

# FreescalE MQX RTOS Example Guide

## Low power example

This document explains the Low power example, what to expect from the example and a brief introduction to the API used.

## The example

The example shows switching between several predefined low power operation-modes. The core and serial driver are being notified and their settings are updated based on new operation mode transparently for the application. The switch is initiated by the on-board button press. In some low power operation modes, the application should be woken up by a hardware event like RTC Alarm interrupt or a Serial line RX interrupt. The immediate settings are being displayed on terminal and they are signaled by different LED patterns.

## Running the example

The Low power application belongs to the set of examples of MQX low power support. The `MQX_ENABLE_LOW_POWER` macro must be set to non-zero in the `user_config.h` file prior to compilation of MQX kernel libraries and the example itself.

To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.

## Explaining the example

The application example creates two tasks:

- **main:** task sets up the BUTTON and LED1 pins and installs the RTC alarm interrupt. Any time the button is pressed this task toggles the LED1 and it changes the operation mode. In some modes, the RTC alarm interrupt or SCI RX interrupt needs to occur to wake up the core. The flow of main task is shown on the diagram below.
- **for\_loop\_led\_task:** sets up the LED2 and toggles its state with a frequency given by current clock configuration. You will see a change of the clock configuration as a change of LED2 blink rate. When the LED2 is not blinking the core is in the low power stop mode and is not executing any task.

The LWGPIO driver is used for the LED and BUTTON pins handling. The `button_led_init()` function initializes the corresponding pins for input and output and also it installs the falling-edge interrupt for the button (with internal pull-up resistor enabled). The ISR handler toggles the LED2 and sets the event to trigger an action in the main task.

The operation modes are set using the Low-power Manager API function `_lpm_set_operation_mode()`. The parameter of this function is one of the LPM constants that correspond to predefined operation modes.

Note: even after a core wakeup event, the current operation mode stays unchanged until the next call to the `_lpm_set_operation_mode` function.

The application uses the SCI and RTC interrupts as events to wake-up the core from some low power operation modes. For the serial line, the interrupt is available only if an interrupt-driven driver is used "ittyX:". The RTC alarm is always set to time "now + 10 seconds". The RTC ISR itself wakes up the core and the `_lpm_wakeup_core()` function is used here to let the core to run also after the ISR finishes. Refer to Kinetis documentation for more information about specific CPU behavior in various low power modes.

Also note that the operation mode `LPM_OPERATION_MODE_WAIT` requires the clock to be switched to 2MHz first. This is done using the LPM API function `_lpm_set_clock_configuration()`. The original clock configuration is restored after the operation mode is changed to default again to avoid unsupported core configuration.

