4. Read inputs

- Read all input ports
- Mask and shift
- Save variable values

```
var_buf = PORTB & 0x00000000;
var_buf2 = (var_buf & 0x01000000);
var_buf2 = var_buf2 >> 6;
var_buf2 = (var_buf & 0x00000000);
var CW = var_buf2 >> 7;
```

5. Write outputs

- Read all port, save unused pins and clear pins to be written
- Shift variables and write port in a single instruction

```
var_buf = PORTA & 0x01111111;
var_buf2 = (var_L & 0x00000011) << 1;
PORT A = var_buf | var_buf2;
var_buf2 = (var_L & 0x00000100) << 2;
var_buf = PORTA & 0x11111111;
PORT B = var_buf | var_buf2;
var_buf2 = var_A << 1;
PORT B = var_buf2 | var_buf;
```

6. State logic truth table and flowchart

- (only when var CLK_flag = 1)
- var CW var current state + after reading the C instruction

<table>
<thead>
<tr>
<th>var CW</th>
<th>var.current_state</th>
<th>var.current_state +1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>state 3</td>
<td>state 9</td>
</tr>
<tr>
<td>0</td>
<td>state 9</td>
<td>state 5</td>
</tr>
<tr>
<td>0</td>
<td>state 10</td>
<td>state 10</td>
</tr>
<tr>
<td>1</td>
<td>state 10</td>
<td>state 11</td>
</tr>
<tr>
<td>0</td>
<td>state 9</td>
<td>state 10</td>
</tr>
<tr>
<td>0</td>
<td>state 9</td>
<td>state 9</td>
</tr>
<tr>
<td>1</td>
<td>state 10</td>
<td>state 10</td>
</tr>
<tr>
<td>0</td>
<td>state 10</td>
<td>state 11</td>
</tr>
<tr>
<td>1</td>
<td>state 9</td>
<td>state 9</td>
</tr>
<tr>
<td>0</td>
<td>state 10</td>
<td>state 10</td>
</tr>
</tbody>
</table>
7. output_log(): truth table and flowchart

<table>
<thead>
<tr>
<th>var.current.state</th>
<th>var.Q</th>
<th>var.L</th>
</tr>
</thead>
<tbody>
<tr>
<td>state5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>state10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>state9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>state6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

IF
var.clk_flag = 1?

YES: SWITCH
CASE

IF
var.current.state = state9

var.current.state = state5

END

var.current.state = state6

clear clk_flag
var.clk_flag = 0

var.L = 9
var.Q = 9

var.L = 5
var.Q = 5

var.L = 10
var.Q = 10

var.L = 6
var.Q = 6

var.clk_flag = 0

var.Q = 0b00000001

END
Using this planning, it is straightforward to obtain the C code and the Proteus schematic copying and adapting files from another P10 example project in [digsys.upc.edu](http://digsys.upc.edu).

In addition, this stepper motor controller can be written as another example of learning materials in:
- P10. Phase #1: Stepper motor controller using an external CLK.
- P12. Phase #2: Stepper motor controller using TM0 to generate CLK, as an improvement from phase #1.

Logically, even previously to µC adaptations, the same project stepper motor controller can be organised in P6 as a canonical FSM using VHDL and EDA tools for a FPGA/CPLD target chip. A programmable CLK generator can be implemented in P8 so that the motor can operate at different rotation angular speeds.