Lab 1 – introduction to FreeRTOS

Preparation

Create a led blinking C++ FreeRTOS project as described in How_to_create_freeRTOS_C++-project.pdf.

Note that hardware setup function is called in main() of blinky.cpp. Hardware setup calls board library initialization functions which may override hardware settings of objects that are initialized before main() is called (namely global objects) → create your objects after hardware setup either in main() or in the tasks.

Exercise 1

Change vUARTTask() so that it prints minutes and seconds since start instead of “Tick: …”.

The task should print something like this (lines are printed at one second intervals):

```
Time: 00:01
Time: 00:02
Time: 00:03
...
Time: 00:59
Time: 01:00
...
```

DEBUGOUT() function works the same way as printf() from C standard library. See for example http://www.cplusplus.com/reference/cstdio/printf/ for more information.

Exercise 2

Change the blinking patterns in the following way:

- Red led blinks SOS morse code.
- Green led toggles on and off so that the led is off on every other SOS.

Exercise 3

Change the program so that when SW1 (wake button) on PIO0_17 is pressed then tick numbers increment ten times per second and once per second when not pressed.

A little recap from a previous course (see next page):

```
/*
 * DigitalIoPin.h
*  Created on: 31.1.2016
 *      Author: krl
 */

#ifndef DIGITALIOPIN_H_
#define DIGITALIOPIN_H_

class DigitalIoPin {
public:
	enum pinMode {
		output,
		input,
		pullup,
		pulldown
	};
	DigitalIoPin(int port, int pin, pinMode mode, bool invert = false);
	virtual ~DigitalIoPin();
	virtual bool read();
	void write(bool value);
private:
	int port;
	int pin;

};

#endif /* DIGITALIOPIN_H_ */


/*
 * DigitalIoPin.cpp
*  Created on: 31.1.2016
 *      Author: krl
 */

#include "DigitalIoPin.h"
#include "chip.h"


DigitalIoPin::DigitalIoPin(int port_, int pin_, pinMode mode, bool invert) : port(port_), pin(pin_) {
	if(mode == output) {
		Chip_IOCON_PinMuxSet(LPC_IOCON, port, pin, IOCON_MODE_INACT | IOCON_DIGMODE_EN);
		Chip_GPIO_SetPinDIROutput(LPC_GPIO, port, pin);
	}
	else {
		uint32_t pm = IOCON_DIGMODE_EN;

		if(invert) pm |= IOCON_INV_EN;

		if(mode == pullup) {
		  pm |= IOCON_MODE_PULLUP;
		}
		else if(mode == pulldown) {
		  pm |= IOCON_MODE_PULLDOWN;
		}

		Chip_IOCON_PinMuxSet(LPC_IOCON, port, pin, pm);
		Chip_GPIO_SetPinDIRInput(LPC_GPIO, port, pin);
	}
}

DigitalIoPin::~DigitalIoPin() {
	// TODO Auto-generated destructor stub
}

bool DigitalIoPin::read() {
	return Chip_GPIO_GetPinState(LPC_GPIO, port, pin);
}

void DigitalIoPin::write(bool value) {
	return Chip_GPIO_SetPinState(LPC_GPIO, port, pin, value);
}
```