AN11611

LPC5410x Low Power Modes and Wake-up Times

Rev. 1.0 — 04 Nov 2014 Application note

Document information

Info	Content
Keywords	LPC5410x Low Power Modes, Wake up times, Cortex-M4F Cortex-M0+
Abstract	This application note introduces the low power modes of the LPC5410x series and wake-up implementation. The application note also provides software examples to enter the low power modes and demonstrates how to measure the power consumption and wake-up times using the LPC54102 LPCXpresso board.

Revision history

Rev	Date	Description
1	20141104	Initial version.

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. LPC5410x Introduction

The LPC5410x series MCU are dual-core microcontrollers for embedded applications featuring multiple communication interfaces with very low power consumption in active and low power modes. Such applications include sensor hubs. The LPC54102 parts have a Cortex-M4 and a Cortex M0+ core included. The LPC54101 parts only have one Cortex-M4 core. Both cores can operate at frequencies of up to 100 MHz.

The LPC5410x includes up to 512 kB of flash memory and up to 104 kB of SRAM. The peripheral complement includes two SPI interfaces, four USARTs, three Fast-mode Plus I^2C Bus interfaces (also supporting High Speed mode as a slave), a 32-bit, general-purpose State Configurable Timer plus an assortment of other timers, including a Real-Time Clock module with a dedicated oscillator, a 12-channel/12-bit, 4.8MSPS ADC. A DMA controller can service most of the peripherals.

This application note introduces the various low power modes of the LPC5410x series, the steps required to enter the low power modes and wake-up implementation. This application note also provides software examples to enter and wake-up from the low power modes.
This application note covers the following topics:

1. Low Power Modes
2. Enter Low Power Modes
3. Wake-Up Implementation
4. Low power mode demo

2. LPC5410x Low Power Modes Introduction

On the LPC5410x series, there are four reduced power modes: Sleep, Deep-Sleep, Power-down, and Deep Power-down modes. In addition, there are several features unique to the LPC5410x family that allow better power consumption control and faster wake-up times. The following sections cover the power saving modes for the LPC5410x. Section 2.4 introduces the clock gating feature of the SRAM blocks on LPC5410x which can be applied to all four low power modes. This feature is particularly helpful to manage the SRAM retention and Power-down mode current consumption optimization.

This chapter ends with a summary of the peripheral states when the part enters a particular low power mode.

2.1 Sleep mode

In Sleep mode, the system clock to the ARM Cortex-M4 and Cortex-M0+ core is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the AHBCLKCTRL0 or AHBCLKCTRL1 registers, continue operation during Sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

2.2 Deep-sleep and Power-down Mode

In Deep-sleep and Power-down mode, the system clock to the processor is disabled as in Sleep mode. The main clock and therefore all peripheral clocks are disabled. Most analog blocks are powered down by default (such as the IRC and the ADC) but can be

selected to keep running through the power API if needed as wake-up sources. See Section 3.4 for details on which analog peripherals are turned off in Deep-sleep and Power-down modes.

Deep-sleep and Power-down mode eliminate power used by analog peripherals and all dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and selected SRAM sections are retained and the logic levels of the pins remain static. Sections of SRAM retention can be defined through the power API.

In Deep-sleep mode, the peripherals receive no internal clocks. The flash is in stand-by mode and the entire SRAM is retained by default. User can also optionally power down unused SRAM blocks in their application for additional power saving. In Power-down mode, only 8kB of SRAM is retained by default and the flash are powered down, decreasing power consumption compared to Deep-sleep mode. The user can also optionally power on more SRAM blocks in their application for better data retention. Wake-up times from Power-down mode are longer compared to the Deep-sleep mode.

2.2.1 Wake-up to SRAM from Power-down Mode

This low power mode is entered when the flash is turned off, the wake-up routine will wake to execute from SRAM avoiding the flash startup time thus drastically reducing the wake-up time. This very useful application use case is explained and is demonstrated in this application note.

2.3 Deep Power-down Mode

For maximal power savings, the entire system is shut down except for the PMU and the RTC domain. Only the general purpose registers in the PMU and the RTC registers are powered and can maintain their internal states Notice that the LPC5410x does not have a dedicated Wake-up pin. The part can wake-up on a pulse on the Reset pin or on an interrupt from the RTC alarm. On wake-up, the part reboots.

2.4 Individually Clocked SRAM Blocks

The clock gating feature implemented on the three SRAM blocks gives more granularities in the control of the low power mode power consumption. Table 1 shows where the three sections of SRAM reside in the LPC5410x MCU memory map and the associated power switches.

Table 1. SRAM Memory Layout

	SRAM0	SRAM1	SRAM2
Size	Up to 64 kB	Up to 32 kB	8 kB
Address range	Begins at 0x0200 0000	Begins at end of SRAM0	Begins at 0x0340 0000
Power Control (via Power API)	First 8kB has a separate switch Remaining SRAM0 has a single power switch	Entire SRAM1 has a single power switch	Entire SRAM2 has a single power switch

There are four switches in the PDRUNCFG register that allow each of the four individual SRAM sections to be retained.

Table 2. PDRUNCFG register (address 0x4000 0210) bit description (0 = Powered; 1 = Powered down)

Bit	Symbol	Description	value
2:0	-	-	-
3	PDEN_IRC_OSC	IRC oscillator output;	
4	PDEN_IRC	IRC oscillator	0
5	PDEN_FLASH	Flash memory	0
6	-	Reserved	
7	PDEN_BOD_RST	Brown-out Detect reset;	0
8	PDEN_BOD_INTR	Brown-out Detect interrupt;	1
9	-	Reserved	-
10	PDEN_ADC0	ADC0. 0 = Powered; 1 = Powered	1
12:11	-	Reserved	-
13	PDEN_SRAM0A	First 8kB of SRAM0;	0
14	PDEN_SRAM0B	Remaining portion of SRAM0	0
15	PDEN_SRAM1	SARM1;	0
16	PDEN_SRAM2	SRAM2;	0
17	PDEN_ROM	ROM;	0
18	-	Reserved	-
19	PDEN_VDDA	Vdda to the ADC, must be enabled for the ADC to work. Also see bit 23.	1
20	PDEN_WDT_OSC	Watchdog oscillator;	1
21	-	Reserved	-
22	PDEN_SYS_PLL	PLL;	1
23	PDEN_VREFP	Vrefp to the ADC, must be enabled for the ADC to work. Also see bit 19;	1
24	PDEN_32K_OSC	32kHz RTC oscillator	0
31:25	-	Reserved	-

2.5 Low Power Modes Summary

Table 3 summarizes which parts of the chip are affected by the various power saving modes.

Table 3. Core/Peripheral States in Low Power Modes

Core/Peripheral States	Sleep	Deep-sleep	Power-down	Deep power-down
Core-M4	off	off	off	off
Core-M0+ (if available)	off	off	off	off
IRC	Configurable[1]	on	off	off
Flash	Configurable[1]	standby	off	off
First 8kB of SRAM0	on	Configurable[2]	on	off
Remaining SRAM0	Configurable[1]	Configurable[2]	Configurable[2]	off
SRAM1	Configurable[1]	Configurable[2]	Configurable[2]	off

Core/Peripheral States	Sleep	Deep-sleep	Power-down	Deep power-down
SRAM2	Configurable[1]	Configurable[2]	Configurable[2]	off
IO pins	on	on	on	off
BOD	Configurable[1]	Configurable[2]	Configurable[2]	off
ROM	Configurable[1]	off	off	off
PLL	Configurable[1]	off	off	off
ADC	Configurable[1]	Configurable[2]	Configurable[2]	off
WDosc/WWDT	Configurable[1]	Configurable[2]	Configurable[2]	off
RTC	Configurable[1]	Configurable[2]	Configurable[2]	Configurable[3]
Other peripherals	Configurable	Configurable[2]	Configurable[2]	off

Notes for the lower modes comparison in Table 3.

1) As an additional power saving measure in **active** and **Sleep** mode, the power to these components can be switched on/off via the Power Configuration Register (PDRUNCFG).

2) While the analog power is removed from most peripheral components in **Deep-sleep** and **Power-down** mode, most analog and digital blocks can be configured to provide a wake-up service for the CPU, will remain active at a lower power level. Section 3.4.3 provides more details on this usage.

3) The RTC oscillator can be used to wake-up the CPU via interrupt from **sleep**, **Deep-sleep** mode and **Power-down** mode or **Deep Power-down** mode to provide a timed reset for the CPU from deep power-down mode

3. Entering Low Power Modes and Waking up

The power management unit (PMU) manages the low power mode entry and wake-up of the core. The PMU is always on as long as VDD is present. The LPC54102 features a comprehensive ROM Power API system that can manage power consumption for active, sleep as well as other low power modes (Deep-sleep, Power-down, Deep power-down). This application note describes how to enter and wake-up from the various low power modes using the ROM power APIs.

3.1 ROM Power API

The API calls to the ROM are performed by executing functions which are pointed by a pointer within the ROM Driver Table as shown in Fig. 1.

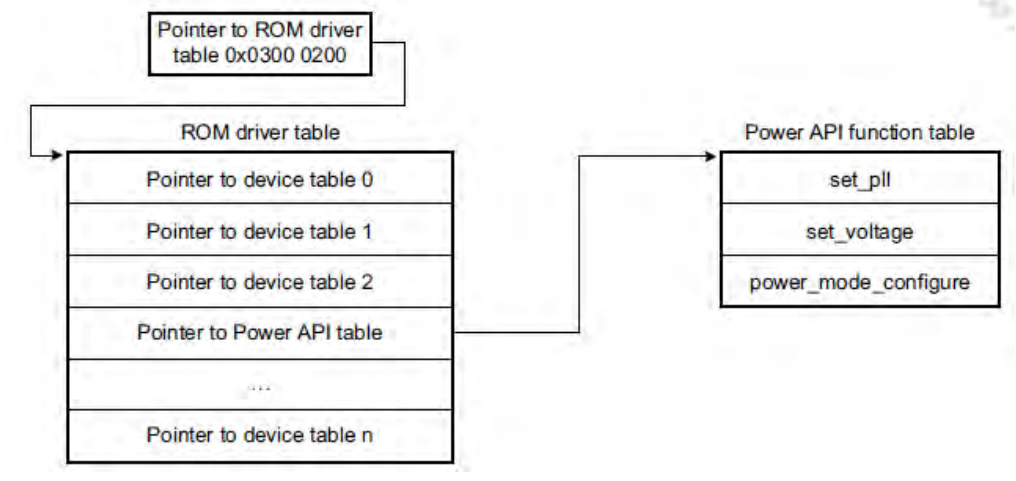

Fig 1. Power API

The Power APIs provide functions to configure the system clock and set up the system for expected performance requirements as explained in Table 4.

Table 4. Power API ROM calls

API call	Description
unsigned int set_pll (multiplier, input_freq)	PLL configuration routine. This API sets up basic PLL operation
unsigned int set_voltage (mode, desired_freq)	Internal voltage configuration routine. This API configures the internal regulator for the desired active operating mode and frequency.
void power_mode_configure (mode, peripheral)	Power mode configuration routine. This API prepares the chip for sleep, deep-sleep, power-down, or deep power-down mode and selects which peripherals can wake-up the part from deep-sleep or power-down modes.

The following elements have to be defined in an application that uses the ROM power profiles:

```
typedef struct _PWRD {
unsigned int (*set_pll)(unsigned int multiplier, unsigned int
input_freq);
unsigned int (*set_voltage)(unsingned int mode, unsigned int
desired_freq);
void (*power_mode_configure)(unsigned int mode, unsigned int
peripheral);
} PWRD;
#define rom_driver_ptr (*(ROM) **) 0x0300 0200)
pPWRD = (PWRD *)(rom_driver_ptr->pPWRD);
```

Fig 2. Using the power API in your application

3.1.1 Power_mode_configure

This application note uses only the power_mode_configure (mode, peripheral) API. The power_mode_configure API prepares the part to enter any of the low power modes. Specifically for the deep-sleep and power-down modes, this API function configures which analog components remain running in those two modes, so that an interrupt from one of the analog peripherals can wake-up the part.

Table 5. Power_mode_configure routine

Routine	Power_mode_configure
Prototype	Void power_mode_configure (mode, peripheral);
Input parameter	Param0: mode (see Table 5) Prama1: peripheral
Result	None

Remark: In addition to the analog peripherals listed with this parameter, the serial peripherals can also wake-up the chip from deep-sleep or power-down modes on an interrupt triggered by an incoming signal. This wake-up scenario is not configured using the power_mode_configure API. For details, please refer to the user manual chapters for USART, SPI and I2C.

3.1.2 Power_mode_configure input parameters

Param0: mode

The mode parameter defines the low power mode and prepares the chip to enter the selected mode. The following modes are valid:

```
#define POWER_SLEEP              0
#define POWER_DEEP_SLEEP         1
#define POWER_POWER_DOWN         2
#define POWER_DEEP POWER_DOWN    3
```

Fig 3. Low Power Modes

Param1: peripheral

The peripheral parameter is a 32-bit value that corresponds to the PDRUNCFG register (as shown in Table 2). If sleep mode (*POWER_SLEEP*) is selected with the mode parameter, the peripheral parameter is ignored.

The peripheral parameter defines which analog peripherals remain running in deep-sleep or power-down modes. Some of these analog peripherals can wake-up the chip from deep-sleep, power-down, or deep power-down mode. For example, the RTC can be selected to remain running thus the RTC alarm wake-up the part from deep-sleep, power-down or deep power-down mode.

The LPCOpen software package defines below macros to allow easy selection of which analog peripheral to be on at Deep-sleep and Power-down modes.

```
/ * Power control definition bits (0 = powered, 1 = powered down)*/
#define SYSCON_PDRUNCFG_PD_IRC_OSC      (1 << 3)        /*!< IRC oscillator output */
#define SYSCON_PDRUNCFG_PD_IRC          (1 << 4)        /*!< IRC oscillator */
#define SYSCON_PDRUNCFG_PD_FLASH        (1 << 5)        /*!< Flash memory */
#define SYSCON_PDRUNCFG_PD_BOD_RST      (1 << 7)        /*!< Brown-out Detect reset */
#define SYSCON_PDRUNCFG_PD_BOD_INTR     (1 << 8)        /*!< Brown-out Detect interrupt */
#define SYSCON_PDRUNCFG_PD_ADC0         (1 << 10)       /*!< ADC0 */
#define SYSCON_PDRUNCFG_PD_SRAM0A       (1 << 13)       /*!< First 8 kB of SRAM0 */
#define SYSCON_PDRUNCFG_PD_SRAM0B       (1 << 14)       /*!< Remaining portion of SRAM0 */
#define SYSCON_PDRUNCFG_PD_SRAM1        (1 << 15)       /*!< SRAM1 */
#define SYSCON_PDRUNCFG_PD_SRAM2        (1 << 16)       /*!< SRAM2 */
#define SYSCON_PDRUNCFG_PD_ROM          (1 << 17)       /*!< ROM */
#define SYSCON_PDRUNCFG_PD_VDDA_ENA     (1 << 19)       /*!< Vdda to the ADC, must be
enabled for the ADC to work */
#define SYSCON_PDRUNCFG_PD_WDT_OSC      (1 << 20)       /*!< Watchdog oscillator */
#define SYSCON_PDRUNCFG_PD_SYS_PLL      (1 << 22)       /*!< PLL0 */
#define SYSCON_PDRUNCFG_PD_VREFP        (1 << 23)       /*!< Vrefp to the ADC, must be
enabled for the ADC to work */
#define SYSCON_PDRUNCFG_PD_32K_OSC      (1 << 24)       /*!< 32 kHz RTC oscillator */
```

Fig 4. PDRUNCFG register bit macro in LPCOpen package

Changing the contents of PDRUNCFG register should only be completed by writing to the PDRUNCFGSET and/or PDRUNCFGCLR register. The LPCOpen package has API calls as shown with below to take care of this procedure.

```
Chip_SYSCON_PowerDown(SYSCON_PDRUNCFG_PD_VREFP);
Chip_SYSCON_PowerDown(SYSCON_PDRUNCFG_PD_32K_OSC);
```

Fig 5. Power on/off Analog Blocks with LPCOpen Package

3.2 Switch to IRC Clocking before Entering Low Power Modes

Before entering any of the four low power modes, the application code should switch to IRC if not already the current clock source. Upon wake-up from Sleep, Deep-sleep and Power-down modes, application code should recover the intended clock source. See below sample implementation using the LPCOpen package.

```
/* Switch main system clock to IRC and power down PLL */
    saved_clksrc = Chip_Clock_GetMainClockSource();
    if (saved_clksrc == SYSCON_MAINCLKSRC_PLLOUT) {
        Chip_Clock_SetMainClockSource(SYSCON_MAINCLKSRC_IRC);
        Chip_SYSCON_PowerDown(SYSCON_PDRUNCFG_PD_SYS_PLL);
    }
/* On wake-up, restore PLL power if needed */
        if (saved_clksrc == SYSCON_MAINCLKSRC_PLLOUT) {
            Chip_SYSCON_PowerUp(SYSCON_PDRUNCFG_PD_SYS_PLL);
            /* Wait for PLL lock */
            while (!Chip_Clock_IsSystemPLLLocked());
            Chip_POWER_SetVoltage(POWER_LOW_POWER_MODE,
Chip_Clock_GetSystemPLLOutClockRate(false));
            /* Use PLL for system clock */
            Chip_Clock_SetMainClockSource(SYSCON_MAINCLKSRC_PLLOUT);
        }
```

Fig 6. Switch to IRC clocking and Recover Saved System Clock

The following sections assume the switching to IRC is done.

3.3 Entering and Waking Up from Sleep Mode

3.3.1 Entering Sleep Mode

Based on the power_mode_configure API, the sleep mode can be easily entered. The LPCOpen software provides another layer of API with same argument inputs as the power_mode_configure API as shown in below sample code.

```
    /* The 2nd and 3rd parameter don't apply in SLEEP mode. */
    Chip_POWER_EnterPowerMode (POWER_SLEEP,0);
```

Fig 7. Entering Sleep Mode

3.3.2 Waking up from Sleep Mode

Any interrupt can wake-up the part from sleep mode. In this application note, PININT is used to wake-up the part from Sleep mode.

3.4 Entering and Waking Up from Deep-sleep and Power-down Mode

3.4.1 Entering Deep Sleep Mode

Based on the power_mode_configure API, the deep-sleep mode can be easily entered. The LPCOpen provides another layer of API with same argument inputs as the power_mode_configure API as shown in below sample code.

```
Chip_POWER_EnterPowerMode  (POWER_DEEP_SLEEP,0);
```

Fig 8. Entering Deep-sleep Mode

The analog peripherals shown below are turned off when the deep-sleep mode is entered with the second argument of power_mode_configure set to 0.

Table 6. Analog Peripherals that are Defaults to Off in Deep-sleep Mode

Symbol in the PDRUNCFG register	Description
PDEN_IRC_OSC	IRC oscillator output
PDEN_IRC	IRC oscillator
PDEN_BOD_INTR	Brown-out detect interrupt
PDEN_ADC0	ADC0
PDEN_VDDA	Vdda to the ADC
PDEN_ROM	ROM
PDEN_VREFP	Vrefp to the ADC
PDEN_SYS_PLL	PLL0

Depending on the application, certain peripherals, for example the ROM, can be kept active by supplying the bit position to the power API as illustrated in Fig 9.

```
Chip_POWER_PowerModeConfigure(POWER_DEEP_SLEEP,
SYSCON_PDRUNCFG_PD_ROM  );
```

Fig 9. Deep-sleep mode with ROM on

Note that entering Deep-sleep mode with the Power API does not turn off the RTC and WDT. If these blocks are not needed in Deep-sleep mode, the user code can turn off these blocks for further power saving in the application.

3.4.2 Entering Power-down mode

Based on the power_mode_configure API, the power-down mode can be easily entered. The LPCOpen provides another layer of API with same argument inputs as the power_mode_configure API as shown in below sample code.

```
Chip_POWER_EnterPowerMode   (POWER_POWER_DOWN,0);
```

Fig 10. Entering Power-down Mode

All analog peripherals powered off in deep-sleep mode are also powered off by default in power down mode. In addition, the analog peripherals shown in Table 7 are also turned off when the power-down mode is entered with second argument of power_mode_configure set to 0. Notice that only the first 8KB of SRAM0 is active by default when entering power-down mode using the power_mode_configure ROM API.

Table 7. Peripherals turned off by default in power-down mode (in addition to the Table 6)

Symbol in the PDRUNCFG register	Description
PDEN_FLASH	Flash memory
PDEN_BOD_RST	Brown-out Detect reset
PDEN_SRAM0B	Remaining portion of SRAM0 from the first 8kB of SRAM0
PDEN_SRAM1	SRAM1
PDEN_SRAM2	SRAM2
PDEN_WDT_OSC	Watchdog oscillator
PDEN_32K_OSC	32kHz RTC oscillator

Depending on the application, user might choose to keep certain additional analog peripherals on. In code example below, the entire SRAM0 and SRAM1 are configured to be active when entering power-down mode using the power API.

```
Chip_POWER_EnterPowerMode  (POWER_POWER_DOWN,
SYSCON_PDRUNCFG_PD_SRAM0B  | SYSCON_PDRUNCFG_PD_);
```

Fig 11. Entering power-down mode with SRAM0 and SRAM1 on (total 96kB SRAM)

3.4.3 Waking up from Deep-sleep and Power-down Mode

The STARTERP0 and STARTERP1 registers enable an interrupt for wake-up from deep-sleep and power-down modes. Note the particular interrupt also needs to be enabled in the NVIC domain.

Some interrupts are typically used in sleep mode only and will not occur during deep-sleep or power-down modes because relevant clocks are stopped. However, it is possible to enable those clocks (significantly increasing power consumption in the reduced power mode), making these wake-up sources possible.

Whether peripheral interrupts can occur during deep-sleep or power-down modes depends on the peripheral, its configuration, and system setup.

The general procedure to configure waking up from deep-sleep and power-down modes is described in Table 8.

Table 8. Configure Wake-up from Deep-sleep and Power-down Mode

• Setup the wake-up peripheral source
• Enable the corresponding interrupt in the NVIC
• Enable the corresponding interrupt in the STARTERP0 or STARTERP1 register
• Enable the corresponding analog block if needed with the PDRUNCFG register
• Enable the interrupt at NVIC as well – do not forget

Table 9. Start enable register 0 (STARTERP0, address 0x4000 0240)
0 = Wake-up disabled; 1 = Wake-up enabled.

Bit	Symbol	Description	Reset value

Bit	Symbol	Description	Reset value
0	WWDT	WWDT interrupt wake-up.	0
1	BOD	BOD interrupt wake-up.	0
2 [4]	-	Reserved.	-
3	DMA [1]	DMA wake-up	0
4	GINT0	Group interrupt 0 wake-up.	0
5	PINT0 [3]	GPIO pin interrupt 0.	0
6	PINT1 [3]	GPIO pin interrupt 1 wake-up	0
7	PINT2 [3]	GPIO pin interrupt 2 wake-up	0
8	PINT3 [3]	GPIO pin interrupt 3 wake-up	0
9	UTICK	Micro-tick Timer wake-up	0
10	MRT [1]	Multi-Rate Timer wake-up	0
11	TIMER0 [1]	Timer 0 wake-up	0
12	TIMER1 [1]	Timer 1 wake-up	0
13	TIMER2 [1]	Timer 2 wake-up	0
14	TIMER3 [1]	Timer 3 wake-up	0
15	TIMER4 [1]	Timer 4 wake-up	0
16	SCT0 [1]	SCT0 wake-up	0
17	USART0 [2]	USART0 interrupt wake-up	0
18	USART1 [2]	USART1 interrupt wake-up	0
19	USART2 [2]	USART2 interrupt wake-up	0
20	USART3 [2]	USART3 interrupt wake-up	0
21	I2C0 [2]	I2C interrupt wake-up	0
22	I2C1 [2]	I2C interrupt wake-up	0
23	I2C2 [2]	I2C interrupt wake-up	0
24	SPI0 [2]	SPI0 interrupt wake-up	0
25	SPI1 [2]	SPI1 interrupt wake-up	0
26	ADC0_SEQA [1]	ADC sequence A interrupt wake-up	0
27	ADC0_SEQB [1]	ADC0 sequence B interrupt wake-up	0
28	ADC0_THCMP [1]	ADC0 threshold and error interrupt wake-up	0
29	RTC	RTC interrupt wake-up	0
30 [4]	-	Reserved.	-
31	MAILBOX	Mailbox interrupt wake-up. At least one CPU must be running in order for a mailbox interrupt to occur. Not practical to be wakeup source in deep-sleep and power-down mode.	0

Table 10. Start enable register 1 (STARTERP1, address 0x4000 0244)
0 = Wake-up disabled; 1 = Wake-up enabled.

Bit	Symbol	Description	Reset value
0	GINT1	Group interrupt 0 wake-up	0

Bit	Symbol	Description	Reset value
1	PINT4 [3]	GPIO pin interrupt 4 wake-up	0
2	PINT5 [3]	GPIO pin interrupt 5 wake-up	
3	PINT6 [3]	GPIO pin interrupt 6 wake-up	0
4	PINT7	GPIO pin interrupt 7 wake-up	0
7:5 [4]	-	Reserved	-
8	RIT [1]	Repetitive Interrupt Timer interrupt wake-up	0
31:9 [4]	-	Reserved	

Notes:

1. Typically used in sleep mode only since the peripheral clock must be running for it to function.
2. Peripheral interrupt. See particular chapter for details of operation.
3. Not for pattern match
4. Read value is undefined, only zero should be written

3.5 Entering and Waking Up from Deep Power-down Mode

3.5.1.1 Entering Deep Power-down Mode

Using the power_mode_configure, it is very easy to put the part into deep power-down mode. The LPCOpen provides another layer of API with same argument inputs as the power_mode_configure API as shown in below sample code.

```
Chip_POWER_EnterPowerMode   (POWER_DEEP_POWER_DOWN,
SYSCON_PDRUNCFG_PD_32K_OSC);
```

Fig 12. Entering Deep Power-down Mode with RTC Active

3.5.1.2 Wake-up from Deep Power-down Mode Using RTC

Two events can wake-up the part from deep power-down mode. Reset pin low pulse or RTC alarm. Below is a summary of steps to setup RTC alarm to wake-up the part from deep power-down mode.

Fig 13. Setting RTC Alarm to Wake-up from Deep Power-down Mode

3.6 Other power consumption reduction considerations

3.6.1 Turning off unused Analog Peripherals

As an example, in this sample project, the RTC is not used in Sleep and Deep-sleep mode, thus manually turning off the RTC providing additional power saving.

3.6.2 Clock Gating the Digital Peripherals

For example: when the IO configuration is done, the user can shut down the IOCON clock for additional power saving in all low power modes.

3.6.3 Handle unused Pins in Sleep/Deep-sleep/Power-down Modes

It is recommended to configure unused pins to GPIO with pull-up and pull-down disabled. In addition, set the pin to output mode with proper level based on the application for reduced leakage current. The function ConfigureUnusedPins in the wake-up project takes care of setting up the unused pins.

3.7 Debugging Session Considerations with Low Power Modes

The debugging session can stay active in Sleep mode but not in Deep-sleep, Power-down and Deep Power-down mode. A reconnect to the debugger from Deep-sleep, Power-down and Deep Power-down mode can be achieved via several methods depending on the user's application

3.7.1 No Wake-up Scheme Provided in User's Application

In this case, user can reset the part if the application does not directly put the part into the Low power modes. The wake-up sample projects presented in this application note does not put the part into low power mode unless the ISP button is pulled low by the user. Notice this is not entering ISP mode. Usage of the ISP button is for the convenience, any other GPIO in the user application system would work for this purpose.

If some applications, as in the rtc example project included in this application note, the part is repeatedly put to Deep Power-down mode and then is repeatedly woken up by the RTC alarm. In this case, the user needs to put the part into ISP mode to allow

reconnection to the debugger. The ISP mode is determined by the state of the P0_31 pin at boot time. LPC5410x supports ISP via USART0. When P0_31 is sampled as low, the LPC5410x enters ISP mode via UART0. Thus the boot code will run and the user can reattached the debugger at this time.

3.7.2 Wake-up Scheme Provided in User's Application

The user can activate the wake-up action to release LPC5410x from the low power modes to allow reconnection to the debugger.

4. Low Power Mode Demos

This application note provides examples to enter all above mentioned low power modes and waking up from Pin interrupt for Sleep, Deep sleep and Power down mode. For waking up from Deep Power-down mode, the RTC alarm functionality is used. On the LPC5410x, the part can also wake-up from Deep power-down mode from the reset pin. The examples are based on NXP's LPCOpen software platform. The examples are based on NXP's LPCOpen software platform. Keil and LPCXpresso too chains are supported.

4.1 Software Setup

Three IDEs were used to verify the sample projects:

Keil 5.12 with Patch for LPC5410x (included in the AN package).

LPCXpresso 7.50 (downloadable from lpcware.com)

Zip file "\LPC5410x_Keil_LowPowerAN.zip" contains the software for implementation based on Keil IDE. User should unzip the Keil project to a folder for testing. For the Keil project, the CMSIS-DAP on board debugger is used. The LQFP54102 development is preconfigured with CMSIS-DAP capability. The CMSIS-DAP configuration tool is also available on the lpcware.com.

Zip file "LPC5410x_LPCXpresso_LowPowerAN.zip" contains the software implementation based on the LPCXpresso IDE. User should import these projects to a LPCXpresso workspace for testing. For the LPCXpresso projects, the redlink debugger is used. User should jumper JP5 on the LPCXpresso board. Please refer to Fig 14 for the location of JP5.

For the low power mode demo, the board library is bypassed. By default, the chip library takes the IRC as input to PLL and sets the main clock to 100MHz. Before entering any low power mode, the application switches the main clock to IRC and powers down the PLL as mentioned in Section 3.2.

4.2 Hardware

The LPCXpresso LPC54102 development board is used in this implementation for the low power mode entry and exit example.

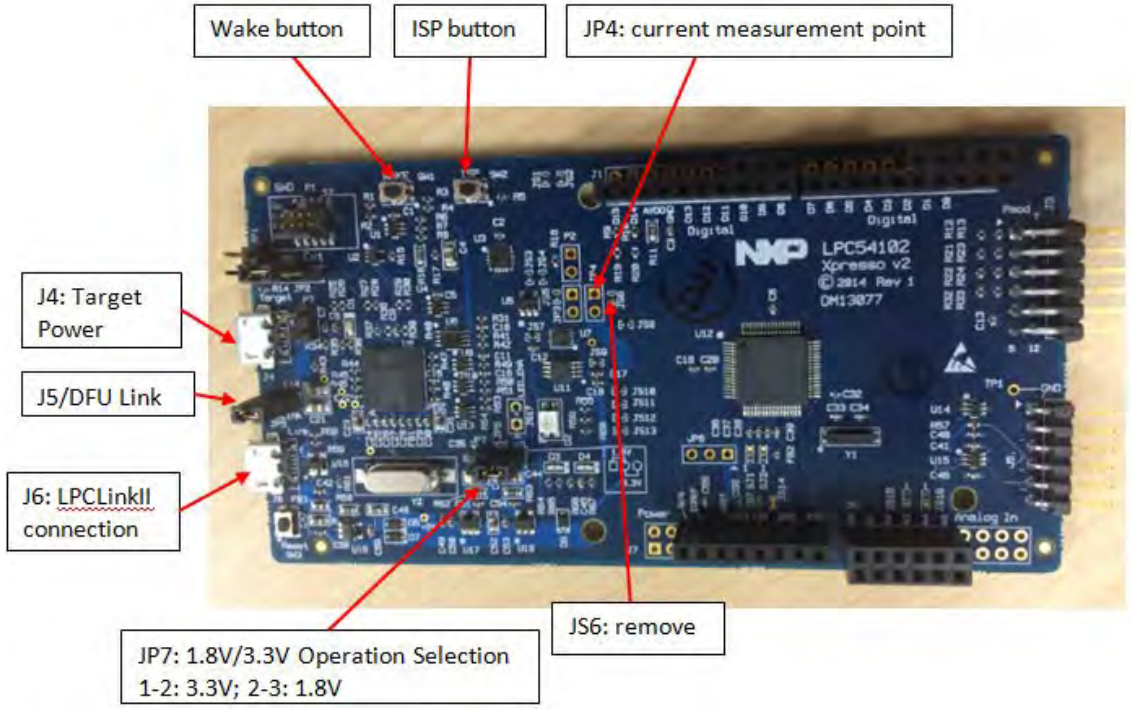

Fig 14. LPC54102 LPCXpresso Development Board

See LPCware.com for the LPC54102 LPCXpresso development board schematic information.

To allow the LPC54102 low power mode power consumption to be measured, remove JS6, install connector at JP4 and insert current meter at JP4 as shown in Fig.15.

In addition, below JS components are removed to reduce the leakage current. If none of these components are removed, users will observe an extra leakage current of about 140uA on top of the specified power consumption for the different low power modes. Please refer to LQFP64 LPCXpresso board silk screen for location of below JS components.

Table 11. Leakage current reduction (Remove below JS components)

Components	Function
JS4	U5 current monitor chip RS- input
JS13	SPI MOSI bridge to Link MCU
JS12	SPI MISO bridge to Link MCU
JS8	SPI SSEL bridge to Link MCU
JS11	Bridge to Link MCU
JS17	RBG LED
JS10	SPI SCK bridge to Link MCU

This development board allows LPC54102 operation at 3.3V or 1.8V. To switch between 3.3V and 1.8V operation, switch jumper on JP7. 1-2 position selects 3.3V and 2-3 position selects 1.8V.

4.3 Low Power Mode Power Consumption Measurement Steps

Please follow below steps for the low power consumption measurement.

1. Connect current meter to JP4 as shown below.

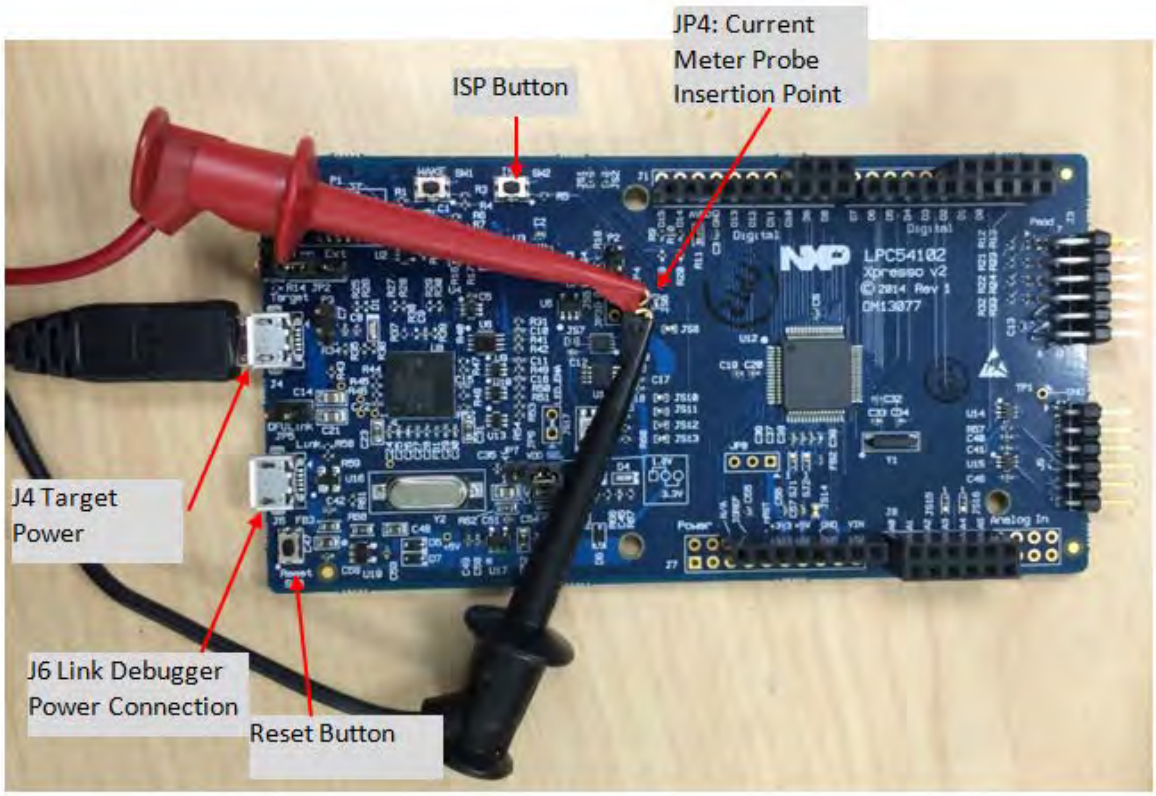

Fig 15. Current Meter Hook up

2. Connect power and Link2 on board debugger from J6 to PC through USB cable.
3. Select which measurement to perform. In below illustration sleep mode is selected.

```
/***********************************/
/* test configuration definition   */
/***********************************/
#define SLEEP_MODE                                  0
#define DEEP_SLEEP_MODE                             1
#define POWER_DOWN_8KB_SRAM0                        2
#define POWER_DOWN_64KB_SRAM0                       3
#define POWER_DOWN_96KB_SRAM0_SRAM1                 4
#define POWER_DOWN_104KB_SRAM0_SRAM1_SRAM2          5
#define DEEP_POWER_DOWN_RTC_ON                      6
#define DEEP_POWER_DOWN_RTC_OFF                     7

/* Below mode only works with the iram target*/
#define SRAM_WAKEUP_TIME_MODE                       8

#define SELECTED_MEASUREMENT            SLEEP_MODE
```

Fig 16. Selection of Low power mode

4. Define test to be Power Consumption Measurement

```
#define CURRENT_MEASUREMENT_MODE        1
```

Fig 17. Define CURRENT_MEASUREMENT_MODE

5. Compile the project and down load the executable to LPC54102.
6. Switch power connection from J6 to J4 (for target power).
7. Press the reset button (SW3).
8. Press the ISP button (SW2). This sample project samples the ISP button before entering any of the low power modes; it will only enter the low power modes if the ISP button is sampled low. Notice that this is using the convenience of having an ISP button on the development board but not for entering ISP mode.
9. Take the current measurement from the current meter.
10. Press the Reset button to wake-up the part from low power modes and reconnect the debugger for the next measurement.

4.4 Wake-up Time Measurement Steps

In this application note, the wake-up time from Sleep, Deep-sleep and Power-down modes is measured from the wake-up IO line (P0_24) being pulled low to the first instruction executed in the wake-up interrupt service routine. This first instruction in the interrupt service routine pulls another IO line (P0_2) from low to high. For Cortex-M4, the maximum interrupt latency is 12 cycles with the zero wait state configuration. Therefore, the measured wake-up time is 1us longer than the LPC54102's true wake-up time.

In addition, an example of entering Power-down mode from SRAM and wake-up to execution in SRAM is provided. This mechanism features an ultra fast wake-up time from Power down mode.

The LPC5410x does not have a dedicated Wake-up pin active in Deep power-down mode. Wake-up from Deep power-down is achieved by the providing a low pulse on the reset pin or use the RTC alarm timer. This application note provides an example of using the RTC alarm timer to wake-up the part from Deep Power-down mode.

4.4.1 Wake-up from Sleep/Deep-sleep/Power-down Mode from Flash Execution

Use the following steps for the wake-up time measurement with Flash target.

1. Install jumper JP4 as shown in Fig. 18.
2. Connect P0_24 wake-up pin to oscilloscope trigger input. P0_24 can be accessed from pin 3 at J1 as shown below.
3. Connect P0_2 to another oscilloscope trace. P0_2 can be access from pin 11 at J2 as shown below.

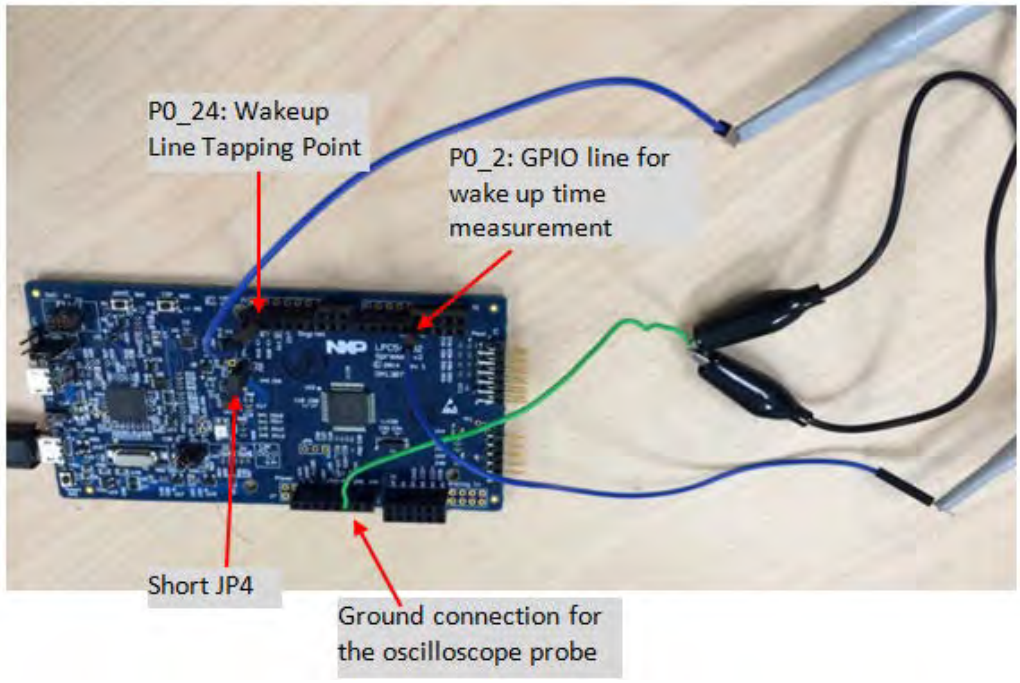

Fig 18. Connect P0_24 and P0_2 for wake-up time measurement

4. Connect power and the on board debugger to a PC through J6.
5. Select which measurement to perform as shown in Fig. 16.
6. Disable the Power Consumption Measurement option.

```
#define CURRENT_MEASUREMENT_MODE      0
```

Fig 19. Disable the Power Consumption Measurement Option

7. Compile the project and download.
8. Switch the USB connection from J6 to J4 (for target power only).
9. Press the reset button (refer to Fig. 14 for location of reset button).
10. Press the ISP button (refer to Fig. 14 for location of ISP button). This sample project samples the ISP button before entering any of the low power modes; it will only enter the low power modes if the ISP button is sampled low. Notice that this is using the convenience of having an ISP button on the development board but not for entering ISP mode.
11. Press the Wake-up button (refer to Fig. 14 for location of Wake-up button).
12. For wake-up time from Sleep, Deep-sleep, Power-down modes, measure time passed on the oscilloscope between wake-up pin (P0_24) going low to P0_2 going high as shown below.

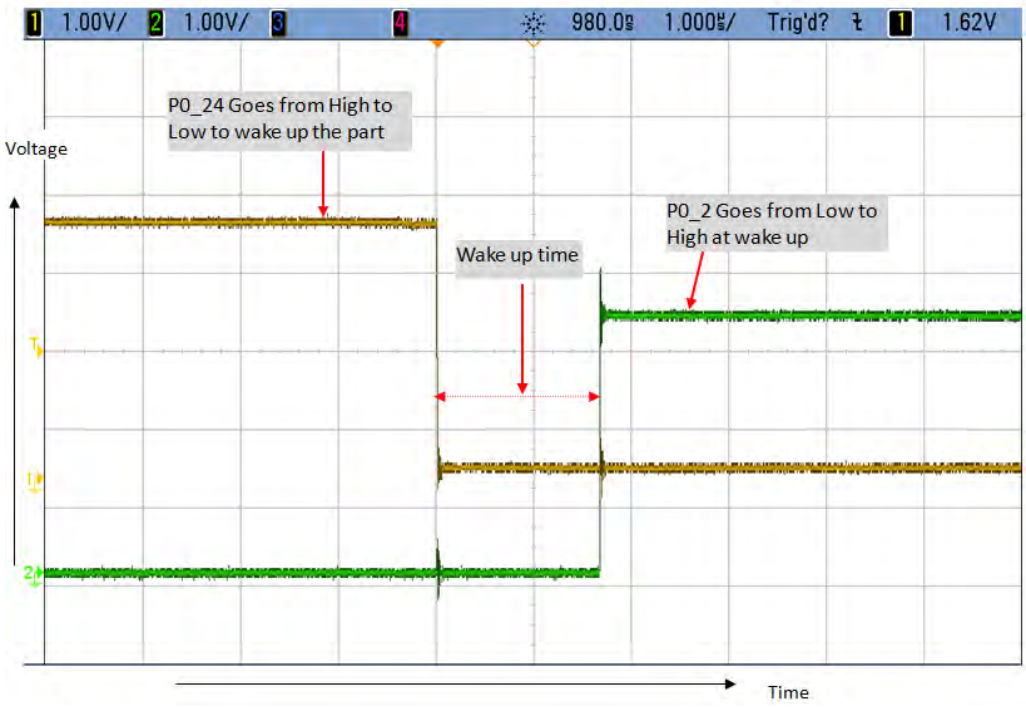

Fig 20. Wake-up Time Measurement from Sleep, Deep-sleep, Power-down Modes

4.4.2 Wake-up from Power-down Mode to SRAM Execution

Turning off the flash whenever possible not only reduces the flash leakage current consumption (which is about 200uA), but also allows a special case of entering the Power-down mode and exit to SRAM execution with a wake-up time in the region of 20us. This is about 50us wake-up time improvement compared to the regular Power-down mode where you enter and wake-up with Flash execution.

On LPC5410x, SRAM0 and SRAM1 are placed on different AHB matrix ports. This allows user programs to potentially obtain better performance by dividing RAM usage among the 2 ports. Such SRAM execution has several beneficial features the user might want to explore in their application.

Only Keil project is implemented to demonstrate the wake-up from Power-down mode to SRAM execution. Follow the steps below for the wake-up time measurement with the SRAM target.

1. Connect current meter to JP4 and Power up the board from J6.
2. Select the iram_nxp_lpcxpresso_lpc5410x from the Keil IDE.

Fig 21. Select the iram_nxp_lpcxpresso_lpc5410x target (Keil example)

3. Select the measurement mode to be *SRAM_WAKEUP_TIME_MODE*

```
#define SELECTED_MEASUREMENT          SRAM_WAKEUP_TIME_MODE
```

Fig 22. Select the Wake-up to SRAM mode

4. Compile and debug the project.
5. Stop the IDE.
6. Press the ISP button (SW2). This sample project samples the ISP button before entering any of the low power modes; it will only enter the low power modes if the ISP button is sampled low. Notice that this is using the convenience of having an ISP button on the development board but not for entering ISP mode.
7. Press the Wake-up button (SW1).

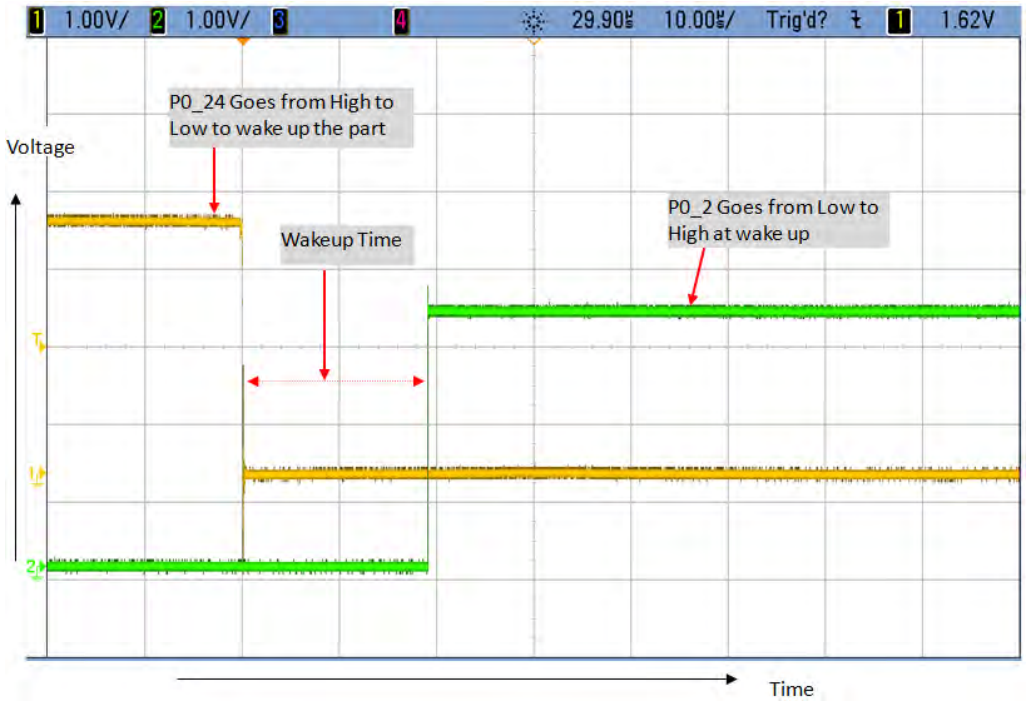

Fig 23. Wake-up from Power-down Mode to SRAM Execution

4.4.3 Wake-up Time Measurement from Deep Power-down Mode

The RTC project establishes a loop where the Deep Power-down Mode is repeatedly entered and the LPC54102 is repeatedly woken up by the RTC alarm. Follow the steps below to verify the system operations.

1. This is a flash based project. User needs to setup the hardware connections as shown in Fig. 18. This project is handling entering and wake-up from Deep Power-down mode only, so no selection of power modes is needed.
2. Compile and download the code.
3. Press the Reset button (SW3).
4. Observe the LPC54102 enters in the Deep Power-down mode when P0_2 is pulled low and wake-up from Deep Power-down mode when P0_2 is pulled high as shown below.
5. To reconnect to the debugger for other downloading or debugging session, the user needs to hold down the ISP button while providing a low pulse on the Reset button. This action puts the LPC54102 to ISP mode where the boot code is run instead of the RTC code. As the boot code is not putting the part into any low power mode without further instructions from the user, the debugger is then able to reconnect to the LPC54102.

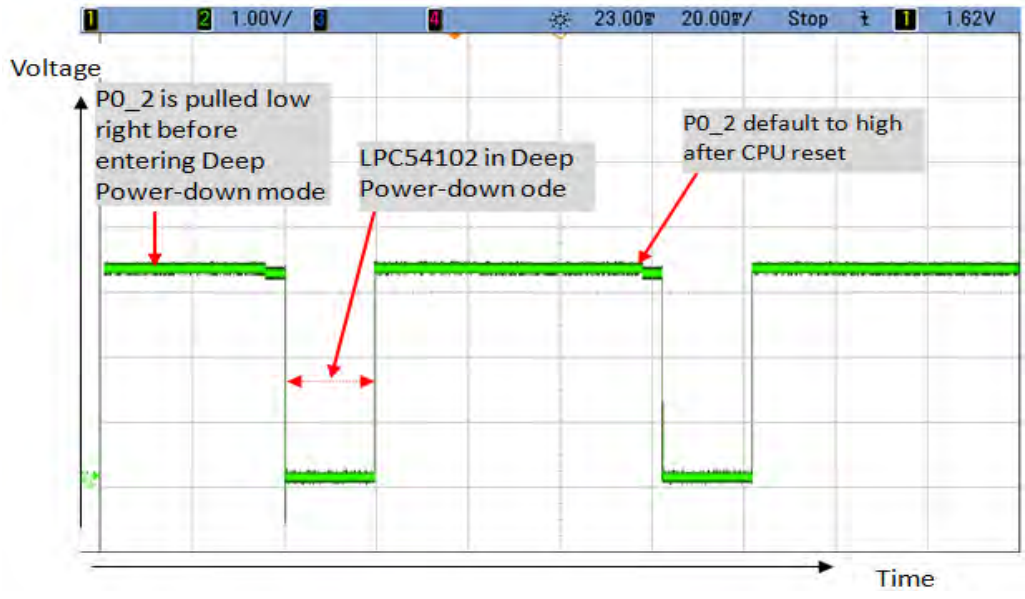

Fig 24. Wake-up from Deep Power-down Mode with RTC Alarm

4.5 Typical Low Power Mode Power Consumption and Wake-up Times

Below are some measurement results taken on a LPCXpresso LPC54102 LQFP64 development board. These wake-up times and power consumption measurement results are very close to those specified in the product datasheet.

Table 12. Low Power Mode Power Consumption and Wake-up Time (VDD=3.3V, Room Temp)

Low Power Mode	SRAM retention	Power VDD=3.3V	Power VDD=1.8V	Wake-up Time (uS) VDD=3.3V	Wake-up Time (uS) VDD=1.8V
Sleep	Default entire 104kB (configurable)	1.21mA	1.16mA	1.6uS	1.6uS
Deep sleep	Default entire 104kB (configurable)	334uA	261uA	17.9uS	22.8uS
Power-down	First 8kB of SRAM0	2.8uA	2.4uA	69.5uS	72.8uS
	64kB SRAM0	5.2uA	5.0uA	Wake-up to SRAM is 18uS	Wake-up to SRAM is 21.4uS
	96kB SRAM0+SRAM1	6.5uA	6.0uA		
	104kB SRAM0+SRAM1+SRAM2	6.9uA	6.4uA		
Deep power-down	No SRAM retention/RTC enabled	400nA	280nA	Depends on RESET line ramp speed (about 200uS)	
	No SRAM retention/RTC disabled	250nA	140nA	Depends on RTC alarm setting (sample code is using a 20mS wake-up RTC alarm)	

5. Conclusion

The LPC5410x series MCU provides great flexibility and multiple options for users to achieve low power consumption and wake-up flexibility. This flexibility allows a trade-off between power consumption and wake-up speed based on the user's application requirements.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. 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 herein and shall have no liability for the consequences of use of such information.

6.2 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.

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.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Trademarks

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

7. List of figures

8. List of tables

9. Contents

For more information, visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 11/04/2014
Document identifier: AN11611