Freescale MQX RTOS Example Guide

ISR example

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

The example

Currently for Kinetis MCU family and Vybrid MCU with Cortex M4 the example illustrates the usage of interrupt which is installed directly into the interrupt vector table of the processor as well as the usage of standard method in MQX to handle interrupt. Installing the interrupt handler into the interrupt vector table of the processor helps reducing the latency compared to servicing the interrupt in the standard way in which the function *_int_kernel_isr()* is invoked before fetching the interrupt handler. In this example the interrupt source associated with push buttons available on demo board is examined. Interrupt occurs when user pushes button. The default interrupt handling method in MQX is also examined and is demonstrated via the system timer interrupt. For other platform MCUs (including Power PC and ColdFire based MCUs, Vybrid MCU with Cortex A5), the example simply shows the interrupt associated with the system timer.

Running the example

For Kinetis MCU family and Vybrid MCU with Cortex M4 the ISR example requires the MQX_ROM_VECTORS macro to be set to zero in the user_config.h file prior to compilation of MQX kernel libraries and the example itself. Apart from those MCUs the default macro settings are sufficient to run this example.
To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.

Explaining the example

For Kinetis MCU family and Vybrid MCU with Cortex M4:
This example uses the lwgpio driver. The example creates two tasks named main_task and blink_led_task. The task main_task does the following job.

- Install a new interrupt handler for the system tick timer interrupt using function *_int_install_isr().* This puts the interrupt handler into the interrupt vector table of MQX kernel. The new interrupt handler posts the light weight semaphore and invokes the old interrupt handler of the system timer interrupt.
- Wait for the light weight semaphore to be available and increment the tick count which is the number of times the system timer interrupt happens since the new interrupt handler is installed into the kernel interrupt vector table (after calling function *_int_install_isr()*).

The task blink_led_task is responsible for

- Initialize a led, a button on demo board by calling functions *init_led()*, *init_interrupt_btn().* In function *init_interrupt_btn()* the button is configured as interrupt source with falling edge triggering

using functions from lwgpio driver: *lwgpio_init()*, *lwgpio_set_functionality()*, *lwgpio_set_attribute()*, *lwgpio_int_init()*. For a button the interrupt handler is installed directly into the interrupt vector table of the processor by a call to function *_int_install_kernel_isr()*. Finally the button is enabled as the interrupt source using functions *_bsp_int_init()* and *lwgpio_int_enable()*. The interrupt handler associated with the button updates the value of a global variable to notify the user task of button being pressed.

- Enter an endless loop to wait for user to press SW1 button and display the current value of tick count. The LED is blinking when user presses button. If the button is pressed again the LED is turned off.

The following picture shows output of the example.

```
====================== ISR Example =======================
Press the SW1 to blink LED1, press it again to turn it off

Led starts blinking at tick No. = 112

Led is off at tick No. = 820

Led starts blinking at tick No. = 1809

Led is off at tick No. = 3425

Led starts blinking at tick No. = 4071
```

For other platform MCUs (including Power PC and ColdFire based MCUs, Vybrid MCU with Cortex A5):

The application example creates a task named main_task. The task main_task does following jobs.

- install new interrupt handler for system timer.
- enter the time delay of 200 ticks.
- print out number of times the system timer interrupt happened.

The application example also defines new interrupt handler for the system tick timer called new_tick_isr() which increments a count of interrupt happened and then invokes the original interrupt handler for the system tick timer.

The following output is expected on the terminal.

Tick count = 200