AN13056

Low-Power Modes and Wake-Up Time

Rev. 1 — 08 April 2022
Application Note

Contents

1 RT600 Introduction

The RT6xx is a family of dual-core microcontrollers featuring an Arm Cortex-M33 CPU combined with a Cadence Xtensa HiFi 4 advanced audio Digital Signal Processor (DSP). The family offers a rich set of peripherals with a feature of very low-power consumption.

This application note introduces the various low-power modes of the RT600 series, the software APIs details to enter in low-power mode and wake-up source used for each low-power mode. This document also describes hardware and software environment as well as procedure to measure supply current and wake-up time for each low-power mode.

The application note covers the following topics:

1. All low-power modes in RT600.
2. Overview on entry and wake-up Implementations for low-power modes.
3. Demonstrate how to measure current and wake-up time for each low-power mode.

2 RT600 low-power modes introduction

On the RT600, there are four reduced power modes:

- Sleep mode
- Deep-sleep mode
- Deep power-down mode
- Full deep power-down mode

There are three special modes of the processor for power reduction like; sleep mode, deep-sleep mode, and deep power-down mode. These modes can be activated by power modes library API from the SDK software package.

Power usage is controlled by settings in the register within the SYSCON block, regulator settings controlled via a Power API, and the operating mode of a CPU. The following modes are supported in order from maximum to minimum power consumption.

2.1 Active mode

In active mode, clocks are enabled to the CPU and memories & peripherals are enabled.

The chip is in active mode after reset and the default power configuration is determined by the boot values of the PDRUNCFG and PSCCTL registers. Power and clocks to selected peripherals can be optimized for power consumption during runtime. The active mode consumes the highest power among all power modes. All low-power modes can be invoked from this power mode.

2.2 Sleep mode

In sleep mode, the clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

The selected peripherals can be clocked to continue the operation during sleep mode and they may generate the interrupts or can be configured as a wake-up source to resume the execution of the processor. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The SRAM contents can be retained based on software configuration.

2.3 Deep-sleep mode

In the deep-sleep mode, the system clock to the processor is disabled like the sleep mode. The main clock, all peripheral clocks, and primary clock source are also disabled.

All the analog blocks are powered down by default but these blocks are configurable to keep running during this mode and can be used as a wake-up source. Deep-sleep mode reduces the overall power consumption by eliminating the power used by analog peripherals, dynamic power used by the processor, memory systems and related controllers, and internal buses. The SRAM contents can be retained based on software configuration.

2.4 Deep power-down mode

In the deep power-down mode, all clocks, the core, and all peripherals are powered down except RTC.

The device can wake-up from deep power-down mode via RESET pin, PMIC_IRQ_N pin, and RTC alarm. The device can enter into deep power-down mode only by CM33. External power supply should be left on deep power-down mode. The contents of SRAM and registers are not retained.

2.5 Full deep power-down mode

The full deep power-down and deep power-down mode are quite similar. In the full deep power-down mode, the whole system shuts down making all power pins externally powered off (except VDD_AO18).

The device can be wake-up by external interrupts such as RESET pin, PMIC_IRQ_N pin, and RTC alarm. Apart from the normal operations like full deep power-down mode which allows the device power pins to be externally powered off. The contents of SRAM and registers are not retained.

2.6 Power down the Interfaces

The RT600 provides a PDRUNCFG registers to power down the desired interface. The power to various analog blocks (RAMs, PLL, oscillators, and many others) can be controlled individually through the PDRUNCFG registers.

SYSCTLx_PDRUNCFGx register controls the power to various blocks during normal operation. Configuring PDRUNCFG is typically accomplished using a Power APIs that handles all the details of altering PDRUNCFG bits. In this application, the peripheral list shown in Table 1, Table 2, Table 3, and Table 4 are power-down by setting the bit in SYSCTLx_PDRUNCFGx register.

Table 1. Run configuration register 1 (SYSCTL0_PDRUNCFG0)

Bit	Symbol	Description
14	LPOSC_PD	1 MHz low power oscillator 0 – Function is Enable 1 – Function is Powered Down
21	ADC_PD	ADC Analog Function 0 – Function is Enable 1 – Function is Powered Down

Table continues on the next page...

Table 1. Run configuration register 1 (SYSCTL0_PDRUNCFG0) (continued)

Bit	Symbol	Description
22	ADC_LP	ADC Low Power Mode 0 – Function is Enable 1 – Function is Powered Down
23	ADCTEMPSNS_PD	ADC Temperature Sensor 0 – Function is Enable 1 – Function is Powered Down
25	ACMP_PD	Analog Comparator 0 – Function is Enable 1 – Function is Powered Down
26	HSPAD0_VDET_LP	FlexSPI high speed pad voltage detects sleep mode 0 – High Speed pad VDET in normal mode 1 - High Speed pad VDET in sleep mode
27	HSPAD0_REF_PD	FlexSPI high speed pad sleep mode 0 – High Speed pad VREF Enabled 1 – High speed pad VREF in power down
28	HSPAD2_VDET_LP	SDIO0 high speed pad voltage detects sleep mode 0 – High Speed pad VDET in normal mode 1 - High Speed pad VDET in sleep mode
29	HSPAD2_REF_PD	SDIO0 high speed pad sleep mode 0 – High Speed pad VDET in normal mode 1 - High Speed pad VDET in sleep mode

Table 2. Run configuration register 2 (SYSCTL0_PDRUNCFG1)

Bit	Symbol	Description
0	PQ SRAM APD	Array Power Down for Power Quad SRAM 0 – Function is Enable 1 – Function is Powered Down
1	PQ SRAM PPD	Periphery Power Down for Power Quad SRAM 0 – Enable 1 – Disable
4	USBHS_SRAM_APD	Array Power Down for USB SRAM

Table continues on the next page...

Table 2. Run configuration register 2 (SYSCTL0_PDRUNCFG1) (continued)

Bit	Symbol	Description
		0 – Function is Enable 1 – Function is Powered Down
5	USBHS_SRAM_PPD	Periphery Power Down for USB SRAM 0 – Function is Enable 1 – Function is Powered Down
6	USDHC0_SRAM_APD	Array Power Down for uSDHC0 SRAM 0 – Function is Enable 1 – Function is Powered Down
7	USDHC0_SRAM_PPD	Periphery Power Down for uSDHC0 SRAM 0 – Function is Enable 1 – Function is Powered Down
8	USDHC1_SRAM_APD	Array Power Down for uSDHC1 SRAM 0 – Function is Enable 1 – Function is Powered Down
9	USDHC1_SRAM_PPD	Periphery Power Down for uSDHC1 SRAM 0 – Function is Enable 1 – Function is Powered Down
10	CASPER_SRAM_APD	Array Power Down for Casper SRAM 0 – Function is Enable 1 – Function is Powered Down
11	CASPER_SRAM_PPD	Periphery Power Down for Casper SRAM 0 – Function is Enable 1 – Function is Powered Down

Table 3. Run configuration register 3 (SYSCTL0_PDRUNCFG2)

Bit	Symbol	Description
0	SRAM_IF0_APD	Array Power Down for SRAM Interface 0 0 – Function is Enable 1 – Function is Powered Down
1	SRAM_IF1_APD	Array Power Down for SRAM Interface 1 0 – Function is Enable

Table continues on the next page...

Table 3. Run configuration register 3 (SYSCTL0_PDRUNCFG2) (continued)

Bit	Symbol	Description
		1 – Function is Powered Down
2	SRAM_IF2_APD	Array Power Down for SRAM Interface 2 0 – Function is Enable 1 – Function is Powered Down
29:3	SRAM_IF[Bit]_APD	Array Power Down for SRAM Interface [Bit] 0 – Function is Enable 1 – Function is Powered Down

Table 4. Run configuration register 4 (SYSCTL0_PDRUNCFG3)

Bit	Symbol	Description
0	SRAM_IF0_PPD	Periphery Power Down for SRAM Interface 0 0 – Function is Enable 1 – Function is Powered Down
1	SRAM_IF1_PPD	Periphery Power Down for SRAM Interface 1 0 – Function is Enable 1 – Function is Powered Down
2	SRAM_IF2_PPD	Periphery Power Down for SRAM Interface 2 0 – Function is Enable 1 – Function is Powered Down
29:3	SRAM_IF[Bit]_PPD	Periphery Power Down for SRAM Interface [Bit] 0 – Function is Enable 1 – Function is Powered Down

2.7 Low-power mode summary

Table 5 describes the peripherals that can be configured during power saving modes.

Table 5. Peripheral configuration in reduced power modes

Peripheral/clock	Reduced power mode		
	Sleep	Deep-sleep	Deep power-down (applies to both deep power-down and full deep power-down modes)
1m_lposc	Software configured	Software configured	Off
16m_irc	Software configured	Software configured	Off
48/60m_irc	Software configured	Software configured	Off
Crystal Oscillator	Software configured	Software configured	Off
RTC and RTC Oscillator	Software configured	Software configured	Off
System PLL	Software configured	Software configured	Software configured
Audio PLL	Software configured	Software configured	Off
SRAM Memory Arrays	Software configured	Software configured	Off
SRAM Periphery	Software configured	Software configured	Off
Boot ROM	On	Off	Off
Other Digital peripherals	Software configured	Software configured	Off
A to D Convertor	Software configured	Software configured	Off
Analog comparator	Software configured	Software configured (the comparator may be on in deep-sleep mode, but cannot generate a wake-up interrupt)	Off

3 Entering low-power modes and waking up

Power to the RT600 is supplied via two power domains. The “main power domain” has number of pins and options, and supplies to the core, peripheral, memories, inputs, and outputs.

There is a secondary *always-on power domain* powered by VDD_AO1V8, it includes the RTC and wake-up timer. This domain always has power as long as sufficient voltage is supplied to VDD_AO1V8.

Power usage is controlled by settings in register within the SYSCON block, regulator settings controlled by Power APIs, and the operating mode of a CPU. This application note describes how to enter and wake-up from the various low-power modes.

3.1 Power control API

The power control APIs provide the functions to configure the system for expected performance requirements. Table 6 shows the list of power APIs used in the application.

Table 6. Power API ROM Calls

Function prototype	API description
void BOARD_SetPmicVoltageForFreq(uint32_t main_clk_freq, uint32_t dsp_main_clk_freq);	PMIC based on input frequency to provide different voltages.
void POWER_EnterSleep(void);	Configures and enters in SLEEP low-power mode.
Void POWER_EnterDeepSleep(const uint32_t exclude_from_pd[4]);	PMC Deep Sleep function call. This function call configures the board to enter in deep sleep mode.
void POWER_EnterDeepPowerDown(const uint32_t exclude_from_pd[4]);	PMC Deep Power Down function call. This function configures the board to enter in deep power down mode.
void POWER_EnterFullDeepPowerDown(const uint32_t exclude_from_pd[4]);	PMC Full Deep Power Down function call. This function configures the board to enter in full deep power down mode.

3.1.1 BOARD_SetPmicVoltageForFreq

The BOARD_SetPmicVoltageForFreq API is used to set different voltages based on input frequency provided.

Table 7. BOARD_SetPmicVoltageForFreq

Routine	API description
Function Prototype	BOARD_SetPmicVoltageForFreq(uint32_t main_clk_freq, uint32_t dsp_main_clk_freq);
Input Parameter	Main clock frequency and DSP main clock frequency
Result	None
Description	PMIC based on input frequency to provide different voltages

3.1.2 POWER_EnterSleep

The POWER_EnterSleep API is used to configure and enter into sleep mode. It stops the system clock to the CPU and suspends the execution of instruction until a reset or interrupt occurs. Selected peripheral function can continue operation during sleep mode and may cause wake-up interrupt.

Table 8. POWER_EnterSleep

Routine	API description
Function Prototype	void POWER_EnterSleep(void);
Input Parameter	None
Result	None
Description	Configures and enters in sleep low-power mode.

3.1.3 POWER_EnterDeepSleep

The POWER_EnterDeepSleep API is used to configure and enter into deep sleep mode. Selected analog blocks can be kept running by using this API in the deep sleep mode and it can be used as wake-up source.

Table 9. POWER_EnterDeepSleep

Routine	API description
Function Prototype	void POWER_EnterDeepSleep(const uint32_t exclude_from_pd[4]);
Input Parameter	Bit mask of the PDRUNCFG0 to PDRUNCFG3 registers to identify peripherals to be powered on during deep sleep mode.
Result	None
Description	Configures and enters into deep sleep low-power mode. Based on the bit mask of the PDRUNCFG0-3 registers, it excludes those peripherals to be powered down in deep sleep mode. These peripherals can be configured as a wake-up source.

3.1.4 POWER_EnterDeepPowerDown

The POWER_EnterDeepPowerDown API is used to configure and enter into the deep power-down mode. In this mode, the contents of the SRAM and registers are not retained. All functional pins are tri-stated in deep power-down mode.

Table 10. POWER_EnterDeepPowerDown

Routine	API description
Function Prototype	void POWER_EnterDeepPowerDown(const uint32_t exclude_from_pd[4]);
Input Parameter	Bit mask of the PDRUNCFG0 to PDRUNCFG3 registers to identify RTC and RTC oscillator to be powered on during deep power-down mode.
Result	None
Description	Configures and enters into deep power-down mode. Based on the bit mask of the PDRUNCFG0-3 registers, it excludes RTC and RTC oscillator to be powered down in deep power-down mode.

3.1.5 POWER_EnterFullDeepPowerDown

The POWER_EnterFullDeepPowerDown API is used to configure and enter into full deep power-down mode. In full deep power-down mode, all rails can be powered off and VDD_AO18 supply can remain powered.

Table 11. POWER_EnterFullDeepPowerDown

Routine	API description
Function Prototype	void POWER_EnterFullDeepPowerDown (const uint32_t exclude_from_pd[4]);
Input Parameter	Bit mask of the PDRUNCFG0 to PDRUNCFG3 registers to identify RTC and RTC oscillator to be powered on during full deep power-down mode.
Result	None
Description	Configures and enters into full deep power-down mode. Based on the bit mask of the PDRUNCFG0-3 registers, it excludes RTC and RTC oscillator to be powered down in full deep power-down mode.

3.1.6 Input parameter

Param0: exclude_from_pd[4]

Bit mask of the PDRUNCFG0 through PDRUNCFG3 registers to identify functions to be powered on during deep sleep, deep power-down, and full deep power-down mode. This parameter is not used for sleep mode.

The exclude_from_pd parameter is a 32-bit value that corresponds to the PDRUNCFG<x> registers. This parameter defines the peripherals that becomes active in the respective low-power mode.

This application uses below macros to allow easy selection of analog peripherals to be powered on in deep-sleep mode.

```
/*Main crystal oscillator*/
#define SYSCTL0_PDRUNCFG0_SYSXTAL_PD_MASK (0x2000U)

/*Main PLL internal regulator and analog functions*/
#define SYSCTL0_PDRUNCFG0_SYSPLLLDO_PD_MASK (0x2000U)
#define SYSCTL0_PDRUNCFG0_SYSPLLANA_PD_MASK (0x4000U)

/*Audio PLL internal regulator and analog functions*/
#define SYSCTL0_PDRUNCFG0_AUDPLLLDO_PD_MASK (0x8000U)
#define SYSCTL0_PDRUNCFG0_AUDPLLANA_PD_MASK (0x10000U)
```

3.2 Switching executional parameters

In this application, there are various executional variations that are considered while performing the measurements in low-power mode. These executional variations are macro defined and directly inflict the power consumption of the RT600.

3.2.1 Switch to FRO 12 MHz clocking before entering sleep mode

By default, RT600 runs at 250 MHz. In this application, SLEEP_AT_48M_IRC_DIV4 macro used to change the clock frequency by 12 MHz FRO.

Before entering into the sleep mode, if the application switch to 12 MHz FRO then after wake-up the application should recover the intended clock. This application code can be enabled by changing #define SLEEP_AT_48M_IRC_DIV4 from 0 to 1, sample implementation is shown below.

```
#if SLEEP_AT_48M_IRC_DIV4
    mainclk_sel[0] = CLKCTL0->MAINCLKSELA;
    mainclk_sel[1] = CLKCTL0->MAINCLKSELB;
    dspclk_sel[0]  = CLKCTL1->DSPCPUCLKSELA;
    dspclk_sel[1]  = CLKCTL1->DSPCPUCLKSELB;
    cpu_div        = CLKCTL0->SYSCPUAHBCLKDIV;
    CLKCTL0->MAINCLKSELA= CLKCTL0_MAINCLKSELA_SEL(0);

    CLKCTL0->MAINCLKSELB= CLKCTL0_MAINCLKSELB_SEL (0);
    CLKCTL1->DSPCPUCLKSELA= CLKCTL1_DSPCPUCLKSELA_SEL (0);
    CLKCTL1->DSPCPUCLKSELB= CLKCTL1_DSPCPUCLKSELB_SEL (0);
    CLKCTL0->SYSCPUAHBCLKDIV = 0;
    while (CLKCTL0->SYSCPUAHBCLKDIV & CLKCTL0_SYSCPUAHBCLKDIV_REQFLAG_MASK);
#endif
    POWER_Entersleep();
#if SLEEP_AT_48M_IRC_DIV4

    CLKCTL0->SYSCPUAHBCLKDIV = cpu_div;
    while (CLKCTL0->SYSCPUAHBCLKDIV & CLKCTL0_SYSCPUAHBCLKDIV_REQFLAG_MASK);
    CLKCTL0->MAINCLKSELA=mainclk_sel[0] & CLKCTLO_MAINCLKSELA_SEL_MASK;
    CLKCTL0->MAINCLKSELB  = mainclk_sel[1] & CLKCTLO_MAINCLKSELB_SEL_MASK;
    CLKCTL1->DSPCPUCLKSELA = dspclk_sel[0] & CLKCTL1_DSPCPUCLKSELA_SEL_MASK;
    CLKCTL1->DSPCPUCLKSELB = dspclk_sel[1] & CLKCTL1_DSPCPUCLKSELB_SEL_MASK;
#endif
```

3.2.2 Exclude peripheral during deep sleep mode

By default, the main crystal oscillator, the main PLL, and the audio PLL are powered down in deep-sleep mode. In this application, #define APP_DEEPSLEEP_RUNCFG0 macro is an argument to the POWER_EnterDeepSleep() API and is used to keep certain peripherals active in deep-sleep mode.

To turn on the main crystal oscillator during deep sleep mode, change the value for the #define APP_DEEPSLEEP_RUNCFG0 macro before passing it to the power API as shown below.

```
#define APP_DEEPSLEEP_RUNCFG0
 (SYSCTL0_POSLEEPCFG0_RBB_PD_MASK|SYSCTL0_PDRUNCFG0_SYSXTAL_PD_MASK)

#define APP_EXCLUDE_FROM_DEEPSLEEP
    (((const uint32 t[]) {APP_DEEPSLEEP_RUNCFG0,
    (SYSCTL0_PDSLEEPCFG1_FLEXSPI_SRAM_APD_MASK),
    APP_DEEPSLEEP_RAM_APD, APP_DEEPSLEEP_RAMPPD}))
POWER_EnterDeepSleep (APP_EXCLUDE_FROM_DEEPSLEEP);
```

3.2.3 Wake-up source as PMIC_IRQ_N in deep power-down and full deep power-down modes

To use PMIC_IRQ_N pin as wake-up source from deep power-down and full deep power-down modes, change WAKE_UP_WITH_PMIC_IRQ_N macro value from 0 to 1.

3.3 Sleep mode

3.3.1 Entering sleep mode

Sleep mode stops CPU execution to save power. The clock to the core is shut off. Peripherals and memories are operational. By using POWER_EnterSleep() API, the RT600 can enter into the sleep mode.

3.3.2 Waking up from sleep mode

RT600 can wake-up from sleep mode by interrupt or when RESET occurs. In this application, pin interrupt generated by user SW1 is used to wake-up the device from sleep mode. After waking up by an interrupt, RT600 is having original power configuration defined by PDRUNCFG and PSCCTL registers.

3.4 Deep sleep mode

3.4.1 Entering deep sleep mode

POWER_EnterDeepSleep() API can be used to enter into the deep sleep mode. In this application note, some peripherals can be kept operational in the deep sleep mode by configuring it using mentioned API.

3.4.2 Waking up from deep sleep mode

GPIO pin interrupts, and selected peripherals such as USB, DMIC, SPI, I2C, USART, WWDT, RTC, and Micro-tick timer can be used to as wake-up source for deep-sleep mode.

In this application, pin interrupt generated by user SW1 is used to wake-up the device from deep-sleep mode. Pin 1 from the Port 1 (P1_1) is used as the source of a pin interrupt and this pin is being pulled low by pressing user SW1. The same interrupt is enabled in the NVIC using EnableDeepSleepIRQ() API.

3.5 Deep power-down mode

3.5.1 Entering deep power-down mode

POWER_EnterDeepPowerDown() API can be used to enter into the deep power-down mode.

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the VDD_AO1V8 pin.

3.5.2 Waking up from deep power-down mode

Waking up from the deep power-down mode can be accomplished by pressing and releasing the SW3 RESET button or by providing low pulse to PMIC_IRQ_N (R85).

3.6 Full deep power-down mode

3.6.1 Entering full deep power-down mode

POWER_EnterFullDeepPowerDown() API can be used to enter into the full deep power-down mode.

Full deep power-down mode allows the power pins (with an exception of VDD_A018) to be externally powered off to save additional power.

3.6.2 Waking up from full deep power-down mode

Waking up from the full deep power-down mode can be accomplished by pressing and releasing the SW3 RESET button or by providing low pulse to PMIC_IRQ_N (R85).

3.7 Other power consumption reduction considerations

3.7.1 Turning off unused analog peripherals

Analog peripherals like ADC, temperature sensor, and analog comparator can be powered down for additional power savings in deep sleep, deep power-down and full deep power-down modes.

3.7.2 Clock gating the digital peripherals

For example, when IO configuration is done, the user can shut down the clock for IOCON to save additional power in low-power modes.

3.8 Debug session consideration with low-power modes

The debug session stays active in sleep mode but not in deep sleep, deep power-down and full deep power-down modes.

3.8.1 No reconnection scheme provided in user’s application

In this application, when the device is switched to deep power-down mode or full deep power-down mode the debug session is terminated. Debug session can be reinitialized after waking up from deep sleep mode.

4 Low-power mode demo

This section provides details on how RT600 MCU can be put into various low-power mode and can wake-up from the same. For sleep and deep sleep modes GPIO pin (P1_1) is configured as wake-up source and for deep power-down and full deep power-down mode RESET pin is configured as wake-up source.

The example is based on RT600 open source SDK platform of NXP and it is developed on SDK_2.8.2_EVK-MIMXRT685.

Keil, IAR and MCUXpresso toolchains are supported in this application.

4.1 Software setup

These IDEs were used to verify the application.

- Keil IDE version 5.31
- IAR IDE 8.50.5 with applied patch for RT600
- MCUXpresso IDE 11.2.0

This low-power mode application is a modified version of sdk example power_manger (can be found inside <SDK >\boards\evkmimxrt685\demo_apps\power_manager\>). The SW package is available with this application note for all three IDEs.

Please download the software package provided with this AN. For IAR and Keil, unzip the package and copy the content above RT600 SDK(SDK_2.8.2_EVK-MIMXRT685). The zipped package contains changes in power_manager.c, fsl_pint.c , startup_MIMXRT685S_cm33.S for arm and startup_MIMXRT685S_cm33.s for iar. For MCUXpresso, use import project(s) from file system to import low-power mode application. For details on building and downloading the application, refer the *Getting Started with MCUXpresso SDK for EVK-MIMXRT685.pdf* document located inside SDK documentation at <SDK >\docs\> location.

4.2 Hardware setup

- RT600 development board.
- Digital multimeter for current measurement.
- Oscilloscope for wake-up time measurement.

For the RT600 low-power consumption measurement, install connector at JP29. Install jumper top to JP22 position 2-3 to pull low LDO_ENABLE and use an off-chip PMIC PCA9420 to supply power to core logic.

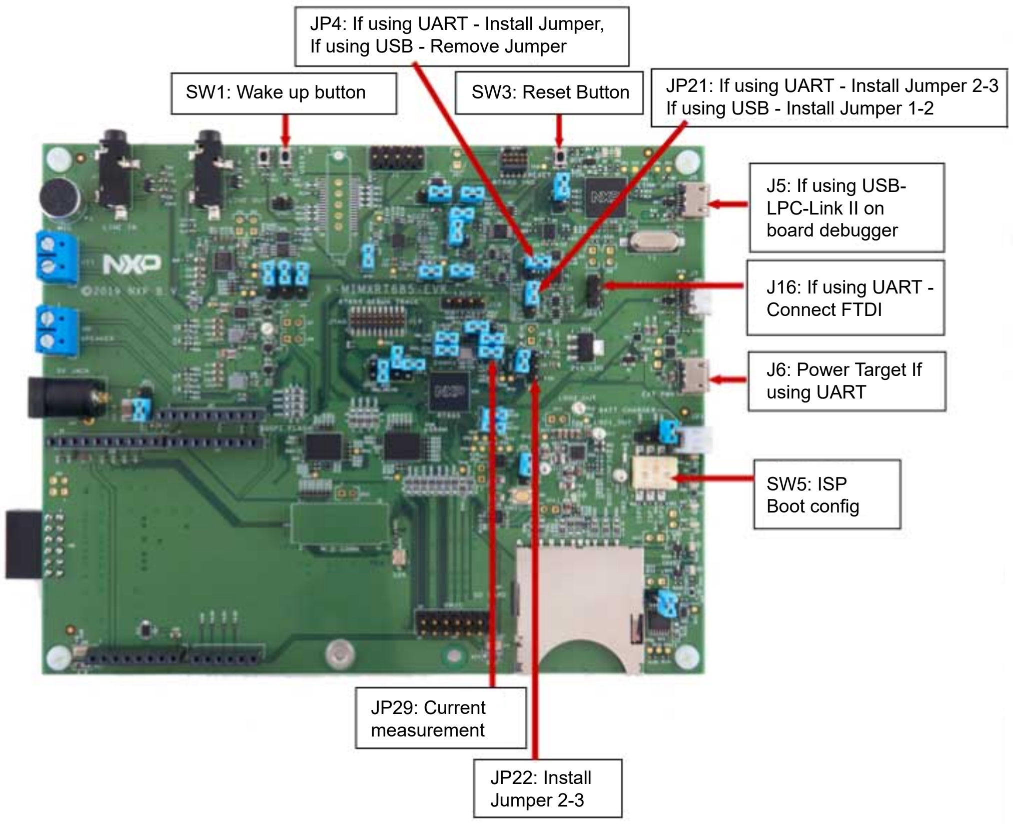

JP4: If using UART - Install Jumper, If using USB - Remove Jumper
SW1: Wake up button
SW3: Reset Button
JP21: If using UART - Install Jumper 2-3 If using USB - Install Jumper 1-2
J5: If using USB-LPC-Link II on board debugger
J16: If using UART - Connect FTDI
J6: Power Target If using UART
SW5: ISP Boot config
JP29: Current measurement
JP22: Install Jumper 2-3

Figure 1. Hardware setup

4.3 Low-power mode power consumption measurement steps

1. If, using UART with FTDI follow below steps:
    - Install jumper on JP4.
    - Install jumper on JP21 2 - 3.
    - Install jumper to JP22 position 2 - 3.
    - Connect FTDI cable to J16.
    - Connect current meter to JP29, as shown in Figure 2.
    - Connect power from J6 to PC through USB cable.
2. If, using USB follow the below steps:
    - Remove jumper from JP4.
    - Install jumper on JP21 1 - 2.
    - Install jumper to JP22 position 2 - 3.
    - Use LPC-Link II UCom port.
    - Connect current meter to JP29, as shown in Figure 2.
    - Connect power and LPC-Link II on board debugger from J5 to PC through USB cable.

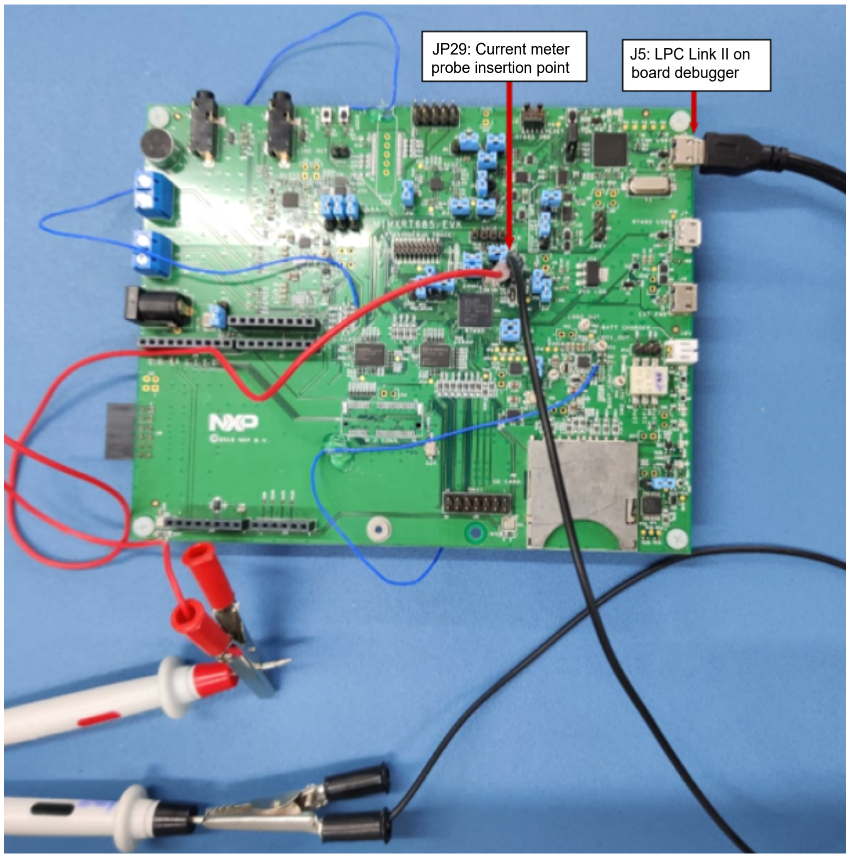
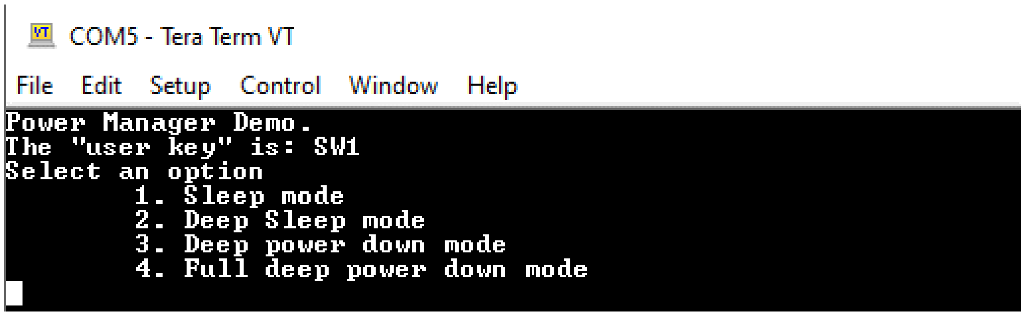

JP29: Current meter probe insertion point
J5: LPC Link II on board debugger

Figure 2. Current measurement setup

3. Start a serial terminal (for example, Tera Term) select RT600 serial port and use the settings for serial console as mentioned, baud rate 115200, no parity, 8 data bits, 1 stop bits.
4. Reset the board using SW3 RESET switch and serial terminal. See Figure 3 for display output.



```
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
```

Figure 3. Console window

5. Enter a number *1* or *2* from the keyboard to put RT600 into sleep or deep sleep mode.
6. Take current measurement from the current meter.

NOTE

Press user SW1 on the board to wake-up device from sleep and deep sleep mode.

7. Repeat steps 3 and 4 to take next measurements after the chip wakes-up from low-power mode.

4.4 Wake-up time measurement steps

The wake-up time for sleep and deep sleep modes can be measured between I/O line (P1_1) being pulled low (on SW1 (USER1) press to wake-up the RT600) and I/O line (P0_31) going low, as this I/O line is being pulled low at 1st instruction inside the SW1 interrupt service routine.

There are two wake-up sources for the RT600 deep power-down and full deep power-down modes as mentioned below.

- RESETN: RT600 can wake-up by providing low pulse to RESETN pin, by pressing RESET switch (SW3).

The wake-up time can be measured between RESETN pin going high (when, press and release RESET (SW3) switch to wake-up the RT600) and I/O line (P0_31) going high, as this I/O line is being pulled high at 15th instruction inside the reset handler.

- PMIC_IRQ_N: The wake-up time can be measured between PMIC_IRQ_N pin going low, (when the low pulse is provided at PMIC_IRQ_N (R85) to wake-up the RT600) and I/O line (P0_31) going high, as this I/O line is being pulled high at 15th instruction inside the reset handler.

For full deep power-down mode: RT600 can wake-up from full deep power-down mode by providing low pulse to PMIC_IRQ_N (R85) pin, this can be achieved by attaching PMIC_IRQ_N pin to GND for 1 or 2 seconds.

For deep power down-mode: To wake-up the RT600 from deep power-down mode using PMIC_IRQ_N, follow the process as mentioned below.

1. P2_12 pin is designed to act as flash reset pin in EVK, so to configure P2_12 as reset pin change OTP efuse value using blhost command-line utility.
2. For using P2_12 as reset, BOOT_CFG1 OTP word should be programmed. For more detail on FlexSPI boot configurations, refer section 41.5.1.2 available in *RT6xx User manual* (document UM11147).
3. Use below blhost command from Windows PowerShell to set P2_12 GPIO as reset pin and enable the flexspi reset pin to reset the flash device. Refer to #unique_43/unique_43_Connect_42_FIG_Z32_B1H_ZNB that shows commands used.

   To use blhost, connect USB cable between port J7 of RT600 and PC and set ISP boot pins ISP0, ISP1 and ISP2 to ON, OFF and OFF respectively.

   NOTE

   When writing in OTP make sure not to modify/update any other OTP word, as the bit once programmed using below command cannot be reverted. It is recommended to read OTP word before and after programming to verify the required bits are updated after programming.

   ```
   ./blhost --usb 0x1fc9,0x0020 --verbose -- efuse-program-once 0x61 0x314000
   ```

4. Once the efuse is programmed, make the ISP pins configurations as default (ON-OFF-ON) and follow the details mentioned in Wake-up time measurement from deep power-down and full deep power-down modes.

4.4.1 Wake-up from sleep and deep sleep mode

Use the following steps for the wake-up time measurement.

1. Connect P1_1 wake-up pin to an oscilloscope probe as trigger input. P1_1 can be accessed from R10, C3, SW1.
2. Connect P0_31 to another oscilloscope probe. P0_31 can be accessed from R398 and Q4.
3. Connect both oscilloscope probes with GND on J25. Refer to the Figure 4 for connection details. Once the connections are done power up the board by connecting USB cable between J5 (if using USB) or J6 (if using UART) and PC.
4. Select sleep or deep sleep as low-power mode by entering *1* or *2* from the keyboard to put the board into low-power mode.
5. Press SW1 USER_1 switch to wake-up the RT600.

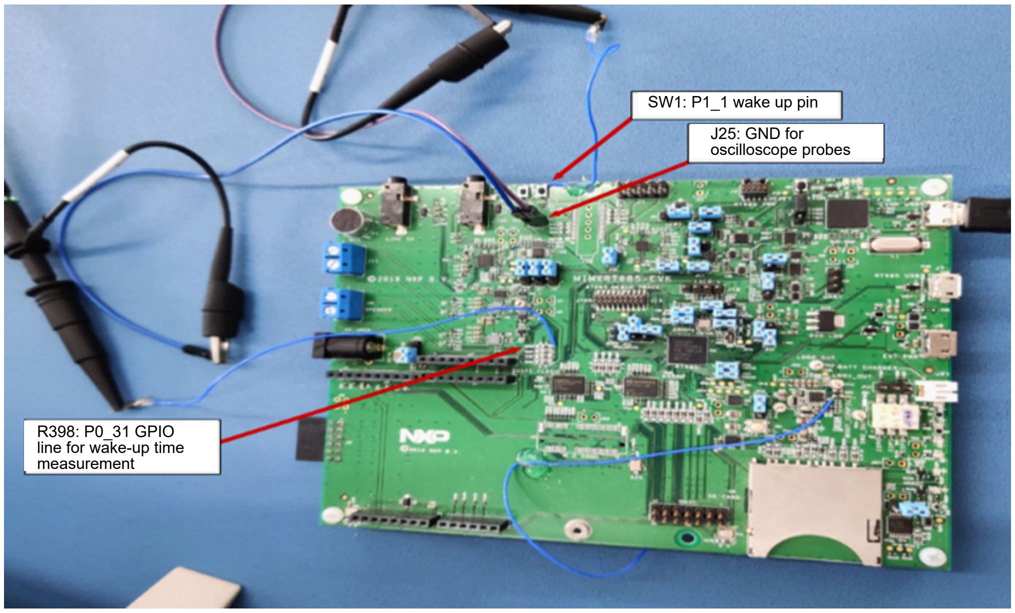

SW1: P1_1 wake up pin
J25: GND for oscilloscope probes
R398: P0_31 GPIO line for wake-up time measurement

Figure 4. Wake-up time measurement setup for sleep and deep sleep mode

6. For wake-up time from sleep and deep-sleep modes, measure time passed on the oscilloscope between the P1_1 wake-up pin going low to P0_31 pin going low as shown in Figure 5 and Figure 6.

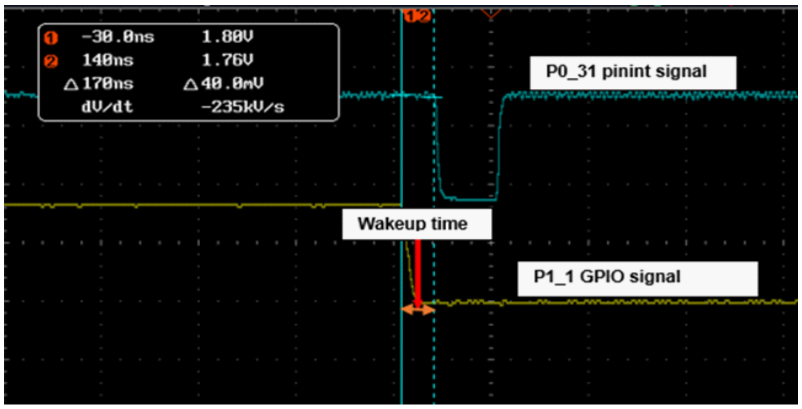


Figure 5. Wake-up time measurements for sleep mode

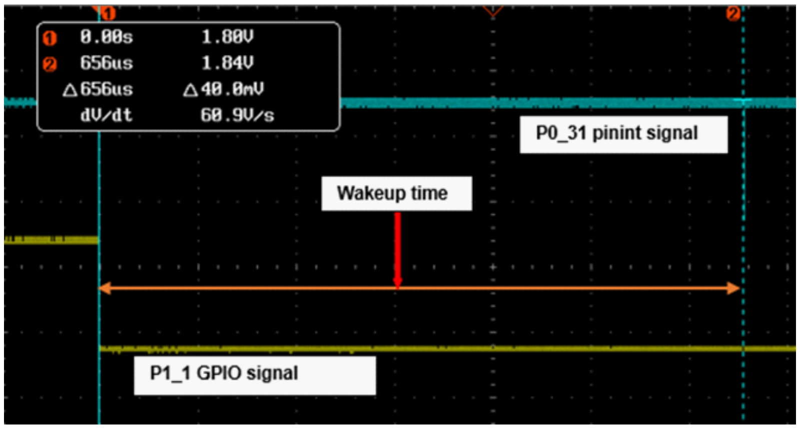


Figure 6. Wake-up time measurements for deep sleep mode

4.4.2 Wake-up time measurement from deep power-down and full deep power-down modes

Use the following steps for the wake-up time measurement:

1. Connect RESETN/PMIC_IRQ_N pin to an oscilloscope probe as trigger input. RESETN and PMIC_IRQ_N can be accessed from JP16 3 and R85 respectively.
2. Connect P0_31 to another oscilloscope probe. P0_31 can be accessed from R398, Q4.
3. Connect both oscilloscope probes with GND on J25. Refer Figure 7 for connection details. Once the connections are done power up the board by connecting USB cable between J5 (if using USB) or J6 (if using UART) and PC.

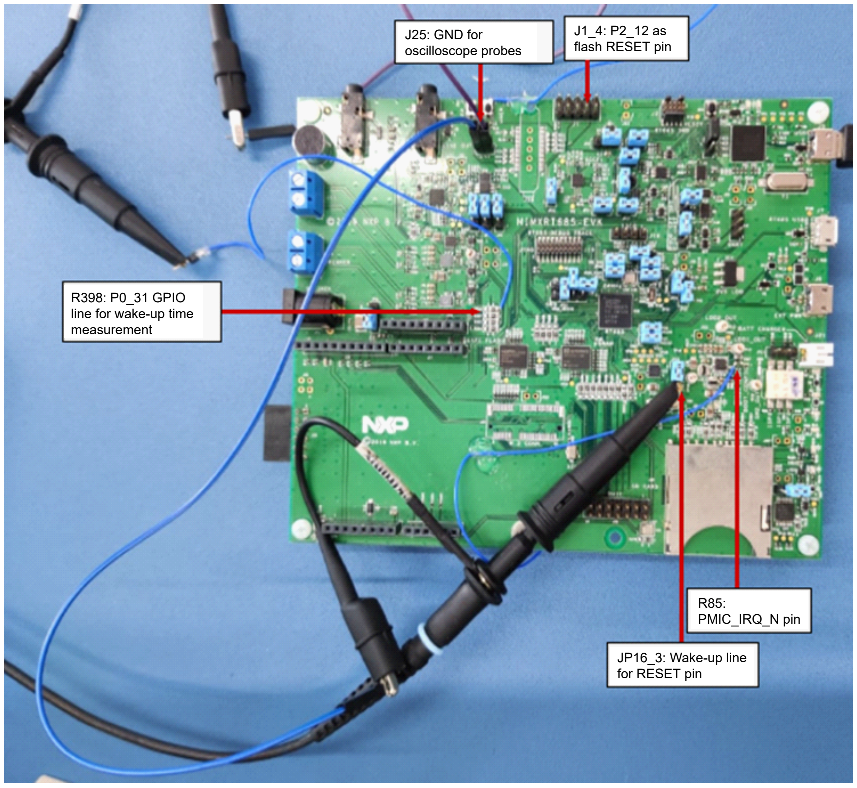

J25: GND for oscilloscope probes
J1_4: P2_12 as flash RESET pin
R398: P0_31 GPIO line for wake-up time measurement
R85: PMIC_IRQ_N pin
JP16_3: Wake-up line for RESET pin

Figure 7. Wake-up time measurements setup for deep power-down and full deep power-down modes

4. Select deep power-down or full deep power-down as low-power mode by entering 3 or 4 from the keyboard.
5. After entering 3 or 4, press any other key on the keyboard to confirm the device to enter deep power-down and full deep power-down mode.
6. For **RESETN** as a wake-up source.
   - Press SW3 RESET BUTTON switch to wake up the RT600.
   - For wake-up time from deep power-down and full deep power-down modes, measure time passed on the oscilloscope between the RESETN pin going high to P0_31 pin going high as shown in Figure 8 and Figure 9.

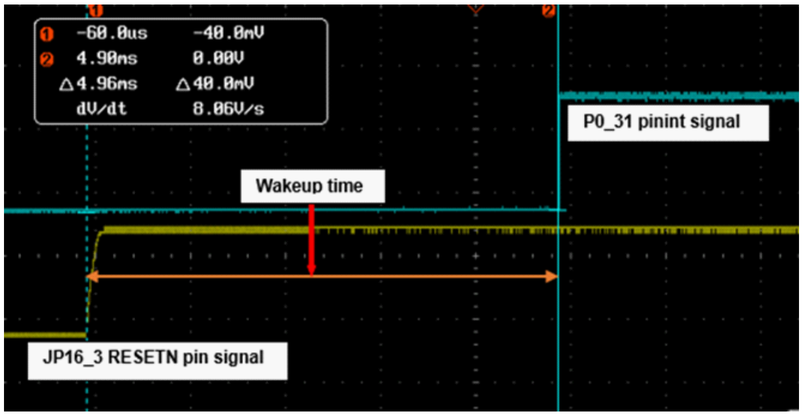


Figure 8. Wake-up time measurement for deep power-down using RESETN pin

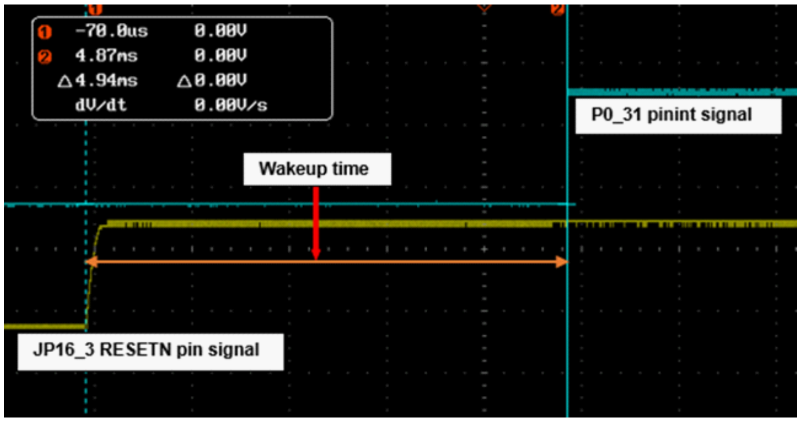


Figure 9. Wake-up time measurement for full deep power-down using RESETN pin

OR

7. For PMIC_IRQ_N as a wake-up source.
   - Skip this step in the case of full deep power-down mode. Attach P2_12 (J1_4) to GND for 1 second (deep power mode only).

     The bootloader can perform the reset process and reset the Flash device to 1-bit SPI compatible mode if the FLEXSPI_RESET_PIN_EN is blown, using the GPIO specified by the combination of FLEXSPI_RESET_PIN_PORT and FLEXSPI_RESET_PIN_GPIO.

     So if we have blown OTP fuse correctly in RT600 EVK, ROM would drive P2_12 to reset flash and this step is not required. But by default, OTP is not fused and user would need to drive P2_12 low.
   - Attach PMIC_IRQ_N (R85) to GND for 1 second to wake-up the RT600.

- For wake-up time from deep power-down and full deep power-down modes, measure time passed on the oscilloscope between the PMIC_IRQ_N pin going low to P0_31 pin going high as shown in Figure 10 and Figure 11.

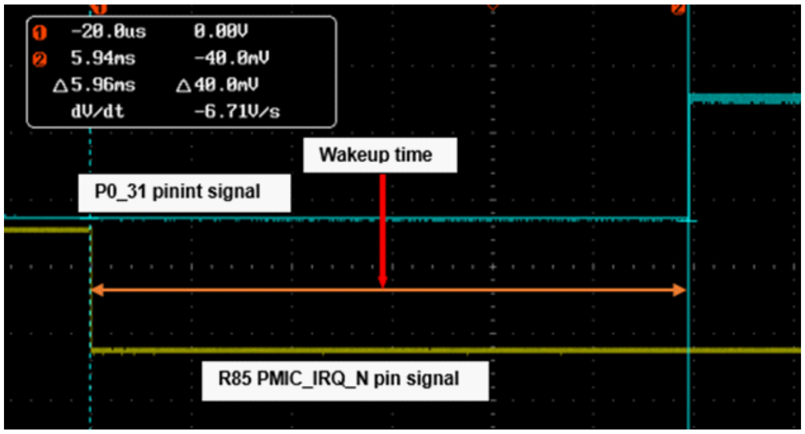


Figure 10. Wake-up time measurements for deep power-down using PMIC_IRQ_N pin

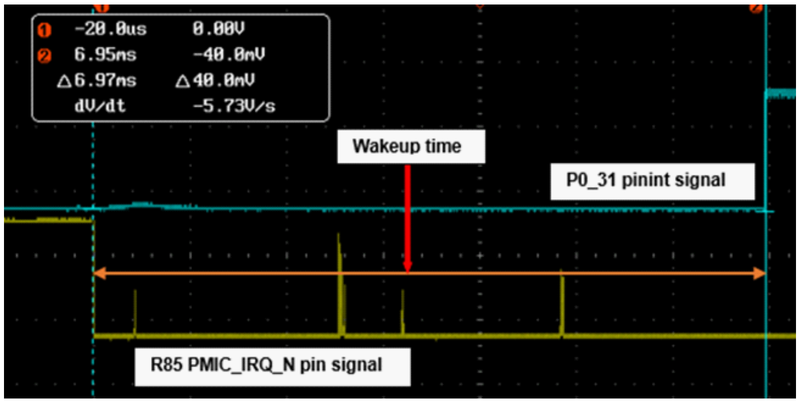


Figure 11. Wake-up time measurements for full deep power-down using PMIC_IRQ_N pin

4.5 Typical low-power mode current and wake-up time

Table 12. Current and wake-up time measurements

Low-power mode	Measurement parameter	Observation	
		Current value	Wake-up time
Sleep	250 MHz	19.87 mA	170 nS
	12 MHz	6 mA	2.8 µS
Deep sleep	Main Crystal Oscillator OFF	89.4 µA	656 µS
	Main Crystal Oscillator ON	107 µA	333 µS
Deep power-down	Reset by RESETN PIN	N/A	4.96 mS

Table continues on the next page...

Table 12. Current and wake-up time measurements (continued)

Low-power mode	Measurement parameter	Observation	
		Current value	Wake-up time
	Reset by PMIC_IRQ_N PIN		5.96 ms
Full deep power-down	Reset by RESETN PIN	N/A	4.94 mS
	Reset by PMIC_IRQ_N PIN		6.97 ms

5 Conclusion

The RT600 provides great flexibility and multiple options for user to achieve low-power consumption with required wake-up configuration. This flexibility allows a trade-off between power consumption and wake-up speed based on the application requirements of the user.

6 References

- *RT6xx User manual* (document UM11147)
- *Dual-core microcontroller with 32-bit Cortex®-M33 and Xtensa HiFi4 Audio DSP CPUs; Up to 4.5 MB SRAM; FlexSPI with cache and dynamic decryption; High-speed USB device/host + Phy; 12-bit 1 Msamples/s ADC; Analog Comparator; Audio subsystems supporting up to 8 DMIC channels; SDIO/eMMC; AES/SHA/Crypto M33 coprocessor; PUF key generation* (document RT600)
- MCUXpresso SDK Release Notes for EVK-MIMXRT685 (can be found inside SDK)
- Getting Started with MCUXpresso SDK for EVK-MIMXRT685 (can be found inside SDK)
- MCUblhost User Guide

7 Revision history

Rev.	Date	Description
1	08 April 2022	• Updated Wake-up time measurement steps • Updated Wake-up time measurement from deep power-down and full deep power-down modes
0	December 2020	Initial release

Legal information

Trademarks