Contact: Sebastian Stöcklin
sebastian.stoecklin@imtek.uni-freiburg.de

Winter term 2016/2017

Exercise sheet 3 - Interfacing external ICs

Simple external peripherals have been controlled in the previous exercises. In practical use cases, however, all is about interfacing more complex chips by using external buses. This even holds for input and output devices, so that LEDs and buttons are controlled by dedicated chips in order to reserve microcontroller pins for some more important tasks. Therefore, we will drive the LEDs `D1` to `D4` and read the buttons `PB1` to `PB4` by using a so-called *shift register* as an intermediate device. While these devices tend to be replaced by more advanced circuitry these days (e.g. FPGAs), the signals you send to the shift register are very similar to the signals of the widely used SPI bus. Basically, you implement one of today's most important bus protocols - on your own!

Shift register:
You should already know the basic structure of a shift register from preceding lectures. In principle, it is a device to convert serial data streams into parallel ones and vice versa. You will need 5 out of 8 data lines of the shift register ICs. Specifically, these are:

- `P2.5:/CLR`*: If applying a 0 to this pin, all outputs of the shift register are reset to 0. Hence, you should set it to 1 for the execution of your program.*
- `P2.2:S0`, `P2.3:S1`*: These two pins control the mode of the first shift register.*
- `P2.0:S0`, `P2.1:S1`*: These two pins control the mode of the second shift register.*
- `P2.4:CK`*: Whenever* `CK` *changes from logic 0 to logic 1, the action which is determined by the given mode (see S0 and S1) is performed.*
- `P2.6:SR(2)`*: This pin specifies the next bit that is shifted into the second shift register.*
- `P2.7:QD(1)`*: With this pin, you can read out the fourth out QD of the first shift register.*

The possible modes and their effects can be found on page 2 of the data sheet (`74HC194_OK.pdf`) of the shift register.
The general principle of a single shift operation is as follows: Select the corresponding mode, set `SR` *depending on your data and let the clock change once from 0 to 1. You might now read from* `QD` *and continue with the next action.*

An example to illustrate: Suppose you want to enable LED `D2`, while all other LEDs should be turned off. For simplicity, we assume that all pins have been correctly initialized. Listing **??** shows the code to perform the required operation.

Listing 1: Example for controlling the shift register

```
// Clear/reset the shift register, i.e. turn all LEDs off.
P2OUT &= ~BIT5;

// Shift register 1 (responsible for the buttons) is ignored here.
// Set the shift register 2 mode right shift mode (S0=1,S1=0),
// stop the reset mode (/CLR = 1) and
// apply the data to be shifted at SR (SR=1 to insert a 1).
P2OUT = BIT0 + BIT5 + BIT6;

// apply a rising clock edge => shift data in
P2OUT |= BIT4;
// reset the clock
P2OUT &= ~BIT4;

// set SR to 0, because we don't want D1 to be on
P2OUT &= ~BIT6;

// apply a rising clock edge => shift data in
P2OUT |= BIT4;
// D2 is enabled, D1 isn't
```

Task 1

a) Realize a running light system (or chasing light, see [1] for an illustration; basically, it's involving to enable and disable the LEDs sequentially) with the LEDs D1 to D4. It should have some functions being well known from a tape recorder, being controlled by the buttons PB1 to PB4: functions:

- PB1 is the **rewind button**. As long as PB1 is pressed, a fast light sequence with about 2 passes per second should run from D4 to D1 (**2 pts.**). After PB1 is released, it should return to its previous state (pause or normal playback) (**1 pt.**).
- PB2 should be the **pause button**. If this is pressed, the light should stop at the current position (**1 pt.**).
- PB3 is the **playback button**. If this is pressed, the sequence should continue from the current position (**1 pt.**) and run with approximately 1 pass per second from D1 to D4 (**1 pt.**).
- PB4 is the **Fast-forward button**. As long as PB4 is pushed, a fast light with about 2 passes per second should run from D1 to D4 (**1 pt.**). After PB4 is released, it should return to its previous state (pause or normal playback) (**1 pt.**).

⚠ Make sure that your program responds to the input regardless of the LED's state. (**1 pt.**).

⚠ Note: This time, please also consider the PxSEL register for initialization.

Bonus: Use the LEDs D5, D6 and D7 as a rotating running light, operating in parallel to the linear light (i.e. all features should be synchronous on both scales, **1 pt.**).

[1] http://de.wikipedia.org/wiki/Lauflicht

Task 2

a) Create a file `feedback.txt` with a brief feedback statement, which contains specific problems and issues you experienced while solving the exercise, additional requests, positive remarks, etc. (**1 pt.**).

b) Import this text file `feedback.txt` in your Code Composer Studio project, so that you can upload it together with your software deliverable.