NXP Semiconductors
Application Note

Document Number: AN5402
Rev. 0, 01/2017

How to Measure the Power Consumption of a Peripheral

Contents

1. Introduction

This Application Note outlines the steps to measure the power consumption of a peripheral and figure out some key points in the measurement. The document takes low-power timer (LPTMR) and LPUART as examples to introduce the method to measure the power consumption of peripheral. The test code is developed in IAR and FRDM-KE15Z board.

The Power Estimation Tool for Kinetis® MCUs lets you estimate and optimize your system's power consumption quickly with a simple graphical interface. The tool measures data using the method described in the Application Note.

2. Measurement method

2.1 State definitions and calculation methods

The basic idea of estimating the power consumption of a peripheral is to measure the current differences between the peripheral's state of un-init and state of working. The current difference is called peripheral adder. However, some factors affect the accuracy of the measurement. Such as the dynamic power consumption of the CPU core, the power mode setting, the accuracy and range of Ammeter, and so on. Therefore, it is essential to find out some ways to fix those problems and improve the measurement accuracy.

To avoid adding the CPU dynamic power consumption into the peripheral adder, one need stop the CPU core when measuring the current. Thus, the power modes of Wait/Stop, under which CPU is stopped is used to measure current. To be more specific, which Power Mode (Wait/Stop) is used depends on the current consumption of peripheral. If current consumption value of the peripheral is around nA grade and the peripheral can work under Stop Mode and is recommended. When measuring communication peripherals LPUART, LPSPI, LPI2C, and so on. To transfer data, use DMA instead of CPU. This means the data transfer between memory and peripheral is handled by DMA without CPU's interaction.

Per above considerations, define the two power states for getting the peripheral adder and they are called BASELINE and RUN.

- BASELINE: Set the clock source, the function clock, the power mode and then initialize the DMA or other peripherals, if needed. Put the chip entering the Wait/Stop mode during the measurement. In the baseline condition, there are no peripheral clock gates enabled. In the baseline only the average current of the power mode and clock configuration are measured.
- RUN: Enable the clock gate, initialize the peripheral, and initiate it. For example, transmitting data, performing a/d conversion and so on, depending on the peripheral. Next, put the chip entering the Wait/Stop Power mode.

When the $IDD_{BASELINE}$ measurement have completed, an interrupt signal is required to wake up the chip under Wait/Stop and change the State from BEASLINE to RUN. The on-board key SW2 is used to trigger an interrupt. After measuring the IDD_{RUN}, you can estimate the power consumption of the peripheral by using the following formula:

$$Adder = IDD_{RUN} - IDD_{BASELINE}$$

For the ammeter, you need to select a suitable range. If the state is changed from BASELINE to RUN and the power mode is Stop mode, you may need to change the range of the ammeter. Otherwise, when the SW2 is pressed, the current value may exceed the ammeter range. This may cause something unexpected, for instance: Reset. Figure 1 shows the whole process of the measurement.

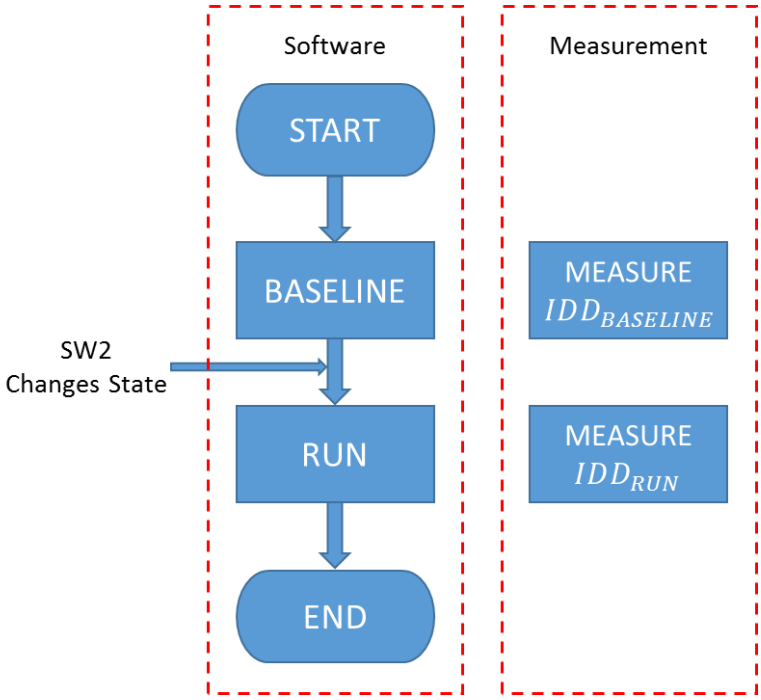

Figure 1. Measurement Flow Chart

2.2 Hardware settings

FRDM-KE15Z board setting:

- Power Supply: The Power Source is 5V DC through PC USB port to SDA USB. On-board LDO can supply 3.3V DC. The examples of this Application Note 5V are selected as power supply, so the 1-2 of J15 needs to be connected.
- Current Measure: J14 with ammeter can be used for measure IDD (VDD+VDDA) of each state.
- State Change: The on-board key SW2 is used for changing State BASELINE -> RUN.

NOTE

The Jumper Caps of J7 and J8 are removed before the program runs because Open-SDA circuit causes more power consumption.

FRDM-KE15Z Board needs to be re-powered after download if the Stop Mode is selected.

For the communication peripheral, the onboard trace length have a big impact on $Adder$ when the work mode is Transmit.

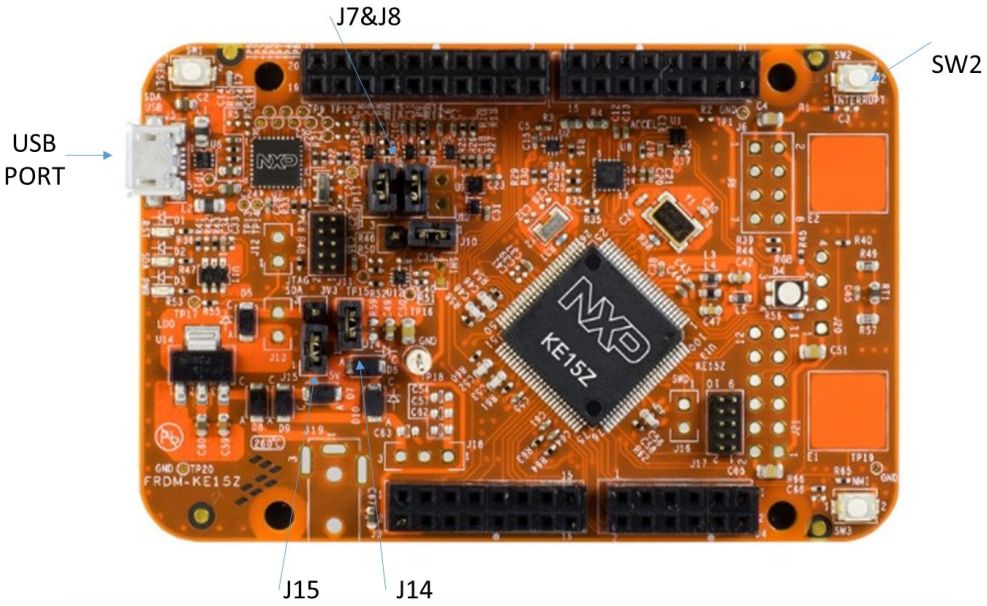

Figure 2. FRDM-KE15Z Board

3 Examples: How to measure the power consumption of LPTMR

You can configure the LPTMR to operate as a time counter with optional prescaler, or as a pulse counter with optional glitch filter, across all the power modes. In this example, the LPTMR is configured as a time counter and lists steps to measure the power consumption of the LPTMR.

The steps are as following:

1. The clock source and the frequency of the function clock is determined at this step. Different configurations make different power consumption. You can configure it according to your requirements. In this example, FIRC 48MHz is selected as Clock Source and the frequency of Function Clock is 48 MHz. The On-board key SW2 needs to configure as GPIO pull-up and interrupt on falling edge. The Stop Mode is selected because the current consumption value of the peripheral is around nA grade.

```
BASELINE()
{
	/********************** BASELINE State *********************/
	lptmr_config_t lptmrConfig;
	port_pin_config_t pinConfig = {0};
	BOARD_BootClockRUN();

	/* Initialize the on-board key SW2 */
	CLOCK_EnableClock(kCLOCK_PortB);
	pinConfig.pullSelect = kPORT_PullUp;
	pinConfig.mux = kPORT_MuxAsGpio;
	PORT_SetPinConfig(BOARD_SW2_PORT, BOARD_SW2_GPIO_PIN, &pinConfig);
	PORT_SetPinInterruptConfig(BOARD_SW2_PORT, \
	BOARD_SW2_GPIO_PIN, kPORT_InterruptFallingEdge);
	EnableIRQ(BOARD_SW2_IRQ);

	/* Configure FIRC as Clock Source and Function Clock is 48 MHz */
	clocks_run_48M_firc(1);
	SCG->FIRCCSR = 0x03;	//Enable FIRC and Enable Stop Mode

	/* BASELINE: Enter Stop Power Mode and Measure IDD_BASELINE here */
	SMC_SetPowerModeStop(SMC,kSMC_PartialStop);
}
```

2. Measure the current consumption $IDD_{BASELINE}$ with Ammeter. Then press SW2 change the State from BASELINE to RUN. And the range of Ammeter may need to change before press the SW2.
3. Select the Timer mode, configure the Timer period and enable Timer. LPTMR's interrupt should not enable because the interrupt will cause the Power mode from Stop to Run. Put the chip entering the Stop mode again.

```
RUN()
{
/********************** RUN State *********************/
         /* Initialize LPTMR */
    CLOCK_SetIpSrc(kCLOCK_Lptmr0, kCLOCK_IpSrcFircAsync);
    LPTMR_GetDefaultConfig(&lptmrConfig);
    lptmrConfig.timerMode = kLPTMR_TimerModeTimeCounter;
    lptmrConfig.prescalerClockSource = kLPTMR_PrescalerClock_0;
    lptmrConfig.enableFreeRunning = false;
    lptmrConfig.bypassPrescaler = false;
    lptmrConfig.value = kLPTMR_Prescale_Glitch_11;
    LPTMR_Init(LPTMR0, &lptmrConfig);

    /* Set timer period, as LPTMR counter is 16-bit only, the clock source must be prescaled
4096 */
         LPTMR_SetTimerPeriod(LPTMR0, USEC_TO_COUNT(LPTMR_USEC_COUNT, \
        (CLOCK_GetFreq(kCLOCK_ScgFircAsyncDiv2Clk)/4096)));

    /* Start counting */
    LPTMR_StartTimer(LPTMR0);

    /* RUN: Enter Stop Power Mode and Measure IDD_RUN here */
    SMC_SetPowerModeStop(SMC,kSMC_PartialStop);
}
```

4. Measure the current consumption IDD_{RUN} with ammeter. Now, the Adder of LPTMR can be estimated.

NOTE

FRDM-KE15Z Board needs to be re-powered after download if the Stop mode is selected.

4 Examples: How to measure the power consumption of LPUART

Low-power universal asynchronous receiver/transmitter (LPUART) is a widely used communication peripheral. For the communication peripheral, the on-board trace length have a big impact on power consumption. This example lists the steps to measure the power consumption of LPUART. Usually the communication peripheral need two boards for measurement. One is transmitter, the other is receiver.

1. The clock source and the frequency of the function clock is determined at this step. Different configurations make different power consumption. You can configure it according to your requirements. In this example, FIRC is selected as Clock Source and the frequency of Function Clock is 48 MHz. The On-board key SW2 need to configure as GPIO pull-up and interrupt on falling edge. The data transfer between memory and peripheral is handled by DMA without CPU's interaction. The Wait Mode is selected because the current consumption value of the peripheral is around uA grade.

```
BASELINE()
{
       /********************* BASELINE State *********************/
    port_pin_config_t pinConfig = {0};
    edma_config_t edmaConfig;
    edma_transfer_config_t transferConfig;
    lpuart_config_t lpuartConfig;
    clocks_run_48M_firc(2);

    /* Initialize the on-board key SW2 */
    CLOCK_EnableClock(BOARD_SW2_CLOCK);
    pinConfig.pullSelect = kPORT_PullUp;
    pinConfig.mux = kPORT_MuxAsGpio;
    PORT_SetPinConfig(BOARD_SW2_PORT, BOARD_SW2_GPIO_PIN, &pinConfig);
PORT_SetPinInterruptConfig(BOARD_SW2_PORT, BOARD_SW2_GPIO_PIN,  \
kPORT_InterruptFallingEdge);
    EnableIRQ(BOARD_SW2_IRQ);

       /* Configure FIRC as Clock Source and Function Clock is 48 MHz */
    clocks_run_48M_firc(1);

    /**
     * DMA/DMAMUX configure
     * DMAMUX configure channel to transmit or receive with transmit/receive module trigger
source.
     * DMA configure source, destination address and continuous working
     */
    DMAMUX_Init(DMAMUX);
#ifdef LPUART_ADDER_RECEIVER
    DMAMUX_SetSource(DMAMUX, 0, 2); // LPUART0 RX
    DMAMUX_EnableChannel(DMAMUX, 0);
#else
    DMAMUX_SetSource(DMAMUX, 0, 3); // LPUART0 TX
    DMAMUX_EnableChannel(DMAMUX, 0);
#endif

    // Init EDMA and Channel0
    EDMA_GetDefaultConfig(&edmaConfig);
    edmaConfig.enableHaltOnError = false;
    EDMA_Init(DMA0, &edmaConfig);
```

```
    EDMA_ResetChannel(DMA0, 0);

    // Disable auto stop request and dma aysnc request
    EDMA_EnableAsyncRequest(DMA0, 0, true);
    EDMA_EnableAutoStopRequest(DMA0, 0, false);

#ifdef LPUART_ADDER_RECEIVER
    transferConfig.srcAddr = LPUART_GetDataRegisterAddress(LPUART0);
    transferConfig.destAddr = (uint32_t)recvBuf;
#else
    transferConfig.srcAddr = (uint32_t)&testData;
    transferConfig.destAddr = LPUART_GetDataRegisterAddress(LPUART0);
#endif
    // Offset applied to current address to form next transfer address
    transferConfig.srcTransferSize = kEDMA_TransferSize1Bytes;
    transferConfig.destTransferSize = kEDMA_TransferSize1Bytes;
    transferConfig.srcOffset = 0;
#ifdef LPUART_ADDER_RECEIVER
    transferConfig.destOffset = 1;
#else
    transferConfig.destOffset = 0;
#endif
    // 1bytes per DMA request
    transferConfig.minorLoopBytes = 1;
    transferConfig.majorLoopCounts = 100;

    EDMA_SetTransferConfig(DMA0, 0, &transferConfig, NULL);
#ifdef LPUART_ADDER_RECEIVER
    DMA0->TCD[0].DLAST_SGA = -sizeof(recvBuf);
#endif

    // Start dma channel0 transfer
    EDMA_EnableChannelRequest(DMA0, 0);

      /* BASELINE: Enter Wait Power Mode and Measure IDD_BASELINE here */
    SMC_SetPowerModeWait(SMC);
}
```

2. Measure the current consumption $IDD_{BASELINE}$ with Ammeter. Then press SW2 change the State from BASELINE to RUN.
3. Configure the LPUART in your required baud rate and mode, and then put the chip entering

WAIT mode again.

```
RUN()
{
       /********************** RUN State **********************/
       /* Initialize LPUART */
    CLOCK_EnableClock(kCLOCK_PortA);
    PORT_SetPinMux(PORTA, 10U, kPORT_MuxAlt3);
    PORT_SetPinMux(PORTA, 11U, kPORT_MuxAlt3);
    CLOCK_SetIpSrc(kCLOCK_Lpuart0, kCLOCK_IpSrcFircAsync);
    LPUART_GetDefaultConfig(&lpuartConfig);
    lpuartConfig.baudRate_Bps = 460800;
    LPUART_Init(LPUART0, &lpuartConfig, CLOCK_GetFreq(kCLOCK_ScgFircAsyncDiv2Clk));

#ifdef LPUART_ADDER_RECEIVER
    LPUART_EnableRxDMA(LPUART0, true);
    LPUART_EnableRx(LPUART0, true);
#else
    LPUART_EnableTxDMA(LPUART0, true);
    LPUART_EnableTx(LPUART0, true);
#endif

    /* RUN: Enter Wait Power Mode and Measure IDD_RUN here */
    SMC_SetPowerModeWait(SMC);
}
```

4. Measure the current consumption IDD_{RUN} with Ammeter. Now, the Adder of LPUART can be estimated.

NOTE

For the communication peripheral, the onboard trace length impacts the $Adder$ when the work mode is Transmit.

5 Conclusion

This Application Note outlines the steps to measure the power consumption of a peripheral and shows the measurement of two modules power consumption. Other modules can also be measured in a similar way. A suitable power mode helps to improve the measurement accuracy.

6 Revision history

Table 1. Revision history

Revision number	Date	Substantive changes
0	01/2017	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN5402
Rev. 0
01/2017