RX Family

LPC Module Using Firmware Integration Technology

Introduction

The RX Family has a variety of options that allow the user to conserve power. This FIT-compliant module provides an API that allows the user to easily configure the RX Family CPUs into their various low power consumption modes and operating power control modes. The module also supports Sleep mode Return Clock Switching. This application note describes the Low Power Consumption (LPC) module API including usage examples.

Target Device

The following is a list of devices that are currently supported by this API:

- RX110, RX111, RX113 Groups
- RX130 Group
- RX210 Group
- RX230, RX231 Groups
- RX64M Group
- RX71M Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- Board Support Package Firmware Integration Technology Module (R01AN1685)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
- Renesas e^2 studio Smart Configurator User Guide (R20AN0451)

Contents

1. Overview

1.1 LPC FIT Module

The LPC FIT module can be used by being implemented in a project as an API. See section 2.12 Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the LPC FIT Module

The operating power control modes for the supported RX CPUs are given in the following table:

Table 1.1 Operating Power Control modes

Supported Operating Power Control modes		
RX110, RX111, RX113, RX130, RX230, RX231	**RX 210**	**RX64M and RX71M**
1. High Speed 2. Middle Speed 3. 4. Low Speed 5.	1. High Speed 2. Middle Speed 1A 3. Middle Speed 1B 4. Low Speed 1 5. Low Speed 2	1. High Speed 2. 3. 4. Low Speed 1 5. Low Speed 2

Each of the operating power control modes has upper and lower limits on the Vcc requirements and the maximum internal clock frequencies supported.

For the RX110, RX111, RX113, for example in High Speed Mode, all the internal clocks can be configured for the system maximum of 32 MHz (when 3.6V > Vcc ≥ 2.7) whereas in Middle Speed Mode, the maximum speeds are limited to 12 MHz (when 3.6V > Vcc ≥ 2.4). In the Low Speed Mode, only the Sub-Clock can be used as the system clock and all internal clocks are limited to a maximum of 32.768 kHz.
The voltage-frequency requirements vary across the RX Family and individual requirements can be found in the specific hardware manual.

In addition to the “Operating” Power Control modes, several Low Power Consumption modes also exist where the CPU is inactive (not operating), namely:

Table 1.2: Low Power Consumption modes

Supported Low Power Consumption modes		
RX110, RX111, RX113, RX130, RX230, RX231	**RX210**	**RX64M and RX71M**
Sleep Deep Sleep Software Standby	Sleep All-Module Clock Stop Software Standby Deep Software Standby	Sleep All-Module Clock Stop Software Standby Deep Software Standby

In each of these modes, certain peripherals are limited or disabled.

1.3 API Overview

Table 1.1 lists the API functions included in this module.

Table 1.3 API Functions

Function	Description
R_LPC_OperatingModeSet	Configures the MCU for the different supported Operating Power Control modes. See Table 1 (Operating Power Control modes).
R_LPC_LowPowerModeConfigure	Configures the MCU for the different Low Power Consumption modes. See Table 2 (Low Power Consumption modes).
R_LPC_LowPowerModeActivate	Enables the Low Power Mode configured by R_LPC_LowPowerModeConfigure()
R_LPC_ReturnClockSwitch	Configures Sleep mode return clock switching
R_LPC_GetVersion	Returns at runtime the driver version number.

1.4 State Transition Diagram

Figure 1.1 to Figure 1.3 shows the state transition diagram for this module.

The following charts show a high level view of the Operating Power Control modes and Low Power Consumption modes as well as the LPC API calls that allow for switching between the modes.

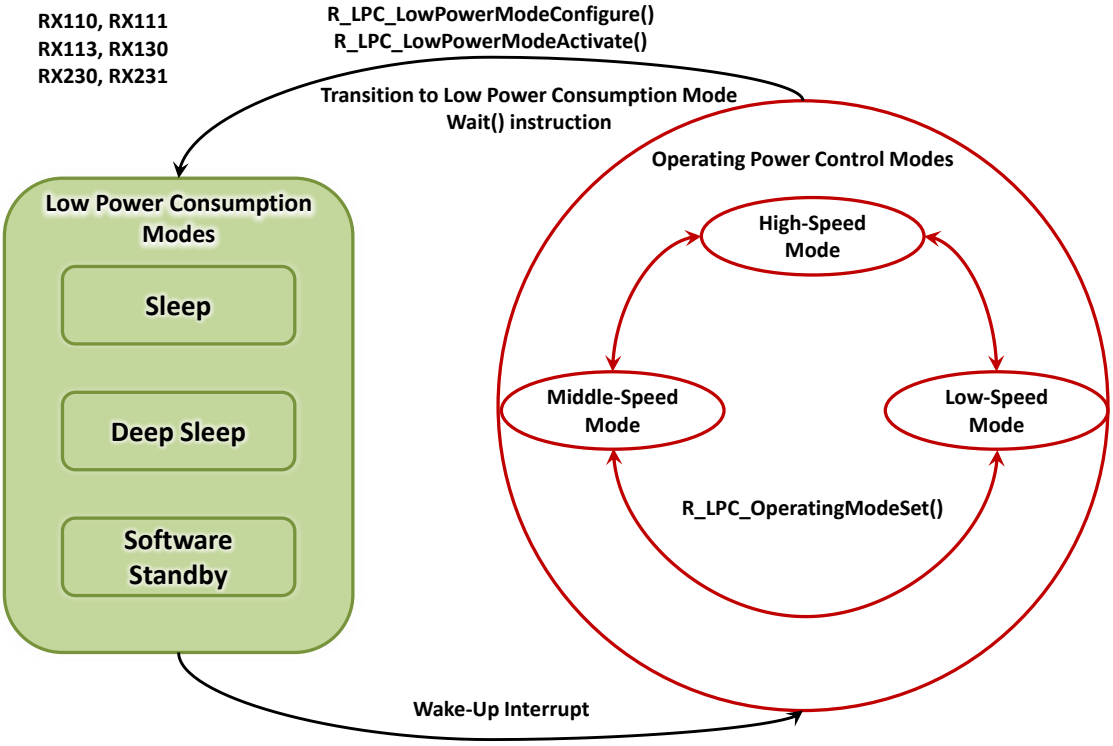

Figure 1.1 LPC API Overview(RX110, RX111, RX113, RX130, RX230, RX231)

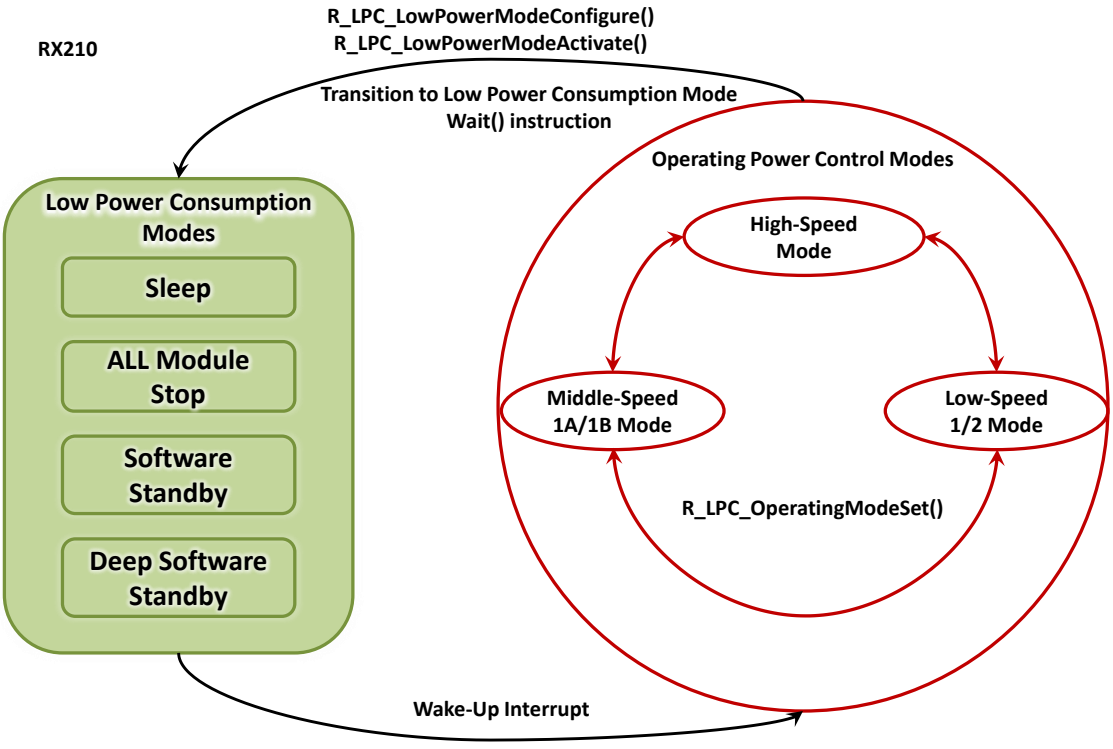

Figure 1.2 LPC API Overview(RX210)

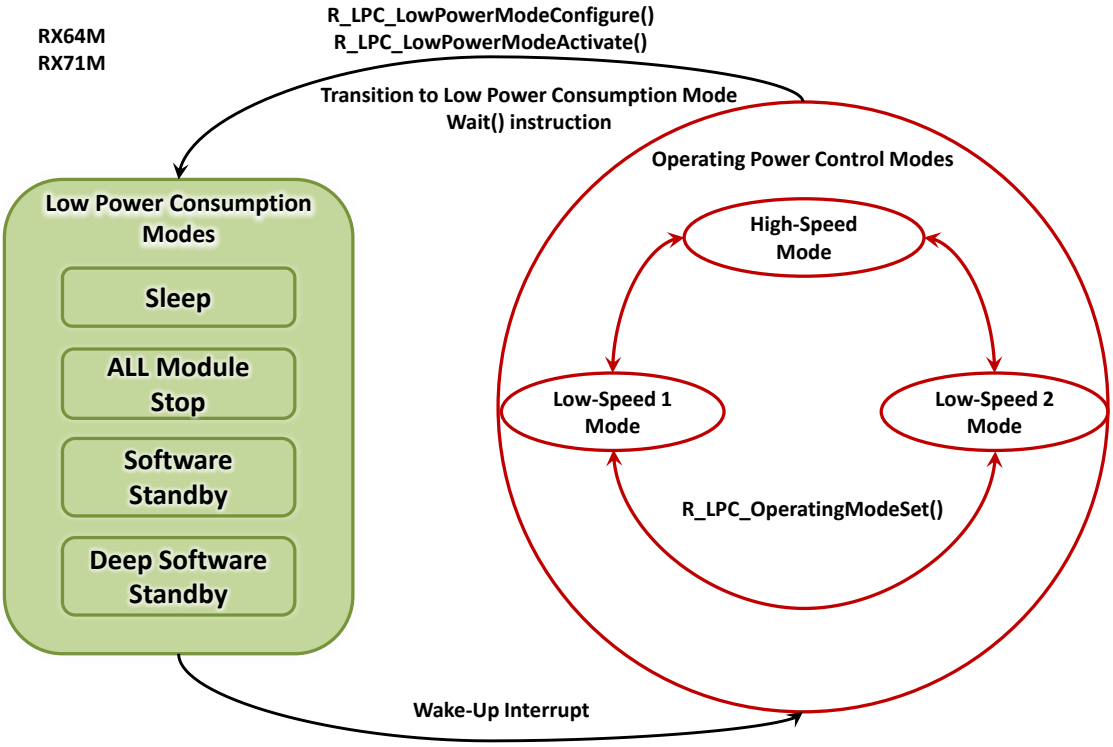

Figure 1.3 LPC API Overview(RX64M, RX71M)

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support the following functions:

- Switching the Operating Power Control Modes given in Table1 in section 1:
- Switching Low Power Consumption Modes given in Table2 in section 1:

2.2 Software Requirements

This driver is dependent upon the following FIT module:

- Renesas Board Support Package (r_bsp)

2.3 Supported Toolchains

This driver has been confirmed to work with the toolchain listed in 6.1, Confirmed Operation Environment.

2.4 Interrupt Vector

This FIT module does not use interrupt vectors.

2.5 Header Files

All API calls and their supporting interface definitions are located in “r_lpc_rx_if.h”.

2.6 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.7 Configuration Overview

The configuration option settings of this module are located in “r_lpc_rx_config.h”. The option names and setting values are listed in the table below:

Table 2.1: Info about the configuration

Configuration options in r_lpc_rx_config.h		
#define	**Default Value**	**Description**
LPC_CFG_PARAM_CHECKING_ENABLE	**1**	If this equate is set to 1, parameter checking is included in the build. If the equate is set to 0, the parameter checking is omitted from the build. Setting this equate to BSP_CFG_PARAM_CHECKING_ENABLE utilizes the system default setting.
(RX210 only) **LPC_CFG_SW_STANDBY_OPTIONS**	**6**	This option is for the RX210 only for configuring the FHSSBYCR register. See section 11.2.18 Flash HOCO Software Control Register in the hardware manual for configuration details.

2.8 Code Size

The sizes of ROM, RAM, and maximum stack usage associated with this module are shown below. Information is presented for a single representative device of the RX100 Series, RX200 Series, RX600 Series, and RX700 Series, respectively.

The sizes of ROM (code and constants) and RAM (global data) are determined by the build-time configuration options described in 2.7, Configuration Overview.

The values in the table below are confirmed under the following conditions.

Module Revision: r_lpc_rx rev1.41

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00

(The option of "-lang = c99" is added to the default settings of the integrated development environment.)

Configuration Options: Default settings

ROM, RAM and Stack Code Sizes (1/2)

Device	Category	Memory Used		Remarks
		With Parameter Checking	Without Parameter Checking	
RX110	ROM:	1610 bytes	371 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	68 bytes	24 bytes	
RX111	ROM:	2287 bytes	371 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	80 bytes	24 bytes	
RX113	ROM:	2293 bytes	371 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	80 bytes	24 bytes	
RX130	ROM:	2285 bytes	371 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	80 bytes	24 bytes	

ROM, RAM and Stack Code Sizes (2/2)				
Device	**Category**	**Memory Used**		**Remarks**
		With Parameter Checking	**Without Parameter Checking**	
RX230, RX231	ROM:	1228 bytes	371 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	52 bytes	24 bytes	
RX64M	ROM:	1235 bytes	342 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	56 bytes	24 bytes	
RX71M	ROM:	1235 bytes	342 bytes	
	RAM:	40 bytes	0 bytes	
	Stack	56 bytes	24 bytes	

2.9 Parameters

This section describes the parameter structure used by the API functions in this module. The enumeration type is located in r_lpc_[device]_if.h(ex: r_lpc_rx64m_if.h) as are the prototype declarations of API functions.

2.9.1 R_LPC_OperatingModeSet Data Types

```
/* Operating power control modes for RX110, RX111, RX113, RX130, RX230, RX231*/
typedef enum lpc operating mode
{
    LPC_OP_HIGH_SPEED   = 0x00,
    LPC_OP_MIDDLE_SPEED = 0x02,
    LPC_OP_LOW_SPEED    = 0x06,
    LPC_OP_INVALID_MODE
} lpc_operating_mode_t;


/* Operating power control modes for RX210 */
typedef enum lpc_operating_mode{
    LPC_OP_HIGH_SPEED      = 0x00,
    LPC_OP_MIDDLE_SPEED_1A = 0x02,
    LPC_OP_MIDDLE_SPEED_1B = 0x03,
    LPC_OP_LOW_SPEED_1     = 0x06,
    LPC_OP_LOW_SPEED_2     = 0x07,
    LPC_OP_INVALID_MODE
}lpc_operating_mode_t;


/* Operating power control modes for RX71M, RX64M */
typedef enum lpc_operating_mode{
    LPC_OP_HIGH_SPEED      = 0x00,
    LPC_OP_LOW_SPEED_1     = 0x06,
    LPC_OP_LOW_SPEED_2     = 0x07,
    LPC_OP_INVALID_MODE
} lpc_operating_mode_t;
```

2.9.2 R_LPC_LowPowerModeConfigure Data Types

```
/* Low Power Modes for RX110, RX111, RX113, RX130, RX230, RX231*/
typedef enum lpc_low_power_mode
{
    LPC_LP_SLEEP,
    LPC_LP_DEEP_SLEEP,
    LPC_LP_SW_STANDBY,
    LPC_LP_INVALID_MODE
} lpc_low_power_mode_t;

/* Low Power Modes for RX210, RX71M, RX64M */
typedef enum lpc_low_power_mode{
      LPC_LP_SLEEP,
      LPC_LP_ALL_MODULE_STOP,
      LPC_LP_SW_STANDBY,
      LPC_LP_DEEP_SW_STANDBY,
      LPC_LP_INVALID_MODE
}lpc_low_power_mode_t;
```

2.9.3 R_LPC_ReturnClockSwitch Data Types

```
/* LPC Sleep Mode Return Clock Switching Sources for RX110, RX111, RX113, RX130,
RX230, RX231*/
typedef enum lpc_clock_switch{
    LPC_LOCO     = 0x00,
    LPC_HOCO     = 0x01,
    LPC_MAIN_OSC = 0x02,
}lpc_clock_switch_t;


/* LPC Sleep Mode Return Clock Switching Sources for RX210, RX71M, RX64M*/
typedef enum lpc_clock_switch{
      LPC HOCO     = 0x01,
      LPC_MAIN_OSC = 0x02,
}lpc_clock_switch_t;
```

2.10 Return Values

This describes the parameter structure used by the API functions in this module. The enumeration type is located in r_lpc_rx_if.h as are the prototype declarations of API functions.

```
/* LPC API error codes */
typedef enum lpc_err
{
  LPC_SUCCESS,
  LPC_ERR_OSC_STOP_ENABLED, // Software Standby cannot be entered if osc stop is enabled.
  LPC_ERR_CLOCK_EXCEEDED,   // Clock exceeds the limit of the operating power control
                            // mode.
  LPC_ERR_ILL_MAIN_CLK_FREQ,// Clock freq. exceeds the limit of the sleep return clock.
  LPC_ERR_ILL_CLOCK_SOURCE, // Illegal clock when sleep mode return clock switching
                            // is enabled
  LPC_ERR_P_E_MODE,         // The operating power control mode cannot be switched while
                            // the flash memory is being programmed or erased (P/E).
  LPC_ERR_DEEP_SLEEP_STATUS,// The condition error for a deep sleep mode.
  LPC_ERR_ILL_PARAM         // (Not used)
  LPC_ERR_ILLEGAL          // Illegal operation
} lpc_err_t;
```

2.11 Callback Function

In this module, the callback function specified by the user is called before transitioning to the low power state.

Please refer to "R_LPC_LowPowerModeActivate ()" in "3 API Functions" for the callback function.

2.12 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e^2 studio
By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio
By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

2.13 “for”, “while” and “do while” statements

In this module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

Target devices describing “WAIT_LOOP”

- RX110, RX111, RX113 Groups
- RX130 Group
- RX210 Group
- RX230, RX231 Groups
- RX64M Group
- RX71M Group

The following shows example of description.

```
while statement example :
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for statement example :
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while statement example :
/* Reset completion waiting */
do
{
    reg = phy read(ether channel, PHY REG CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API Functions

R_LPC_OperatingModeSet ()

This function configures the MCU for the supported Operating Power Control modes (See Table 1).

Format

```
lpc_err_t   R_LPC_OperatingModeSet(
   lpc_operating_mode_t   e_mode
)
```

Parameters

lpc_operating_mode_t e_mode

The modes for all supported MCUs are specified in enum *lpc_operating_mode_t* in section 0.

Return Values

LPC_SUCCESS:
LPC_ERR_CLOCK_EXCEEDED: // Clock exceeds the limit of the operating power control mode.
LPC_ERR_P_E_MODE: // The operating power control mode cannot be switched while
// the flash memory is being programmed or erased (P/E).

Properties

Prototyped in file “r_lpc_rx_if.h”

Description

Depending upon the mode chosen and the MCU, the maximum speed of the internal clocks ICLK, PCLKB, PCLKD and FCLK is limited. For example, in Low-Speed operating mode on the RX110, RX111, RX113, RX130, RX230, RX231, and in Low-Speed operating mode 2 on the RX210, only the sub-clock can be used as the system clock. See Table 11.3 (RX71M, RX64M and RX210) and Table 11.4 (RX110, RX111, RX113, RX130, RX230, RX231) in the Hardware manual for clock limitations. If the argument to this function cannot support the current internal clock frequencies, then an error is returned. When switching the clock source from a lower frequency to a higher frequency, make certain that the operating power control mode is configured to support the range. Failure to do so will result in improper CPU operation.

Reentrant

No

Example

```
lpc_err_t err;

err = R_LPC_OperatingModeSet(LPC_OP_MIDDLE_SPEED);
```

Special Notes:
When switching operating power control modes and internal clock frequencies, it is important to first make sure that the frequencies of internal clocks are set within the range supported by the operating power control mode. When moving between operating power control modes/frequencies, use the following sequence:

1. Moving from low power and low internal frequencies to higher power and higher clock frequencies:
    a. Use R_LPC_OperatingModeSet() to move to higher power operating mode.
    b. Increase internal clock frequencies
2. Moving from high power and high internal clock frequencies to low power and low internal frequencies:
    a. Decrease internal clock frequencies
    b. Use R_LPC_OperatingModeSet() to move to lower power operating mode.

R_LPC_LowPowerModeConfigure ()

This function configures the low power consumption modes (see Table 2) when the WAIT instruction is executed.

Format

```
lpc_err_t   R_LPC_LowPowerModeConfigure(
    lpc_low_power_mode_t e_mode
)
```

Parameters

lpc_low_power_mode_t e_mode

The modes for all supported MCUs are specified in enum lpc_low_power_mode_t.

Return Values

LPC_SUCCESS:

Properties

Prototyped in file “r_lpc_rx_if.h”

Description

This function configures the MCU for the different Low Power Consumption modes shown in Table 2. Note that this function does not activate the low power mode, but configures the registers for the specified mode. To activate the low power mode, use the R_LPC_LowPowerModeActivate() function.

The CPU will be stopped once any of these modes are activated; however a few of the peripherals and clocks can operate in these modes. For more details refer to the User’s Manual: Hardware.

Reentrant

No

Special Notes:

None

R_LPC_LowPowerModeActivate ()

This function activates the Low Power Consumption mode configured in R_LPC_LowPowerModeConfigure ().

Format

```
lpc_err_t   R_LPC_LowPowerModeActivate(
   void (*pcallback)(void* pdata)
)
```

Parameters

*void (*pcallback)(void* pdata)*
Function to be called before activating low power mode.

Return Values

LPC_SUCCESS:
LPC_ERR_OSC_STOP_ENABLED: // Cannot enter software standby if oscillation stop detection
// is enabled.
LPC_ERR_ILL_CLOCK_SOURCE: // Illegal clock when sleep mode return clock switching is enabled
LPC_ERR_ILL_MAIN_CLK_FREQ: // Clock freq. exceeds the limit of the sleep return clock.
LPC_ERR_DEEP_SLEEP_STATUS:// The condition error for a deep sleep mode
LPC_ERR_ILLEGAL // Illegal operation other than above

Properties

Prototyped in file “r_lpc_rx_if.h”

Description

This function activates the low power mode by calling the wait() function. The hardware manual specifies the sequence for entering low power mode as follows:

1. Disable interrupts.
2. Configure the interrupt source to wake the MCU up from the low power mode.
3. Ensure that the last IO register write is successful.
4. Execute the wait instruction to enter the low power mode. The wait instruction will internally enable interrupts.

This function implements the sequence as follows:

1. Disable interrupts.
2. Call the callback function specified by the argument. The callback function should configure the wake-up interrupt source and make sure that the last IO register write is complete before returning. The user can pass a FIT_NO_FUNC pointer if the interrupt has already been configured.
3. Execute the wait instruction.

When this function is executed, an error is returned under the following conditions.

Table 3.1: Limitations and Return Values when Entering Sleep Mode

Limitation	Return Value on an Error	CPU
When sleep mode return clock switching is enabled, the sub-clock must be selected.	LPC_ERR_ILL_CLOCK_SOURCE	RX110, RX111, RX113, RX130, RX230, RX231
When sleep mode return clock switching is enabled, the system clock or the sub-clock must be selected.	LPC_ERR_ILL_CLOCK_SOURCE	RX210, RX64M, RX71M
When the operating power control mode after returning from sleep mode is middle speed mode, HOCO cannot be selected as the return clock.	LPC_ERR_ILL_CLOCK_SOURCE	RX110, RX111, RX113, RX130, RX230, RX231
When the operating power control mode after returning from sleep mode is middle speed mode, and the main clock is selected as the sleep mode return clock, the internal clock must be set to comply with the limitation of the middle speed mode.	LPC_ERR_ILL_MAIN_CLK_FREQ	RX110, RX111, RX113, RX130, RX230, RX231
When HOCO is selected as the sleep mode return clock, set the division ratios of ICLK, FCLK, BCLK, PCLKB, and PCKD to 2 or greater (1/2, 1/4, etc.).	LPC_ERR_ILLEGAL	RX210
When HOCO is selected as the sleep mode return clock, HOCO must be powered on.	LPC_ERR_ILL_CLOCK_SOURCE	RX210, RX64M, RX71M

Table 3.2: Limitations and Return Values when Entering All-Module Clock Stop Mode

Limitation	Return Value on an Error	CPU
The module stop control register must be specified to meet the conditions of all-module clock stop mode.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M
All-module clock stop mode cannot be entered during flash memory P/E mode.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M

Table 3.3: Limitations and Return Values when Entering Deep Sleep Mode

Limitation	Return Value on an Error	CPU
The MSTPCRA.MSTPA28 bit must be set to 1 (transition to the module-stop state for DMAC/ DTC is made) before entering deep sleep mode.	LPC_ERR_DEEP_SLEEP_STATUS	RX110, RX111, RX113, RX130, RX230, RX231
Deep sleep mode cannot be entered during flash memory P/E mode.	LPC_ERR_ILLEGAL	RX110, RX111, RX113, RX130, RX230, RX231

Table 3.4: Limitations and Return Values when Entering Software Standby Mode

Limitation	Return Value on an Error	CPU
When the oscillation stop detection function is enabled, software standby mode cannot be entered.	LPC_ERR_OSC_STOP_ENABLED	RX110, RX111, RX113, RX130, RX230, RX231, RX210, RX64M, RX71M
Enter software standby mode while the DMAST.DMST bit is 0.	LPC_ERR_ILLEGAL	RX230, RX231, RX210, RX64M, RX71M
Enter software standby mode while the DTCST.DTCST bit is 0.	LPC_ERR_ILLEGAL	RX110, RX111, RX113, RX130, RX230, RX231, RX210, RX64M, RX71M
With the sub-clock selected as the clock source of the system clock, operate the RTC (RCR3.RTCEN = 1) or the low-speed on-chip oscillator (LOCOCR.LCSTP = 0) when entering software standby mode	LPC_ERR_ILLEGAL	RX210
Software standby mode cannot be entered during flash memory P/E mode.	LPC_ERR_ILLEGAL	RX110, RX111, RX113, RX130, RX230, RX231, RX210, RX64M, RX71M

Table 3.5: Limitations and Return Values when Entering Deep Software Standby Mode

Limitation	Return Value on an Error	CPU
When the oscillation stop detection function is enabled, deep software standby mode cannot be entered.	LPC_ERR_OSC_STOP_ENABLED	RX210, RX64M, RX71M
Enter deep software standby mode while the DMAST.DMST bit is 0.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M
Enter deep software standby mode while the DTCST.DTCST bit is 0.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M
With the sub-clock selected as the clock source of the system clock, operate the RTC (RCR3.RTCEN = 1) or the low-speed on-chip oscillator (LOCOCR.LCSTP = 0) when entering deep software standby mode	LPC_ERR_ILLEGAL	RX210
Deep software standby mode cannot be entered during flash memory P/E mode.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M
When using IWDT with auto-start mode, deep software standby mode cannot be entered if the OFS0.IWDTSLCSTP bit is 0 (counting stop is disabled).	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M
When using IWDT with register start mode, deep software standby mode cannot be entered if the IWDTCSTPR.SLCSTP bit is 0.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M
When voltage monitoring 1 reset is enabled (LVD1CR0.LVD1RI = 1) or voltage monitoring 2 reset is enabled (LVD2CR0.LVD2RI = 1) in the voltage detection circuit, deep software standby mode is not entered.	LPC_ERR_ILLEGAL	RX210, RX64M, RX71M

Reentrant
No

Example
```
lpc_err_t err;

err = R_LPC_LowPowerModeConfigure (LPC_LP_SLEEP);
err = R_LPC_LowPowerModeActivate(FIT_NO_FUNC);
```

Special Notes:
When parameter checking is enabled for the module, this function checks for a variety of conditions that prevent the MCU from entering a low power mode. While it is important to have this feature enabled in the development phase, it can be disabled during release to allow for a more rapid entry into low power modes.

Before entering Deep Sleep or Software Standby mode, ensure that DTC transactions are not pending and the DTC module is stopped.

R_LPC_ReturnClockSwitch ()

This function configures the MCU to switch clock sources on waking up from Sleep mode.

Format

```
lpc_err_t R_LPC_ReturnClockSwitch(
    lpc_clock_switch_t e_clock_source,
    bool enable
)
```

Parameters

lpc_clock_switch_t e_clock_source

This parameter selects the clock source to be used at the time of release from sleep mode. The supported clock sources are specified in the enum lpc_clock_switch_t in section 2.10.3.

bool enable

Enables or disables clock source switching at the time of release from sleep mode. The clock source selected by e_clock_source is enabled only when enable = 1.

Return Values

LPC_SUCCESS:

Properties

Prototyped in file “r_lpc_rx_if.h”

Description

This function will configure the return clock switching parameter that allows the clock source to be switched on returning from Sleep Mode to the HOCO, LOCO or Main Clock. The following items have to be followed to allow for Return Clock Switching:

1. RX110, RX111, RX113 MCUs :
   - When entering Sleep Mode, the system clock should be the Sub-Clock oscillator. On exiting sleep, the operating mode will return to whatever the operating power control mode was before entering sleep.
   - If the Main OSC is chosen as the Sleep Return clock source, middle speed mode is the return mode. Make sure the internal clock after returning from sleep mode does not exceed the limits of middle speed mode.
2. RX130, RX231 MCUs :
   - When entering Sleep Mode, the system clock should be the Sub-Clock oscillator. On exiting sleep, the operating mode will return to whatever the operating power control mode was before entering sleep.
   - If Middle Speed mode is the return mode and the Main OSC is chosen as the Sleep Return clock source, make sure the internal clock after returning from sleep mode does not exceed the limits of middle speed mode.
3. RX64M, RX71M MCUs:
   - When entering sleep mode, select LOCO or the sub-clock as the clock source.

Reentrant

No

Example

```
lpc_err_t err;

err = R_LPC_ReturnClockSwitch(LPC_MAIN_OSC, true);
```

Special Notes:

None.

R_LPC_GetVersion ()

This function returns the driver version number at runtime.

Format
```
void   R_LPC_GetVersion(void);
```

Parameters
none

Return Values
Version number.

Properties
Prototyped in file “r_lpc_rx_if.h”

Description
Returns the version of this module. The version number is encoded such that the top 2 bytes are the major version number and the bottom 2 bytes are the minor version number.

Reentrant
Yes.

Example
```
uint32_t   version;

version = R_LPC_GetVersion();
```

Special Notes:
None

4. Usage Examples

4.1 Example sequence for entering higher power operating modes, RX1xx MCUs

The RX110, RX111, RX113, RX130 MCUs have internal regulators that control power to the chip. Configuring the regulators to supply higher power before moving into a higher power state is necessary for proper operation.

Below is an example that shows the sequence of operations and API calls necessary to move from a lower power state to a higher powered one. It is assumed that at the start of this sequence, the system clock source is the sub-clock and the operating power control mode is Low Speed mode.

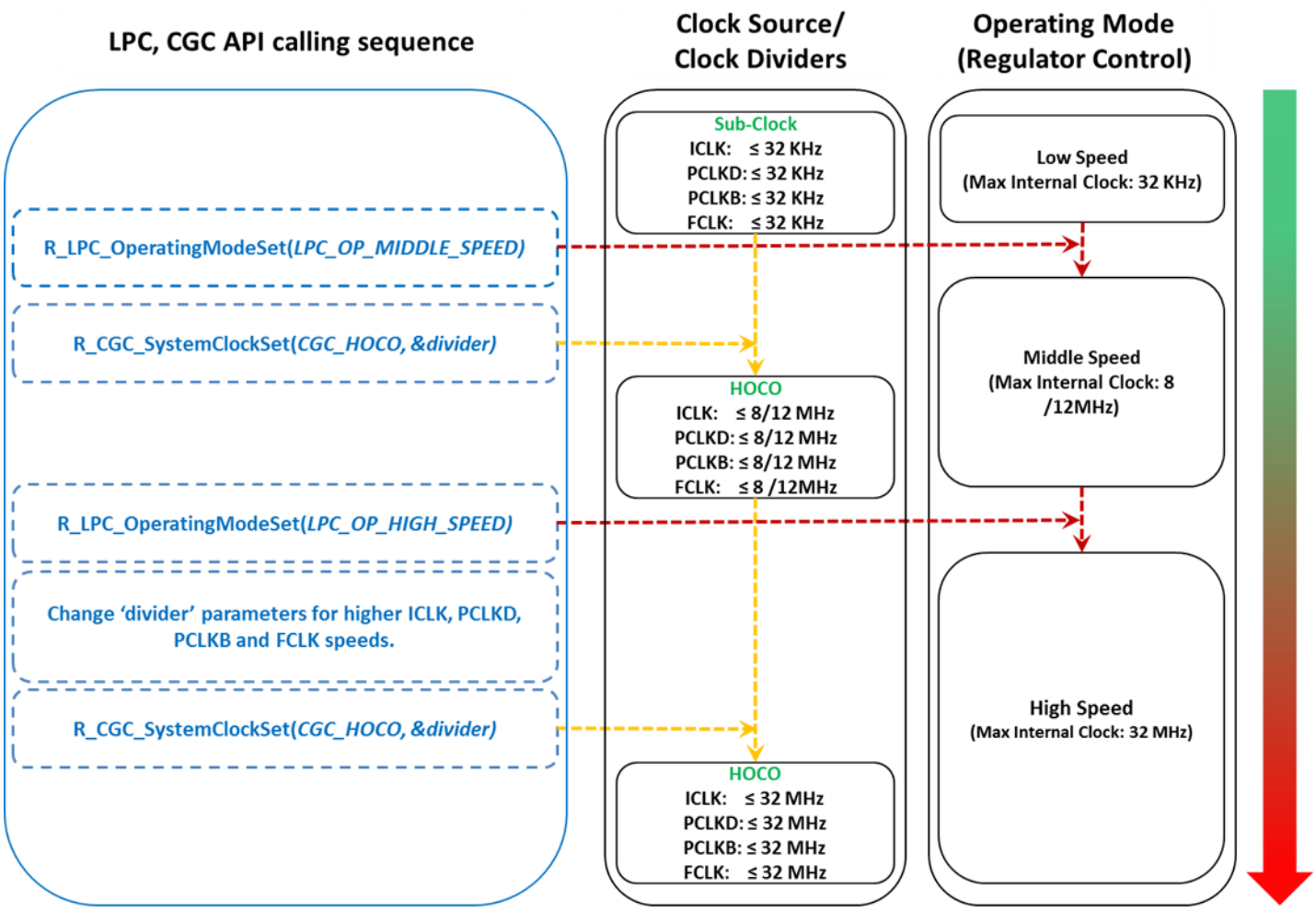

Figure 4.1: Sequence for moving from a low power to high power state (RX110, RX111, RX113, RX130)

4.2 Example sequence for entering lower power operating modes, RX1xx MCUs

When moving to lower power states, it is important to first move to the lower power state before switching the regulator down for lower supply voltage.

Below is an example that shows the sequence of operations and API calls necessary to move from a higher power state to a lower powered one. It is assumed that at the start of this sequence, the system clock source is the HOCO and the operating power control mode is High Speed mode.

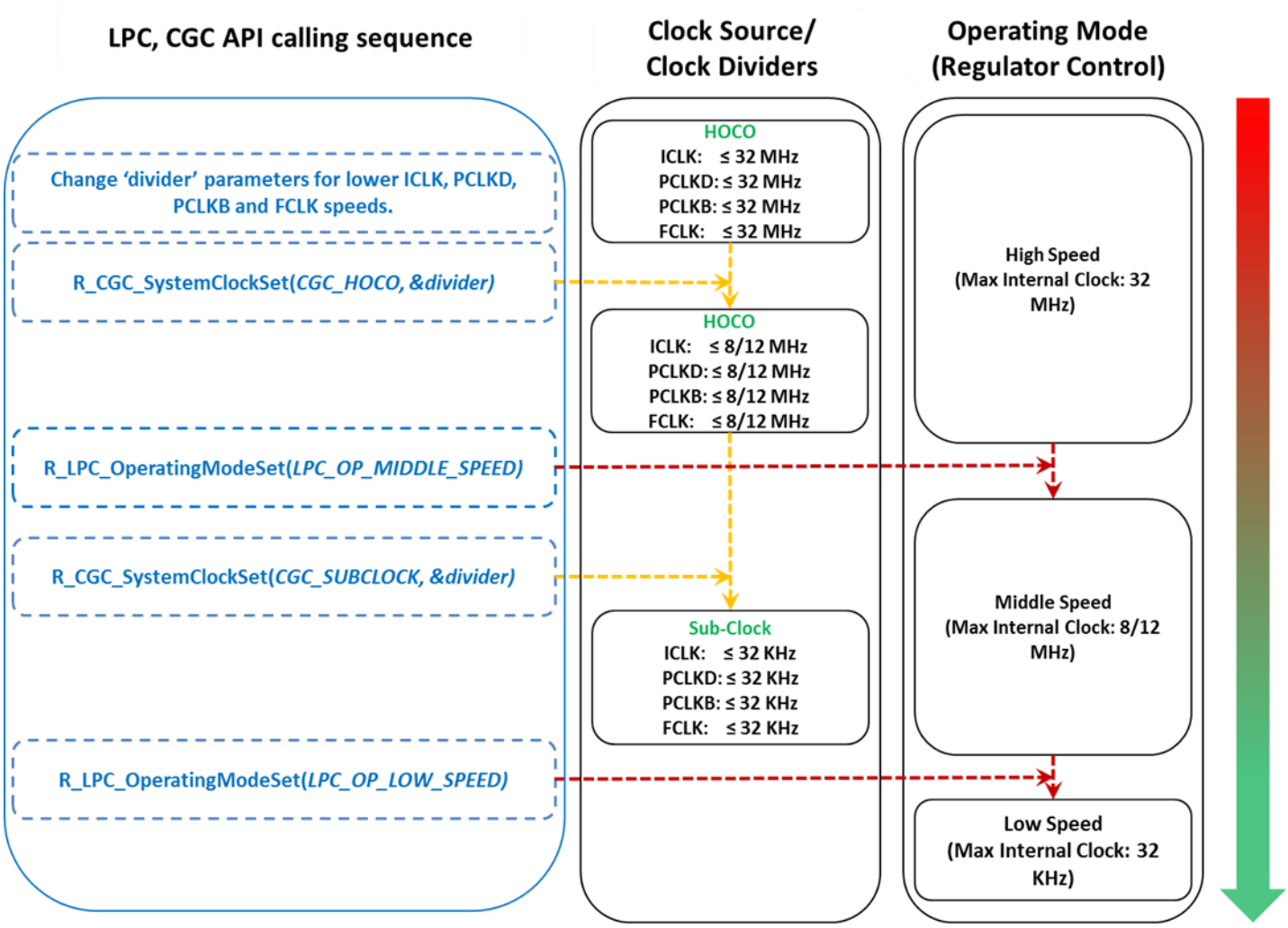

Figure 4.2: Sequence for moving from a high power to low power state (RX110, RX111, RX113, RX130)

4.3 Example sequence for entering higher power operating modes, RX210

The RX210 has internal regulators that control power to the chip. Configuring the regulators to supply higher power before moving into a higher power state is necessary for proper operation.

Below is an example that shows the sequence of operations and API calls necessary to move from a lower power state to a higher powered one. It is assumed that at the start of this sequence, the system clock source is the sub-clock and the operating power control mode is Low Speed mode.

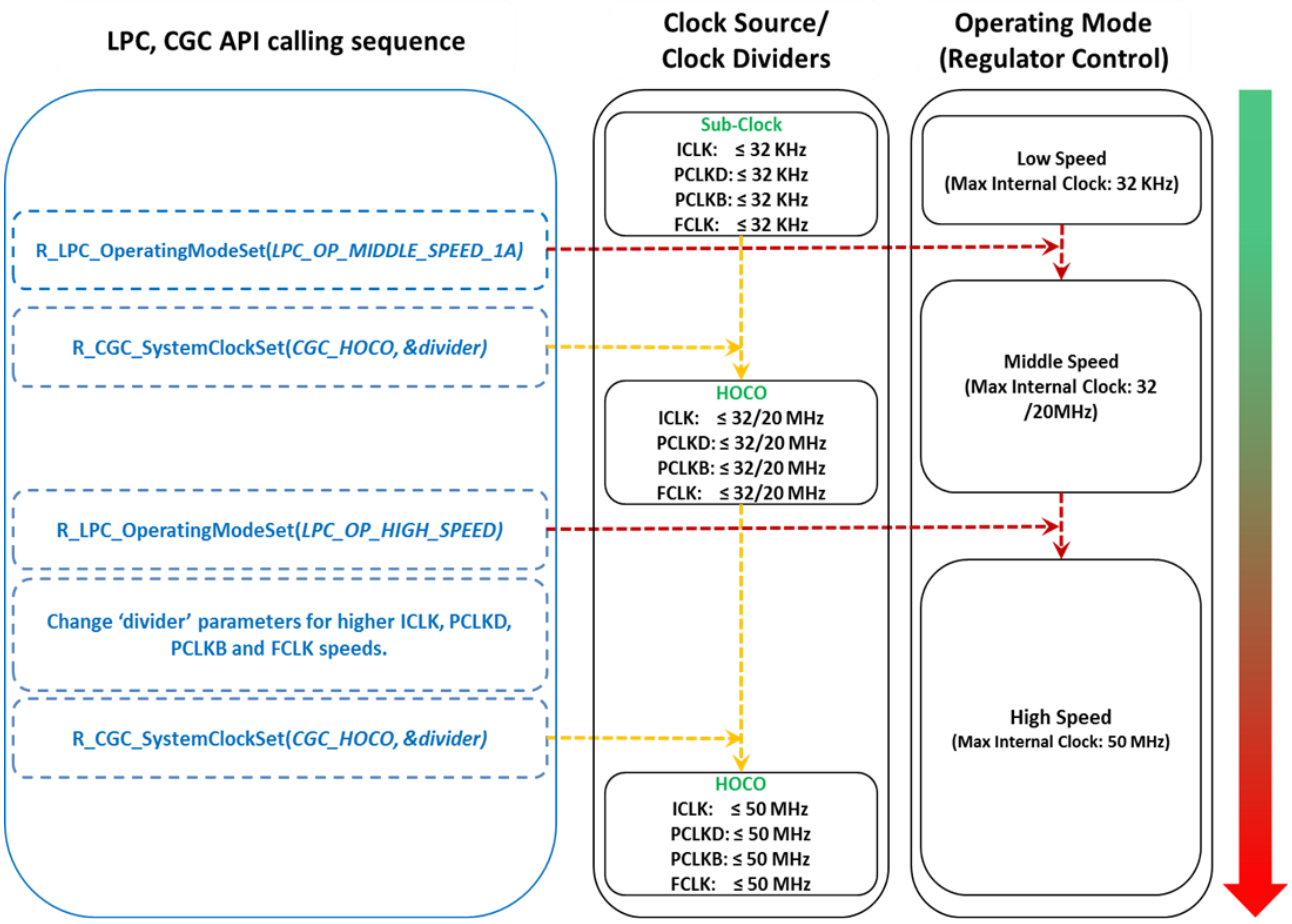

Figure 4.3: Sequence for moving from a low power to high power state (RX210)

In addition to this, there is also a Return Clock Switching function that automatically switches the main clock source on waking up out of Sleep.

4.4 Example sequence for entering lower power operating modes in the RX210

When moving to lower power states, it is important to first move to the lower power state before switching the regulator down for lower supply voltage.

Below is an example that shows the sequence of operations and API calls necessary to move from a higher power state to a lower powered one. It is assumed that at the start of this sequence, the system clock source is the HOCO and the operating power control mode is High Speed mode.

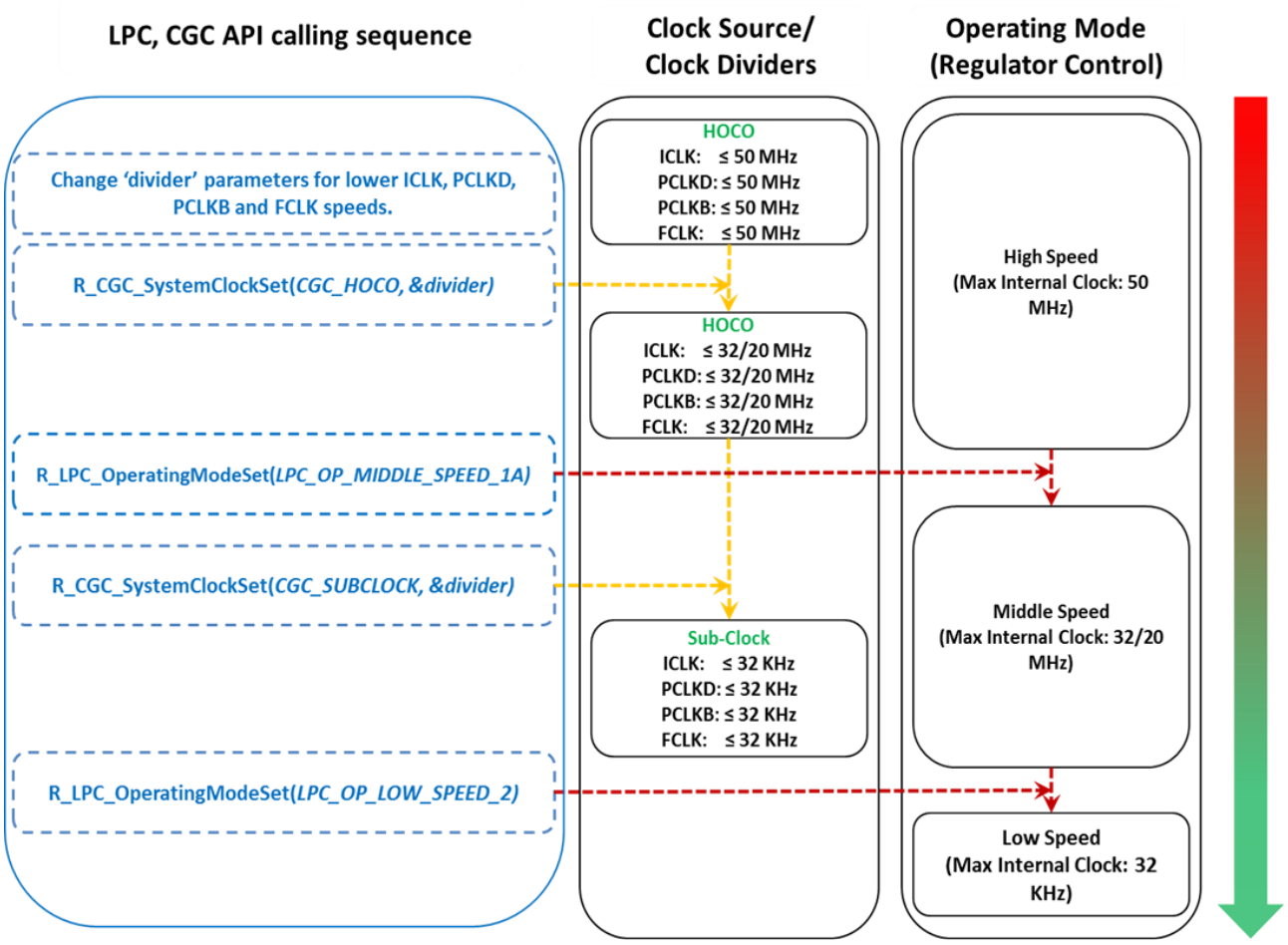

Figure 4.4: Sequence for moving from a high power to low power state (RX210)

5. Demo Projects

Demo projects are complete stand-alone programs. They include function main() that utilizes the module and its dependent modules (e.g. r_bsp). This FIT module has the following demo projects:

5.1 lpc_demo_rskrx113

A simple demo of the RX113 Low Power Consumption (LPC) for the RSKRX113 starter kit (FIT module "r_lpc_rx"). The demo loops toggling LED 0 every second. When switch 2 (SW2) is first pressed, STANDBY mode is entered and LED 0 stops toggling as the CPU does not run in standby mode. When switch 2 is pressed again, standby mode is exited and the LED begins flashing again.

1. Compile and download the sample code.
2. Execute the demo program.

Note: By removing R221 (0 ohm resistor on the bottom side of the board) and fitting an ammeter across J11, it is possible to observe the current consumption as the MCU switches between normal and standby mode. For accurate measurement, once the board has been programmed using the debugger, the debugger should be disconnected and an external power supply should be used to power the board during the test. When measurement is complete, place a jumper on J11 for normal operation.

Boards Supported

RSKRX113

5.2 lpc_demo_rskrx231

This is the demo program for the RX231 Low Power Consumption (LPC) for the RSKRX231 starter kit. The demo steps through several power modes in the following order:

(1) High-speed operating mode
(2) Middle-speed operating mode
(3) Low-speed operating mode
(4) High-speed operating mode (2nd time)
(5) Sleep mode with return clock source switching enabled
(6) Software standby mode
(7) Deep sleep mode

Once the demo has been compiled, downloaded and is executing, press SW2 to transition from one power mode to the next. As the Operating Power Control modes are entered LED 0 will flash according to the clock speed for that mode. LED 0 will stop flashing in the Low Power Consumption modes (Sleep, Software Standby, ...) as the CPU does not run in those modes. Upon entering each power mode, a message will be sent to the console window in e^2 studio indicating the state of the MCU. When the demo has completed, the MCU will remain in Deep Software Standby mode and the program will need to be reset to step through the power modes again.

Note: By placing an ammeter across J8, it is possible to observe the current consumption as the MCU switches between Operating Power modes and Power Consumption Modes. It will be necessary to remove R55 (0 ohm resistor next to J8) to perform that test. For accurate current measurement, once the board has been programmed using the debugger, the debugger should be disconnected and an external supply used to power the board during the test. After testing replace R55 or install the jumper in J8 for normal operation.

5.3 lpc_demo_rskrx71M

This is the demo program for the RX71M Low Power Consumption (LPC) module for the RSKRX71M starter kit. The demo steps through several power modes in the following order:

(1) High-speed operating mode
(2) Low-speed operating mode 1
(3) Low-speed operating mode 2
(4) Sleep mode with return clock source switching enabled
(5) All-module clock stop mode
(6) Software standby mode
(7) Deep software standby mode

Once the demo has been compiled, downloaded and is executing, press SW2 to transition from one power mode to the next. As the Operating Power Control modes are entered LED 0 will flash according to the clock speed for that mode. LED 0 will stop flashing in the Low Power Consumption modes (Sleep, Software Standby, ...) as the CPU does not run in those modes. Upon entering each power mode, a message will be sent to the console window in e^2 studio indicating the state of the MCU. When the demo has completed, the MCU will remain in Deep Software Standby mode and the program will need to be reset to step through the power modes again.

Note: By placing an ammeter across J21, it is possible to observe the current consumption as the MCU switches between Operating Power modes and Power Consumption Modes. Insure that R217 (0 ohm resistor next to J21) has not been installed. For accurate current measurement, once the board has been programmed using the debugger, the debugger should be disconnected and an external supply used to power the board during the test. Replace the jumper on J21 for normal operation when finished with the demo.

5.4 Adding a Demo to a Workspace

Demo projects are found in the FITDemos subdirectory of the distribution file for this application note. To add a demo project to a workspace, select File>Import>General>Existing Projects into Workspace, then click “Next”. From the Import Projects dialog, choose the “Select archive file” radio button. “Browse” to the FITDemos subdirectory, select the desired demo zip file, then click “Finish”.

5.5 Downloading Demo Projects

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module needs to be downloaded. To download the FIT module, right click on this application note and select “Sample Code (download)” from the context menu in the *Smart Brower >> Application Notes* tab.

6. Appendices

6.1 Confirmed Operation Environment

This section describes confirmed operation environment for the LPC FIT module.

Table 6.1 Confirmed Operation Environment (Rev. 1.41)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00. Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.41

6.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file “platform.h”.

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)”

- Using e^2 studio:

  Application note “Adding Firmware Integration Technology Modules to Projects (R01AN1723)”

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note “Board Support Package Module Using Firmware Integration Technology (R01AN1685)”.

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_lpc_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

Revision History

Rev.	Date	Description	
		Page	Summary
1.40	Oct. 01, 2016	-	Initial release
1.41	Apr. 01, 2019	-	Changes associated with functions: Added support setting function of configuration option Using GUI on Smart Configurator. [Description] Added a setting file to support configuration option setting function by GUI.
		1	Changed Related Document.
		4	Moved 1.1 LPC FIT Module. Changed 1.2 Overview of the LPC FIT Module.
		5	Moved 1.3 API Overview. Changed 1.4 State Transition Diagram.
		8	Deleted Hardware Resource Requirement. Deleted Limitations. Changed 2.3 Supported Toolchains. Added 2.4 Interrupt Vector. Changed 2.5 Header Files. Changed 2.6 Integer Types.
		9	Changed 2.7 Configuration Overview.
		10	Changed 2.8 Code Size.
		12	Changed 2.9 Parameters.
		13	Changed 2.10 Return Values. Added 2.11 Callback Function.
		14	Changed 2.12 Adding the FIT Module to Your Project.
		15	Added 2.13 “for”, “while” and “do while” statements.
		25	Changed R_LPC_GetVersion.
		31	Added 5.5 Downloading Demo Projects.
		32	Added 6.1 Confirmed Operation Environment.
		33	Added 6.2 Troubleshooting.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.