Freescale MQX RTOS Example Guide

SPI example

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

The example

The example shows the usage of the SPI driver to perform the basic operation including reading data from, writing data to and erasing the memory of a SPI serial flash memory chip.

Running the example

What SPI port is used depends on the connection between the MCU board and the Memory board. Settings are in the file spi_settings.h which contains macros:

SPI_INSTANCE 0 – index of SPI peripheral, used 0-SPI0
SPI_TRANSFER_TIMEOUT 1000 – timeout in microseconds used in calling SPI blocking transfer function
SPI_SS_PORT PORTD_BASE – port used for slave select
SPI_SS_GPIO GPIOD_BASE - gpio used for slave select
SPI_SS_PIN 0 – pin used for slave select

To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.
The MCU board and the Memory board are connected via the 4-wire SPI connection. The SPI_SCK, SPI_MOSI, SPI_MISO, SPI_SS pins are used.

Explaining the example

The application example creates only one task called main_task which is responsible for configuring the SPI protocol, erasing memory in memory board and reading data from and writing data to the memory board. The MCU is the master and the SPI serial flash memory is the slave in the SPI data transfer.

- For using SPI is needed initialize and configure SPI bus. For configure is used structure which contains polarity and phase of clock signal, first sending bit (MSB or LSB), baud rate and number bits per frame. In this example is a used setting: clock phase – first edge, clock polarity – active high, first sending bit is MSB, baud rate 500000 bits per sec and 8 bits per frame.

- The slave select pin is configured as GPIO.

- SPI driver works in interrupt mode, the interrupt must be installed in MQX and file fsl_dspi_irq.c must not be included in project. To install service routine in MQX must be called function _int_install_isr() with parameters: IRQ number of interrupt, pointer to SPI handler function and index of SPI peripheral.

- The SPI serial flash memory is prepared to run the reading and writing test.
  - memory_read_status() read the status of the serial flash memory
  - memory_set_protection() remove the write protection and memory_chip_erase() erase the serial flash memory.

```
Read memory status register ... 0x10
Write unprotect memory
Read memory status register ... 0x10
Erase whole memory chip ... Erased
```

- Using functions memory_write_byte(), memory_read_byte(), memory_read_status() functions defined in the file spi_memory.c we can check the operation of serial flash with single data byte read and write process.

```
Write byte at address 0x0000f0:
0xba
Read byte from address 0x0000f0:
0xba
Compare write/read bytes ... OK
```

- Using function memory_write_data(), memory_read_data(), memory_read_status() functions defined in the file spi_memory.c we can check the operation of serial flash memory with read and write process for a sequence of data bytes.

```
Write data at address 0x0001f0:
Hello,World!
Read data from address 0x0001f0:
Hello,World!
Compare write/read data ... OK

Write data at address 0x0002f0:
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyz1234567890
Read data from address 0x0002f0:
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyz1234567890
Compare write/read data ... OK
```

- Next is test the property of the SPI protocol in which the master and slave node exchange data simultaneously. Here the master writes the read command (0x03) and an address (0x0000f0) to the serial flash memory in addition to 6 dummy data bytes (0x00) to get the data in that address. The exchanged data in its SPI receive buffer is then displayed on the terminal.

```
Simultaneous write and read - memory read from 0x000000f0 (10):
Write: 0x03 0x00 0x00 0xf0 0x00 0x00 0x00 0x00 0x00 0x00
Read: 0xff 0xff 0xff 0xff 0xba 0xff 0xff 0xff 0xff 0xff
Simultaneous read/write (data == 0xba) ... OK
```

- The main_task exits and example is finished.