Practice of March 5, 2015

State machines

The objective of this lab is to program state machines with a micro-controller.

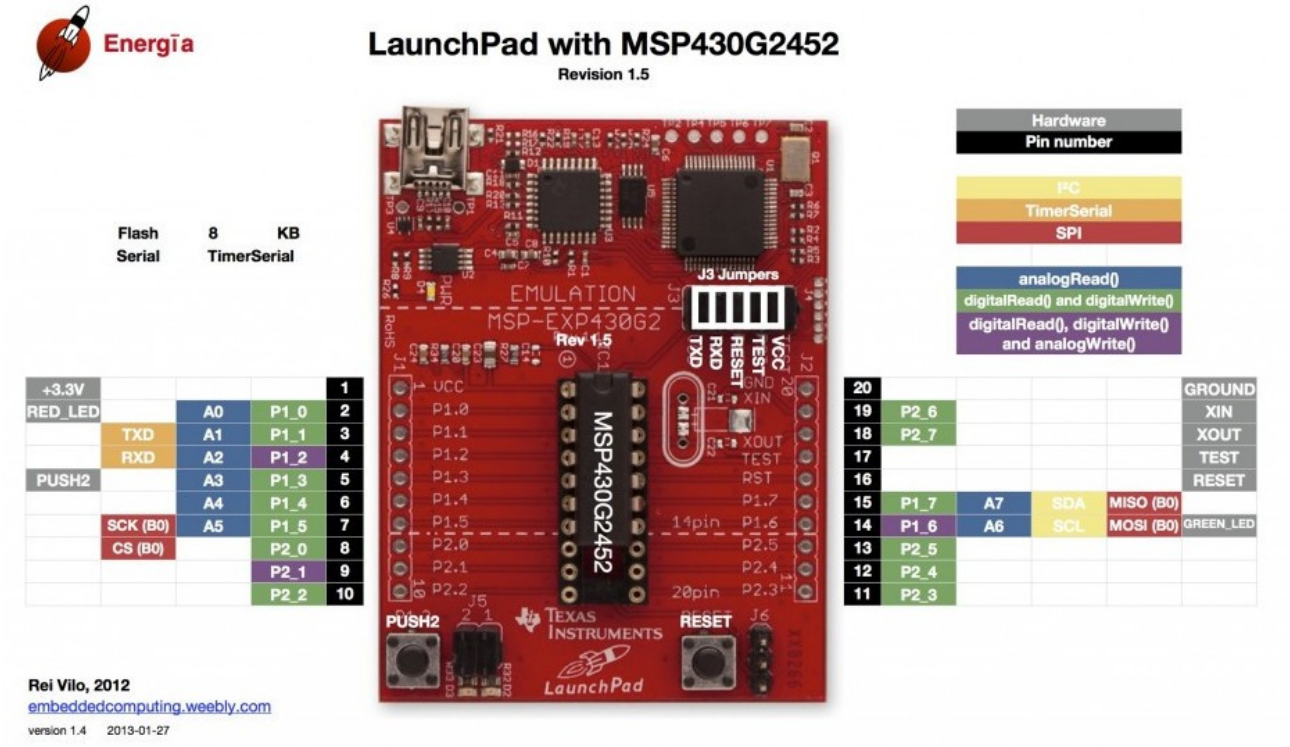
To have up to 3 push-buttons with the LaunchPad, a shield was prepared, which is placed on the Launchpad. The two additional buttons are on P1.4 and P1.5. They can be read by digitalRead (). You can use these definitions :

```
#define Pous2On (!digitalRead (P1_4))
#define Pous3On (!digitalRead (P1_5))
```

Do not forget to use corresponding initialisations in the `Setup()` :

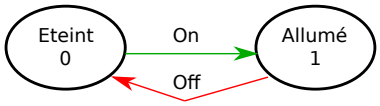
```
pinMode (P1_4, INPUT_PULLUP);
pinMode (P1_5, INPUT_PULLUP);
```

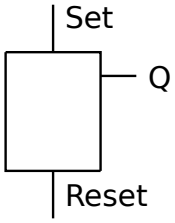
For information, here is a useful table showing the use of LaunchPad pins :



1) Set-Reset flip-flop

The Set-Reset flip-flop is probably the simplest state machine you can imagine. It has two states. Here is his state graph and electronic symbol :





It is easy to implement it by the following program :

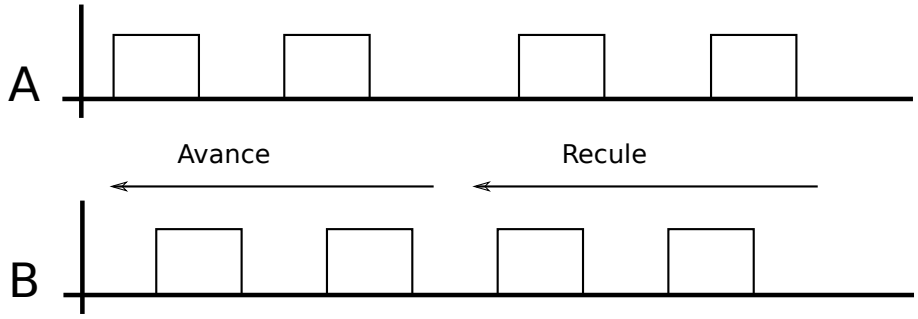
```
void loop () {
  if (Pous2On) { LedRougeOn) ; }
  if (Pous3On) { LedRougeOff) ; }
}
```

Rewrite a more complex program for the same purpose, using the method proposed in the course :

- use a **state variable**
- in the main loop, select each possible state (with a `switch... case` structure or with `if...` statements)
- for each state, select the corresponding outputs : detect each **transition** in the state it comes from (by a test on the corresponding inputs) and then modify the stage variable.

2) Stepper Moteur

A simple stepper motor control can be done with two-phase signals :



Look at http://en.wikipedia.org/wiki/Stepper_motor for more détails

or look at the video :http://pyr.ch/coursera/pas-a-pas.mp4

You can see that the good sequence is used to select the direction of the rotation.

Draw the system status graph.

Program the motor control that moves forward or backward at a fixed speed of one step per second, depending on the input ***Direction***. Use the reg and green LEDs for the outputs and the P1.3 Push-button for the Direction.

Practice *: same assemblies with a step-by-step motor will be available during the practice. You can make them running ! Use P2.0 and P2.1 pins for the connection with Logidules wires.*

3) Time management

While continuing to apply the method for writing a state machine program, build a timer for a staircase lighting. *Instead of 3 minutes, use 3 seconds for your tests...*

Draw the state graph. Notice that a transition is no more a condition of input, but the end of the timer.

You can program it like this :

```
void loop () {
  switch state {
  case ...
  case ...
  }
  if (Timer >0) {
    Timer--;
  }
  delay (1);
}
```

The main loop is clocked at 1 kHz (by a 1ms delay). The `Timer` variable is decremented each cycle when it is upper then zero.

When a transition must be the beginning of a wait, the `Timer` variable is assigned to the value of the waiting (for *3 seconds : `Timer = 3000`*). The transition is the end of the waiting can be written by : `if (Timer == 0) { state = …} ;`

4) *(Send your functional code to pyr@pyr.ch)*

Improve the control of the drilling machine, making the Start button becomes as an emergency stop during movement of the drill. A subsequent press on Start button make the drill going back on top.

Warning: new states will be required. For example, the fact that the drill is in the down state is not sufficient for the Start button to stop the movement: the system would immediately be stopped, even before the start button is released !