information on Power modes for Kinetis family please refer to Reference section.

3 LLS implementation on Kinetis Design Studio

This section lists steps needed to create a basic bare board example for FRDM-KL26Z. This basic application uses one external interrupt from PTA5 to send MCU to LLS mode and it uses SW1 (PTC3) to wake MCU up.

3.1 As NVIC is disabled in LLS mode, LLWU is used to wake CPU up (through WIC module). This module can be configured to use up to 16 external sources and/or 8 internal sources. These sources are chip-specific and can be consulted on Chip's reference manual. Following table shows all available sources for KL26Z chip.

Table 3-15. Wakeup Sources

	LLWU pin	Module source or pin name
External	LLWU_P5	PTB0
	LLWU_P6	PTC1
	LLWU_P7	PTC3
	LLWU_P8	PTC4
	LLWU_P9	PTC5
	LLWU_P10	PTC6
	LLWU_P14	PTD4
	LLWU_P15	PTD6
Internal	LLWU_M0IF	LPTMR0
	LLWU_M1IF	CMP0
	LLWU_M2IF	Reserved
	LLWU_M3IF	Reserved
	LLWU_M4IF	TSI0
	LLWU_M5IF	RTC Alarm
	LLWU_M6IF	Reserved
	LLWU_M7IF	RTC Seconds

Figure 4 Available Wake-up Sources on LLWU

In this example PTC3 is used, so LLWU_P7 will be configured to generate a wake-up event.

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE7		WUPE6		WUPE5		WUPE4	
Reset	0	0	0	0	0	0	0	0

LLWU_PE2 field

Enables and configures the edge detection for the wakeup pin.

00 External input pin disabled as wakeup input
01 External input pin enabled with rising edge detection
10 External input pin enabled with falling edge detection
11 External input pin enabled with any change detection

Figure 5 Configure external source in LLWU

3.2 Allow LLS in Power Mode Protection Register.

Power Mode Protection register provides protection for entry into any low-power run or stop mode. This register (SMC_PMPROT) can be written only once after any system reset. To enable Low-Leakage Stop mode it is necessary to write '1' to ALLS field (bit 3).

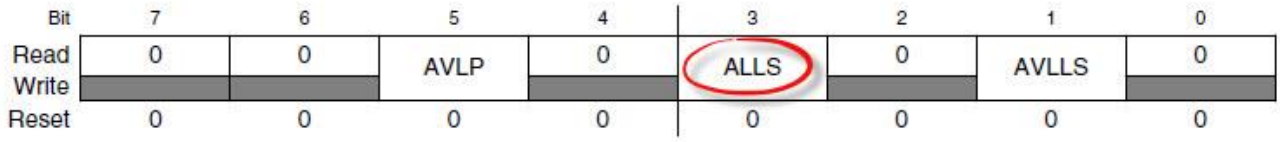


Figure 6 Allow LLS in SMC_PMPROT register

3.3 In LLS mode, LLWU interrupt must be enabled in order to get a successful wake-up event. In LLWU's Interrupt Service Routine (ISR), do not forget to clear the flag: LLWU_F1 and LLWU_F2 register for external events and/or LLWU_F3 for internal module's events.

3.4 Before entering to LLS mode, you can disable all unused pins to reduce power consumption by every pin.

3.5 To enter in LLS mode, SMC_PMCTRL register should be written. For LLS, 0b011 must be set to STOPM field. DeepSleep bit at SCR register (SCB_SCR) must be written to enter in deep sleep mode, at the same time, a WFI assembler command must be called to effectively enter in LLS mode.

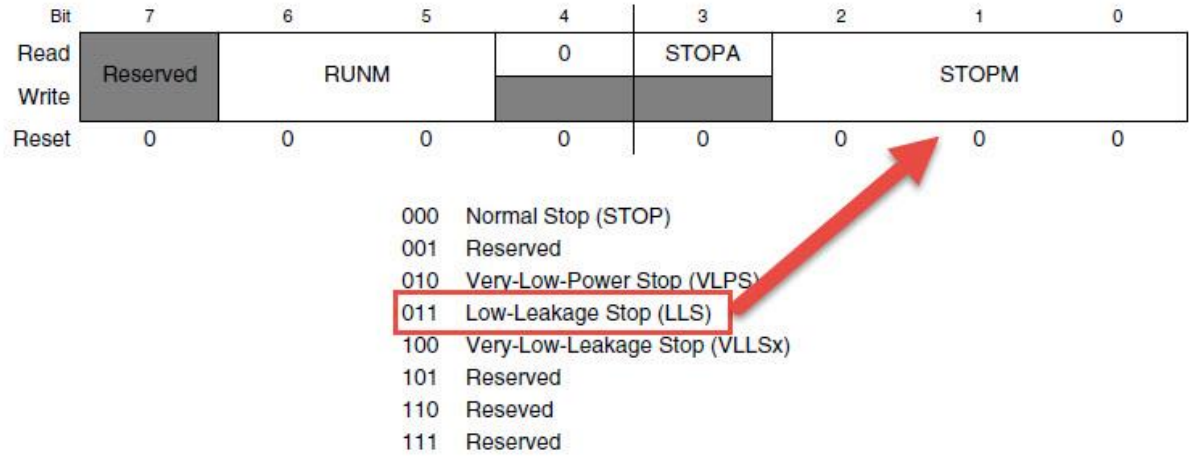


Figure 7 Enter to LLS mode

3.6 MCU will stay in LLS mode until a wake-up event is triggered on LLWU module. Once this event is received, MCU is waken-up and it is ready to attend LLWU's ISR, then, it continues with normal execution code. Appendix A shows the full main source file for this example.

4 VLLS implementation on Kinetis Design Studio

This section lists steps needed to create a basic bare board example for FRDM-KL26Z. This basic application uses one external interrupt from PTA5 to send MCU to VLLS mode and it uses SW1 (PTC3) to wake MCU up.

4.1 Since VLLS modes return to RUN (or VLPR) through RESET event, it is necessary to release I/O pins and certain peripherals to their normal state, it is done by checking ACKISO bit flag (found in PMC_REGSC register), if it is set, then, it is necessary to write '1' to release I/O and peripherals.

4.2 Verify that previous reset event was caused due any wake-up source from LLWU. The WAKEUP flag in RCM_SRS0 register will be set when a reset has been caused by an enabled LLWU module wakeup source while the chip was in a Low leakage mode.[3]

4.3 As NVIC is disabled in VLLS mode, LLWU is used to wake CPU up (through WIC module). Configure the wakeup sources that you prefer (check section 3.1 for detailed information). In this example, same External source is used (PTC3 or LLWU_P7)

4.4 Allow VLLS in Power Mode Protection Register.

Power Mode Protection register provides protection for entry into any low-power run or stop mode. This register (SMC_PMPROT) can be written only once after any system reset. To enable Very Low-Leakage Stop mode it is necessary to write '1' to AVLLS field (bit 1).

Bit	7	6	5	4	3	2	1	0
Read	0	0	AVLP	0	ALLS	0	AVLLS	0
Write								
Reset	0	0	0	0	0	0	0	0

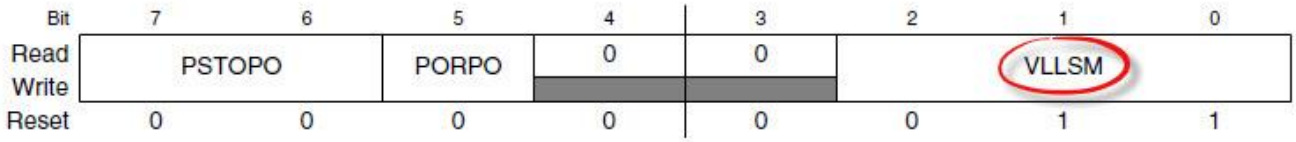
Figure 8 - Allow VLLS modes

[3] If external RESET pin causes the reset event when MCU was in VLLS mode, WAKEUP flag will be set along with PIN flag in RCM_SRS0 register.

4.5 LLWU interrupt can be enabled and it will be attended after the RESET event is completed. In LLWU's Interrupt Service Routine (ISR), do not forget to clear the flag: LLWU_F1 and LLWU_F2 register for external events and/or LLWU_F3 for internal module's events.[4]

4.6 Before entering to VLLS mode, you can disable all unused pins to reduce power consumption by every pin.

4.7 Before entering in VLLS mode, it is needed to specify which VLLS mode will be used. In SMC_STOPCTRL register select the VLLS mode that you desire.



VLLS Mode Control

This field controls which VLLS sub-mode to enter if STOPM=VLLSx.

000 VLLS0
001 VLLS1
010 Reserved
011 VLLS3
100 Reserved
101 Reserved
110 Reserved
111 Reserved

Figure 9 – Selecting VLLS Mode

4.8 Now, Select the VLLS mode at SMC_PMCTRL register.

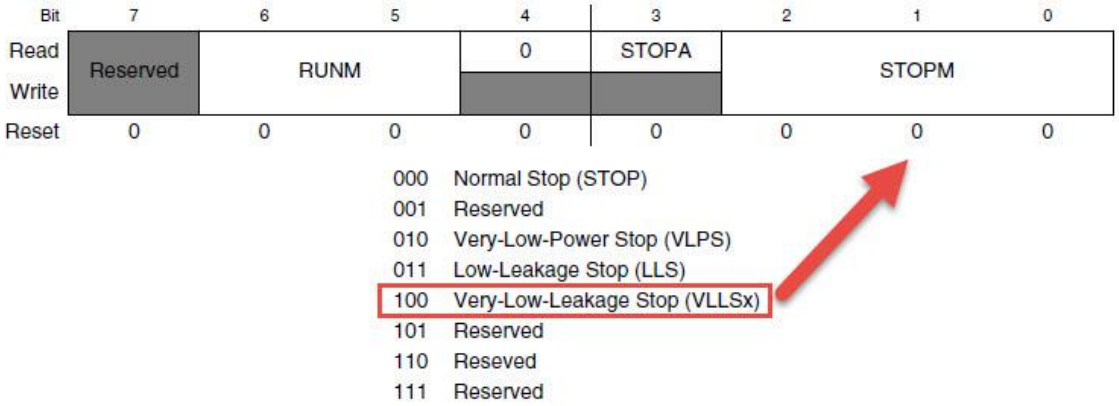


Figure 10 - SMC_PMCTRL register for setting VLLS

[4] When using internal modules as wake-up source, you can clear module's flag (For example: RTC's flag that causes the interrupt) from their appropriate register besides clearing LLWU's flag as well.

4.9 MCU is ready to enter to VLLS mode (be sure to set DeepSleep bit at SCR register (SCB_SCR) and call WFI assembler instruction). Now it will stay in VLLS mode until a wake-up event is triggered on LLWU module. This wake-up event will trigger a RESET event and MCU will attend LLWU's handlers after MCU is fully restored. Appendix B shows the full main source for this example.

As VLLSx modes wake the MCU up through RESET event, it sometimes will be necessary to save some critical data for the application. In this case, it is necessary to save this data in SRAM's location that is still being powered. There is also the possibility to save this critical data in the System Register File area, which is a 32-byte location that is always powered even in VLLS0 mode (where all SRAM is powered off). Next chapter shows a basic example on how to store data in this memory area.

5 Saving data in System Register File area

Some Kinetis devices include a 32-byte register file that is powered in all power modes. Besides of this register, some other devices include also the VBAT register file that is powered via VBAT.

This system register file retains its contents during Low-voltage detect (LVD) events and it is only reset during a Power-On Reset.

For KL26Z, this system register file's base address is 0x4004_1000 and it can be seen in MCU's reference manual Chapter 4 Memory Map, where a memory map table for Peripheral Bridge is shown:

Table 4-2. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4003_C000	60	—
0x4003_D000	61	Real-time clock (RTC)
0x4003_E000	62	—
0x4003_F000	63	DAC0
0x4004_0000	64	Low-power timer (LPTMR)
0x4004_1000	65	System register file
0x4004_2000	66	—
0x4004_3000	67	—
0x4004_4000	68	—
0x4004_5000	69	Touch sense interface (TSI)
0x4004_6000	70	—
0x4004_7000	71	SIM low-power logic
0x4004_8000	72	System integration module (SIM)
0x4004_9000	73	Port A multiplexing control
0x4004_A000	74	Port B multiplexing control
0x4004_B000	75	Port C multiplexing control
0x4004_C000	76	Port D multiplexing control
0x4004_D000	77	Port E multiplexing control
0x4004_E000	78	—
0x4004_F000	79	—

Figure 11 - System Register File location

Next snipped code shows a basic implementation that counts the times that MCU has entered to VLLS modes. As this address location will retain its value, every time that MCU enters in VLLS mode, counter will increment. Every time that MCU is waken up, this counter value will be read, when this counter is multiple of 5, Green LED will be turned on.

Appendix C shows this basic implementation.

6 Prepare FRDM-KL26Z

6.1 Before loading the examples in FRDM-KL26Z it is necessary to remove some components and add another ones. J1 as well as J3 connectors should be added in order to use PTA5 pin.

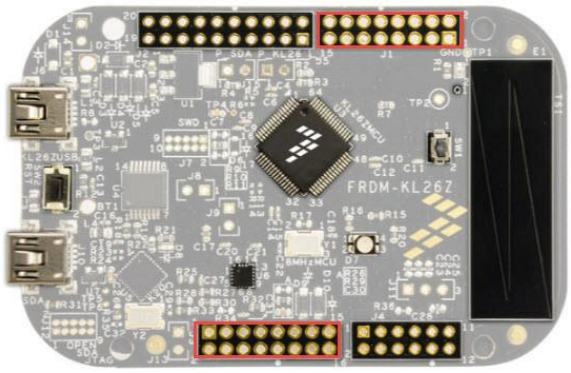


Figure 12 - Populate J1 and J3

6.2 To measure an estimated current consumption for KL26Z MCU, R3 and R2 should be removed, after this, add J5 jumper as shown below:

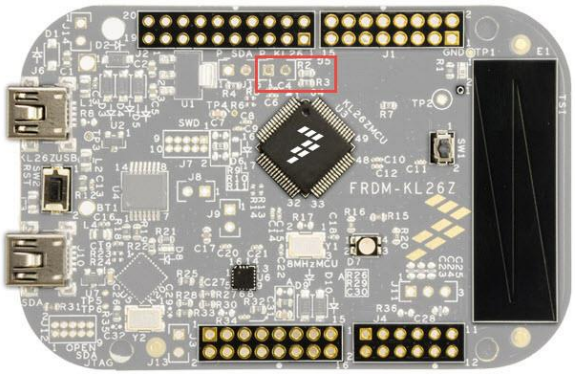


Figure 13 - Add J5. Remove R2 and R3

6.3 Connect an external switch between J1_12 and J3_14. This SW is used to enter into LLS/VLLS mode.

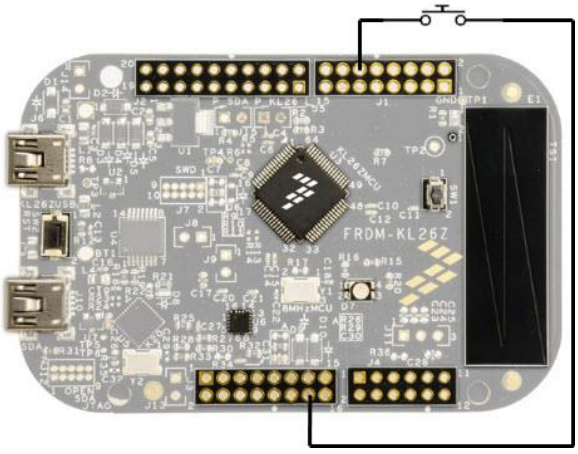


Figure 14 - Add push-button on PTA5

6.4 Connect an ammeter on J5's terminal.

7 Running LLS example

7.1 Open KDS and import FRDM-KL26Z_LLS_KDS project.

7.2 Compile and verify that no errors are shown.

7.3 Download the project in FRDM-KL26Z (Rev B).

7.4 Disconnect from debug session. Reset the board by pressing on board SW2.

7.5 Measure MCU's current consumption, it will be close to 4mA.

7.6 Press external push-button to enter LLS mode. Red LED will be turned on.

7.7 Measure MCU's current consumption, it will be close to 3uA.

7.8 Press on board SW1 to wake MCU up. Green LED will be turned on (Red LED will be turned off) and current consumption will increase to 4 mA again.

7.9 Repeat the process as needed.

8 Running VLLS example

8.1 Open KDS and import FRDM-KL26Z_VLLSx_KDS project.

8.2 If System Register File functionality is desired, set macro TEST_SYSTEM_REGISTER_FILE to 1, otherwise, set it to zero.

```
#define TEST_SYSTEM_REGISTER_FILE    0
```

8.3 In SMC.h file define the VLLS mode that you prefer by setting `VLLS_MODE` macro to *`eVLLS0Mode`*, *`eVLLS1Mode`* or *`eVLLS3Mode`*.

8.4 Compile and verify that no errors are shown.

8.5 Download the project in FRDM-KL26Z (Rev B).

8.6 Disconnect from debug session. Reset the board by pressing on board SW2.

8.7 Measure MCU's current consumption, it will be close to 4mA.

8.8 Press external push-button to enter VLLS mode. Red LED will be turned on.

8.9 Measure MCU's current consumption, it will be close to 800nA.

8.10 Press on board SW1 to wake MCU up. Blue LED will be turned on (Red LED will be turned off) to signal that MCU has been reset by a wake-up source trigger, current consumption will increase to 4 mA again.

8.11 If System Register File functionality is enabled, after waking MCU up a multiple of 5 times, Green LED will be turned on instead of Blue LED.

8.12 Repeat the process as needed.

9 Reference

- AN4503: Power Management for Kinetis MCUs
- AN4470: Using Low Power modes on Kinetis Family

Appendix A: Source code for LLS implementation

- Main Source File

```
volatile uint8_t g_bGoToSleep = 0;
volatile uint32_t g_bWakeUpEvent = 0;

int main(void)
{
    /* Write your code here */
    /* Init Green and Red LEDs pins */
    GPIO_outputInit(ePortE, 29);
    GPIO_outputInit(ePortE, 31);
    /* Init SW1 and PTA5 as input pins */
    /* Enable PTC3 as input with internal pull up resistor */
    GPIO_inputInit(ePortC, 3, ePullUp, eIntDmaDisabled);
    /* Enable PTA5 as input, with internal pull-up and generate a falling edge interrupt */
    GPIO_inputInit(ePortA, 5, ePullUp, eInterruptFallingEdge);
    /* Turn Green LED off */
    GPIO_setOutput(ePortE, 31, eHigh);
    /* Turn Red LED off */
    GPIO_setOutput(ePortE, 29, eHigh);

    /* Enable interrupt for PORTA on NVIC module */
    NVIC_EnableIRQ(PORTA_IRQn);

    /* Configure LLWU */
    LLWU_init();
    /* Configure LLWU_P7 as falling edge wake-up source */
    LLWU_enableExternal(7, eExtFallingEdge);
    /* Enable interrupt for LLWU unit on NVIC module */
    NVIC_EnableIRQ(LLWU_IRQn);

    /* LLS is allowed */
    SMC_allowLLSMode();

    /* This for loop should be replaced. By default this loop allows a single stepping. */
    for (;;) {
        if (g_bGoToSleep)
        {
            g_bGoToSleep = 0;
            /* Turn Green LED off */
            GPIO_setOutput(ePortE, 31, eHigh);
            /* Turn Red LED on */
            GPIO_setOutput(ePortE, 29, eLow);

            /* Disable unused pins */
            GPIO_deInitPin(ePortE, 31); /* Green LED */
            GPIO_deInitPin(ePortA, 5); /* PTA5 (SW) is not used when already in LLS mode*/
            /* Specify that LLS is selected */
            SMC_setLLSMode();
        }
```

```
        if (g_bWakeUpEvent)
        {
            g_bWakeUpEvent = 0;
            /* Turn Red LED off */
            GPIO_setOutput(ePortE, 29, eHigh);
            /* Initialize pins when waking up */
            GPIO_inputInit(ePortA, 5, ePullUp, eInterruptFallingEdge);
            GPIO_outputInit(ePortE, 31);
            /* Turn Green LED on */
            GPIO_setOutput(ePortE, 31, eLow);
        }
    }
    /* Never leave main */
    return 0;
}

void PORTA_IRQHandler (void)
{
    if (PORTA->PCR[5] & PORT_PCR_ISF_MASK)
    {
        /* Clear interrupt flag */
        PORTA->PCR[5] |= PORT_PCR_ISF_MASK;
        g_bGoToSleep = 1;
    }
}

void LLWU_IRQHandler (void)
{
    if(LLWU_getExternalFlag(7))
    {
        /* Clear flag on LLWU_P7 */
        LLWU_clearExternalFlag(7);
        g_bWakeUpEvent = 1;
    }
}
```

Appendix B: Source code for VLLS implementation

- Main Source file

```
volatile uint8_t g_bGoToSleep = 0;

int main(void)
{
    uint16_t resetSource = 0;
    /* Write your code here */

    /* Release I/O pads in case reset was caused by waking up from VLLS modes */
    PMC_releaseIOpads();

    /* Init Blue and Red LEDs pins */
    GPIO_outputInit(ePortE, 29);
    GPIO_outputInit(ePortD, 5);

    /* Turn Red LED off */
    GPIO_setOutput(ePortE, 29, eHigh);
    /* Turn Blue LED off */
    GPIO_setOutput(ePortD, 5, eHigh);

    /* Check most recent reset source*/
    resetSource = RCM_getResetSource();
    if (resetSource & eWakeUpSource)
    {
        /* Recent reset was caused by waking for VLLSx modes */
        /* Turn Blue LED on */
        GPIO_setOutput(ePortD, 5, eLow);
    }
    /* Init SW1 and PTA5 as input pins */
    /* Enable PTC3 as input with internal pull up resistor */
    GPIO_inputInit(ePortC, 3, ePullUp, eIntDmaDisabled);
    /* Enable PTA5 as input, with internal pull-up and generate a
       falling edge interrupt */
    GPIO_inputInit(ePortA, 5, ePullUp, eInterruptFallingEdge);

    /* Enable interrupt for PORTA on NVIC module */
    NVIC_EnableIRQ(PORTA_IRQn);

    /* Configure LLWU */
    LLWU_init();
    /* Configure LLWU_P7 as falling edge wake-up source */
    LLWU_enableExternal(7, eExtFallingEdge);
    /* Enable interrupt for LLWU unit on NVIC module */
    NVIC_EnableIRQ(LLWU_IRQn);

    /* VLLS is allowed */
    SMC_allowVLLSMode();
```

```
    for (;;) {
        if (g_bGoToSleep)
        {
            g_bGoToSleep = 0;
            /* Turn Blue LED off */
            GPIO_setOutput(ePortD, 5, eHigh);
            /* Turn Red LED on */
            GPIO_setOutput(ePortE, 29, eLow);

            /* Disable unused pins */
            GPIO_deInitPin(ePortD, 5); /* Blue LED */
            GPIO_deInitPin(ePortA, 5); /* PTA5 (SW) is not used when
            already in VLLS mode*/
            /* Specify that VLLS is selected */
            SMC_setVLLSxMode();
        }
    }
    /* Never leave main */
    return 0;
}

void PORTA_IRQHandler (void)
{
    if (PORTA->PCR[5] & PORT_PCR_ISF_MASK)
    {
        /* Clear interrupt flag */
        PORTA->PCR[5] |= PORT_PCR_ISF_MASK;
        g_bGoToSleep = 1;
    }
}

void LLWU_IRQHandler (void)
{
    if(LLWU_getExternalFlag(7))
    {
        /* Clear flag on LLWU_P7 */
        LLWU_clearExternalFlag(7);
    }
}
```

Appendix C. System Register File Usage

```
/* Define System Register Address for this KL26 MCU */
#define SYSTEM_REGISTER_FILE_ADDR           0x40041000u

volatile uint32_t *systemRegPtr = (uint32_t *)SYSTEM_REGISTER_FILE_ADDR;

int main(void)
{
    uint16_t resetSource = 0;
    /* Write your code here */

    /* Release I/O pads in case reset was caused by waking up from VLLS modes */
    PMC_releaseIOpads();

    /* Init Blue and Red LEDs pins */
    GPIO_outputInit(ePortE, 29);
    GPIO_outputInit(ePortE, 31);
    GPIO_outputInit(ePortD, 5);

    /* Turn Red LED off */
    GPIO_setOutput(ePortE, 29, eHigh);
    /* Turn Green LED off */
    GPIO_setOutput(ePortE, 31, eHigh);
    /* Turn Blue LED off */
    GPIO_setOutput(ePortD, 5, eHigh);

    /* Check most recent reset source*/
    resetSource = RCM_getResetSource();
    if (resetSource & eWakeUpSource)
    {
        /* Recent reset was caused by waking for VLLSx modes */
        /* Turn Blue LED on */
        GPIO_setOutput(ePortD, 5, eLow);
    }
    else
    {
        /* Reset counter when reset event was not caused by being in VLLS */
        *systemRegPtr &= 0;
    }
    /* Init SW1 and PTA5 as input pins */
    /* Enable PTC3 as input with internal pull up resistor */
    GPIO_inputInit(ePortC, 3, ePullUp, eIntDmaDisabled);
    /* Enable PTA5 as input, with internal pull-up and generate a falling edge
interrupt */
    GPIO_inputInit(ePortA, 5, ePullUp, eInterruptFallingEdge);

    /* Enable interrupt for PORTA on NVIC module */
    NVIC_EnableIRQ(PORTA_IRQn);

    /* Configure LLWU */
    LLWU_init();
```

```
    /* Configure LLWU_P7 as falling edge wake-up source */
    LLWU_enableExternal(7, eExtFallingEdge);
    /* Enable interrupt for LLWU unit on NVIC module */
    NVIC_EnableIRQ(LLWU_IRQn);

    /* VLLS is allowed */
    SMC_allowVLLSMode();
    for (;;) {
        if (((*systemRegPtr % 5) == 0) &&
                (*systemRegPtr != 0))
        {
            /* Turn Blue LED off */
            GPIO_setOutput(ePortD, 5, eHigh);
            /* Turn Green LED off */
            GPIO_setOutput(ePortE, 31, eLow);
        }
      if (g_bGoToSleep)
        {
            g_bGoToSleep = 0;
            /* Turn Blue LED off */
            GPIO_setOutput(ePortD, 5, eHigh);
            /* Turn Green LED off */
            GPIO_setOutput(ePortE, 31, eHigh);
            /* Turn Red LED on */
            GPIO_setOutput(ePortE, 29, eLow);

            /* Disable unused pins */
            GPIO_deInitPin(ePortD, 5); /* Blue LED */
            GPIO_deInitPin(ePortE, 31); /* Green LED */
            GPIO_deInitPin(ePortA, 5);
            /* Increment in System Register file address time that MCU has entered
             * to VLLS mode */
            (*systemRegPtr)++;
            /* Specify that VLLS is selected */
            SMC_setVLLSxMode();
        }
    }
    /* Never leave main */
    return 0;
}
```