Freescale MQX RTOS Example Guide

watchdog example

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

The example

The watchdog example code is used to demonstrate how to use a software watchdog for a task. It creates a soft watchdog for the task which restarts it in a loop. The loop time is increased each time until the watchdog timer expires.

Running the example

Start HyperTerminal on the PC (Start menu->Programs->Accessories->Communications).
Make a connection to the serial port that is connected to the board (usually will be COM1).

Set it for 115200 baud, no parity, 8 bits and click OK.

Then we compile the project watchdog. When compiling there may be compiling error like this:

Error : preprocessor #error directive
watchdog.h line 43 #error WATCHDOG component is currently disabled in MQX kernel.

To remove this error please set MQX_USE_SW_WATCHDOGS to 1 in user_config.h and recompile kernel.

In directory C:\Program Files\Freescale\Freescale MQX 3.x \config\common\small_ram_config.h please change

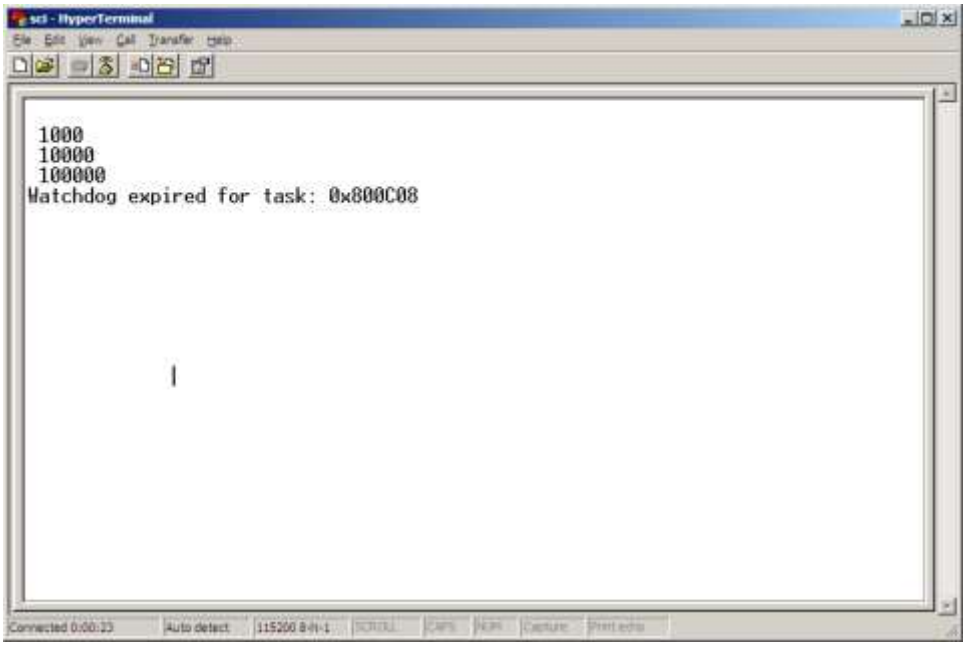
```
#define MQX_USE_SW_WATCHDOGS                0
```

to

```
#define MQX_USE_SW_WATCHDOGS                1
```

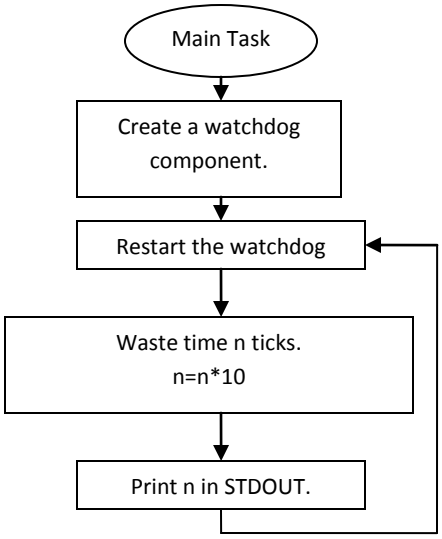
And rebuild bsp and psp libraries. These libraries can be found in C:\Program Files\Freescale\Freescale MQX 3.7\mqx\build\cw10
This may take several minutes depending on the speed of the PC.

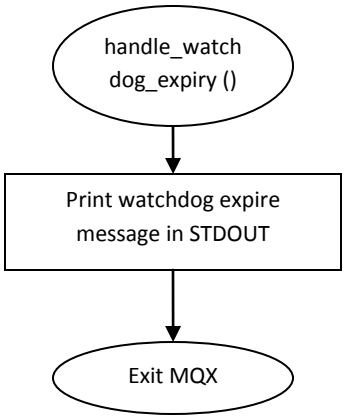
Then the watchdog example can be compiled and downloaded to target board. We can see the running results displayed in the HyperTerminal:



Explanation of the example

The application demo creates only one main task. The flow of the task is described in the next figure.





The main task first initializes a MQX_TICK_STRUCT using the following line:

```
_time_init_ticks(&ticks, 10);
```

The value 10 is used to initialize struct ticks, and the struct ticks will be used to restart the watchdog later in the main loop.

Then the main task creates a watchdog component using the following line:

```
result = _watchdog_create_component(BSP_TIMER_INTERRUPT_VECTOR,
      handle_watchdog_expiry);
```

By this a software watchdog is created for the task and the handler for watchdog expiry is assigned. When the watchdog is not restarted in time, the watchdog will expire and handle_watchdog_expiry() will be called automatically.

Then main task enters a loop, in the loop it first restarts the watchdog with the parameter ticks which was initialized in previous steps. And then it wastes some time - the time will be increased 10 times every loop. Then it stops the watchdog and prints the elapsed time and continues with the next loop.

When the value exceeds the watchdog expiration period, handle_watchdog_expiry will be called, and watchdog expiry message along with the task number will be printed to STDOUT.