Freescale MQX RTOS Example Guide

Frequency change example

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

The example

The example shows switching between several predefined clock configurations. The core and serial driver are being notified and their settings are updated based on new clock configuration transparently for the application. The switch is initiated by the on-board button press. New clock settings are displayed on terminal and the change is also visible on the rate the LED toggles.

Running the example

The Frequency Change 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 three tasks:

- **main:** task sets up the BUTTON and waits for the BUTTON press event in an endless loop. Any time the button is pressed this task changes the clock configuration and prints out new clock settings. The flow of main task is shown on the diagram below.

- **led_task:** sets up the LED1 and toggles its state once per second regardless of the current clock configuration. This is a demonstration of clock-independent task.

- **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.

The LWGPIO driver is used for the LED and BUTTON pins handling. The button_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 sets the event to trigger an action in the main task.

The clock settings are changed using the Low-power Manager API function _lpm_set_clock_configuration(). The parameter of this function is one of the BSP constants that correspond to predefined clock configurations.

The actual frequencies are obtained using the Clock Manager API function _cm_get_clock(). The parameters are (the current) clock

configuration and an identifier of the clock source whose clock frequency is to be obtained.