

Freescale MQX RTOS Example Guide

GPIO example

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

The example

The GPIO example performs various open - close operations on several sets of pin tables, which explain and demonstrate the use of "pin tables" as the output of GPIO fopen command. At the end of the main_task an infinite loop runs and reads the status of the pin connected to Switch 1. This same value is written to the pin connected to the LED1. If board has no SWITCH button, blink the LED1 instead.

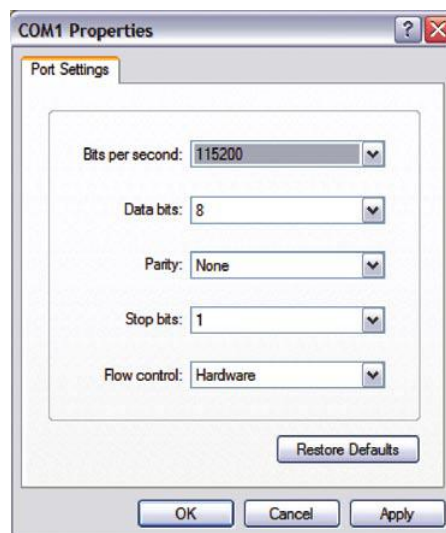
Running the example

Run HyperTerminal on the PC (Start menu->Programs->Accessories->Communications).

Make a connection to the serial port which is connected to the board (usually will be COM1).



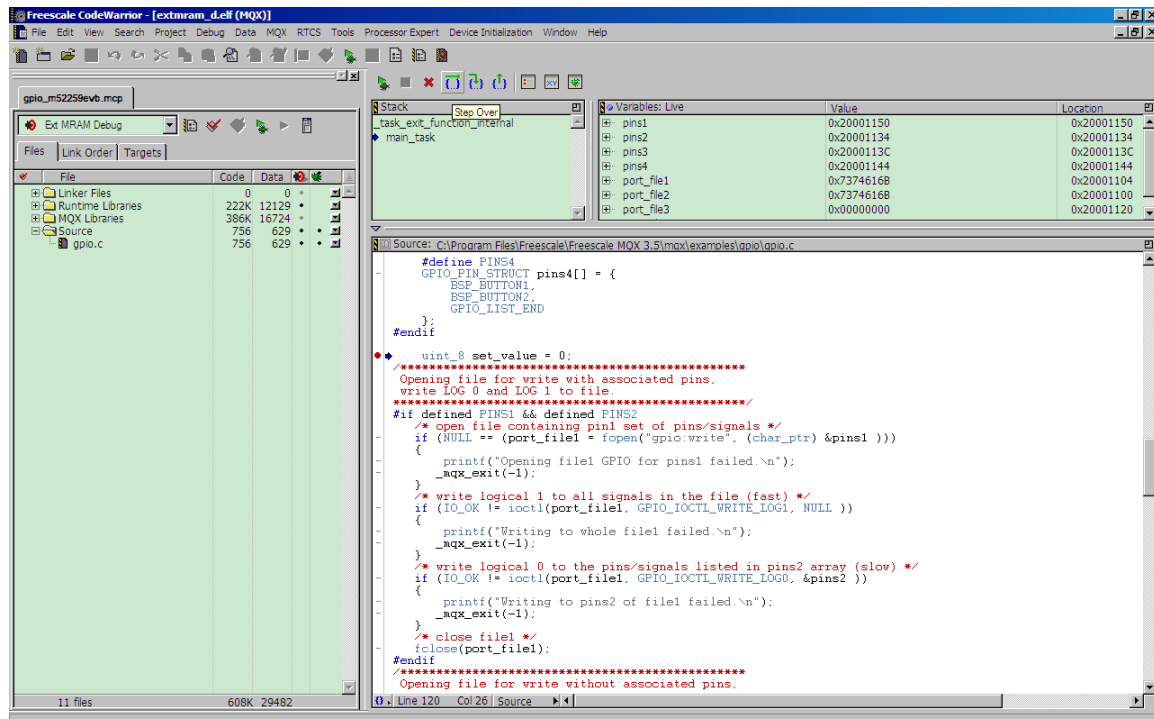
Set it with 115200 baud, no parity, 8 bits and click OK.




The GPIO example is focused mainly on debugging and realizing operations on a set of pins. It is recommended to run GPIO example step by step in debugging mode.

Run CodeWarrior for ColdFire software, open GPIO project. Set breakpoint at code line 120 [uint_8 set_value = 0] of [gpio.c] file.


Select release target, then press debug button.



Note: If select [Flash] target, you need to program Flash memory with related image file (*.s19,*.bin) first.

Press  (Step over) button or press F10 shortcut key to run examples in CodeWarrior Debug mode. The LEDs on board could be turned on /off. When the serial terminal shows the next info:
"Use SWITCH on board to switch LED on or off"

Press SWITCH1 button to control one LED on/off.

Note: If there is no SWITCH1 button on board, press  (Run) button to run code and one LED blinked instead.

Explaining the example

The application example creates only one main task. The flow of the task is described in next figure. It includes three demos on GPIO application.

To access GPIO pins, it is needed to open the GPIO device with a parameter specifying set of pins to be used. So, the main task starts with four pin tables' definition. The pin_table is an array of GPIO_PIN_STRUCT ended with GPIO_LIST_END. Each pin table includes related controlled pin (LED or SWITCH), and default status.

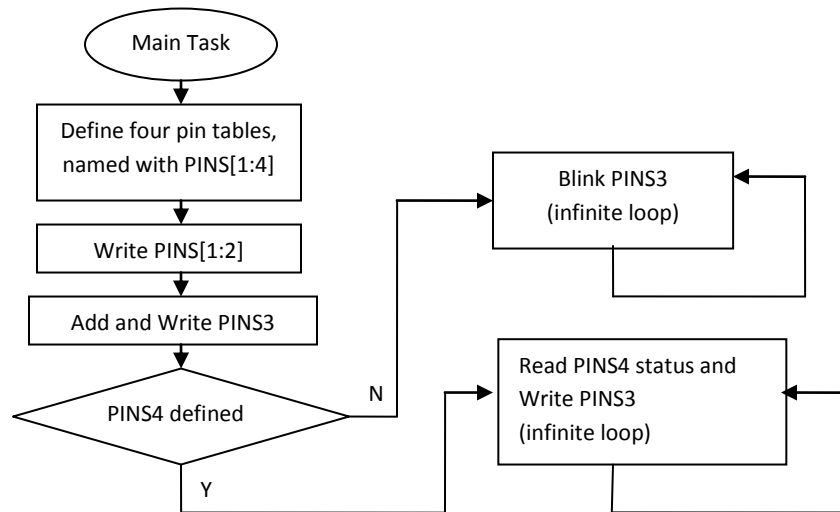
Demo 1 is writing logic 1 or logic 0 to PIN[1:2] (LED) pins. The following line using fopen() calls GPIO write application and write PINS1 related pins with defined status value.

```
port_file1 = fopen("gpio:write", (char_ptr) &pins1 )
```

The following line using ioctl() writes a logic 1 to PINS1 related (LED) pins:

```
if (IO_OK != ioctl(port_file1, GPIO_IOCTL_WRITE_LOG1, NULL ))
```

Same using ioctl() to write logic 0 to PINS2 related (LED) pins.



Demo 2 is adding PINS3 pin to a null file and write logic 0 to PINS3 related (LED) pin. The following line using fopen() opens a file not containing any pin/signal:

```
port_file2 = fopen("gpio:write", (char_ptr) NULL)
```

Using ioctl() to add PINS3 related pin to opened file and also using ioctl() to write logic 0 to PINS3 related (LED) pin.

Demo 3 is detecting PINS4 input status, then control PINS2 (LED) pin output. Using the following line to open PINS4 for input:

```
port_file3 = fopen("gpio:read", (char_ptr) &pins4 )
```

By reading PIN4 (SWITCH) button status, in infinite loop to write related logic value to PIN2 (LED) pin.

If the board doesn't have buttons, the demo opens PINS3 file and blinks the LEDs inside the infinite loop using time delay to set 1 second time interval.

```
_time_delay(1000);
```