Lab 3: Interrupts

Nick Scheele

Part 0

Main function

```
// Read Table 95 in the LPC user manual and setup an interrupt on a switch
connected to Port0 or Port2
// a) For example, choose SW2 (P0_30) pin on SJ2 board and configure as input
//.   Warning: P0.30, and P0.31 require pull-down resistors
// b) Configure the registers to trigger Port0 interrupt (such as falling edge)
gpio_s sw2 = gpio__construct_as_input(0, 30);
LPC_GPIOINT->IO0IntEnF |= (1 << 30); // enable interrupt

// Install GPIO interrupt function at the CPU interrupt (exception) vector
// c) Hijack the interrupt vector at interrupt_vector_table.c and have it call our
gpio_interrupt()
//    Hint: You can declare 'void gpio_interrupt(void)' at interrupt_vector_table.c
such that it can see this function

lpc_peripheral__enable_interrupt(LPC_PERIPHERAL__GPIO, gpio_interrupt, "sw
interrupt");
// Most important step: Enable the GPIO interrupt exception using the ARM Cortex M
API (this is from lpc40xx.h)
NVIC_EnableIRQ(GPIO_IRQn);

// Toggle an LED in a loop to ensure/test that the interrupt is entering ane
exiting
// For example, if the GPIO interrupt gets stuck, this LED will stop blinking
gpio_s Led0 = gpio__construct_as_output(1, 18);
while (1) {
 delay__ms(100);
 gpio__toggle(Led0);
}
```

Helper function

```
void gpio_interrupt(void) {
 // a) Clear Port0/2 interrupt using CLR0 or CLR2 registers
 LPC_GPIOINT->IO0IntClr |= (1 << 30); // clear interrupt
 // b) Use fprintf(stderr) or blink and LED here to test your ISR
```

```
 fprintf(stderr, "Interrupt in progress");
}
```

```
--------------------------------------------------------------------------------
peripherals_init(): Low level startup
WARNING: SD card could not be mounted

I2C slave detected at address: 0x38
I2C slave detected at address: 0x64
I2C slave detected at address: 0x72

entry_point(): Entering main()
Starting RTOS
Interrupt in progress
```

Part 1

Main function

```
switch_pressed_signal = xSemaphoreCreateBinary();

configure_your_gpio_interrupt(); // TODO: Setup interrupt by re-using code from
Part 0
NVIC_EnableIRQ(GPIO_IRQn);       // Enable interrupt gate for the GPIO

gpio_s Led0 = gpio__construct_as_output(1, 18);
xTaskCreate(sleep_on_sem_task, "sem", (512U * 4) / sizeof(void *), (void *)&Led0,
PRIORITY_LOW, NULL);
```

Helper function

```
void sleep_on_sem_task(void *p) {
 gpio_s Led0 = gpio__construct_as_output(1, 18);
 fprintf(stderr, "Sleep Function");
 while (1) {
   xSemaphoreTake(switch_pressed_signal, portMAX_DELAY);
   fprintf(stderr, "Toggle LED\n");
   gpio__toggle(Led0);
 }
}

// Step 2:
void gpio_interrupt(void) {
 xSemaphoreGiveFromISR(switch_pressed_signal, NULL);
 LPC_GPIOINT->IO0IntClr |= (1 << 30); // clear interrupt
 fprintf(stderr, "Interrupt cleared\n");
}

void configure_your_gpio_interrupt() {
 gpio_s sw2 = gpio__construct_as_input(0, 30);
 LPC_GPIOINT->IO0IntEnF |= (1 << 30); // enable interrupt
 lpc_peripheral__enable_interrupt(LPC_PERIPHERAL__GPIO, gpio_interrupt, "sw
interrupt");
}
```

```
peripherals_init(): Low level startup
WARNING: SD card could not be mounted

I2C slave detected at address: 0x38
I2C slave detected at address: 0x64
I2C slave detected at address: 0x72

entry_point(): Entering main()
Starting RTOS
Sleep FunctionInterrupt cleared
Toggle LED
Interrupt cleared
Interrupt cleared
Toggle LED
```

Part 2

Main function

```
gpio0__attach_interrupt(29, GPIO_INTR__RISING_EDGE, pin29_isr);
gpio0__attach_interrupt(30, GPIO_INTR__FALLING_EDGE, pin30_isr);

NVIC_EnableIRQ(GPIO_IRQn); // Enable interrupt gate for the GPIO
```

gpio_isr.c

```
static function_pointer_t gpio0_callbacks[32];

void gpio0__attach_interrupt(uint32_t pin, gpio_interrupt_e interrupt_type,
function_pointer_t callback) {
 // 1) Store the callback based on the pin at gpio0_callbacks
 gpio0_callbacks[pin] = callback;
 // 2) Configure GPIO 0 pin for rising or falling edge
 gpio_s sw2 = gpio__construct_as_input(0, pin);
 if (interrupt_type)                    // rising edge
   LPC_GPIOINT->IO0IntEnR |= (1 << pin); // enable inturupt
 else
   LPC_GPIOINT->IO0IntEnF |= (1 << pin); // enable inturupt
 lpc_peripheral__enable_interrupt(LPC_PERIPHERAL__GPIO,
gpio0__interrupt_dispatcher, "sw interupt");
}

// We wrote some of the implementation for you
void gpio0__interrupt_dispatcher(void) {
 // Check which pin generated the interrupt
 const uint32_t pin_that_generated_interrupt = interruptPinDetection();
 function_pointer_t attached_user_handler =
gpio0_callbacks[pin_that_generated_interrupt];

 // Invoke the user registered callback, and then clear the interrupt
 attached_user_handler();
 clear_pin_interrupt(pin_that_generated_interrupt);
}

int interruptPinDetection() {
 uint32_t status = LPC_GPIOINT->IO0IntStatR | LPC_GPIOINT->IO0IntStatF;
 for (uint32_t i = 0; i < 32; ++i) {
   if (status & 0x1)
     return i;
```

```
   status = status >> 1;
 }
 return 0;
}

void clear_pin_interrupt(uint32_t pin) {
 LPC_GPIOINT->IO0IntClr |= (1 << pin); // clear inturupt
}
```

```
--------------------------------------------------------------------------------
peripherals_init(): Low level startup
WARNING: SD card could not be mounted

I2C slave detected at address: 0x38
I2C slave detected at address: 0x64
I2C slave detected at address: 0x72

entry_point(): Entering main()
Starting RTOS
Interrupt pin31
Interrupt pin30
Interrupt pin30
Interrupt pin31
Interrupt pin31
Interrupt pin30
Interrupt pin30
Interrupt pin31
Interrupt pin30
Interrupt pin31
Interrupt pin31
Interrupt pin30
```