Project 1

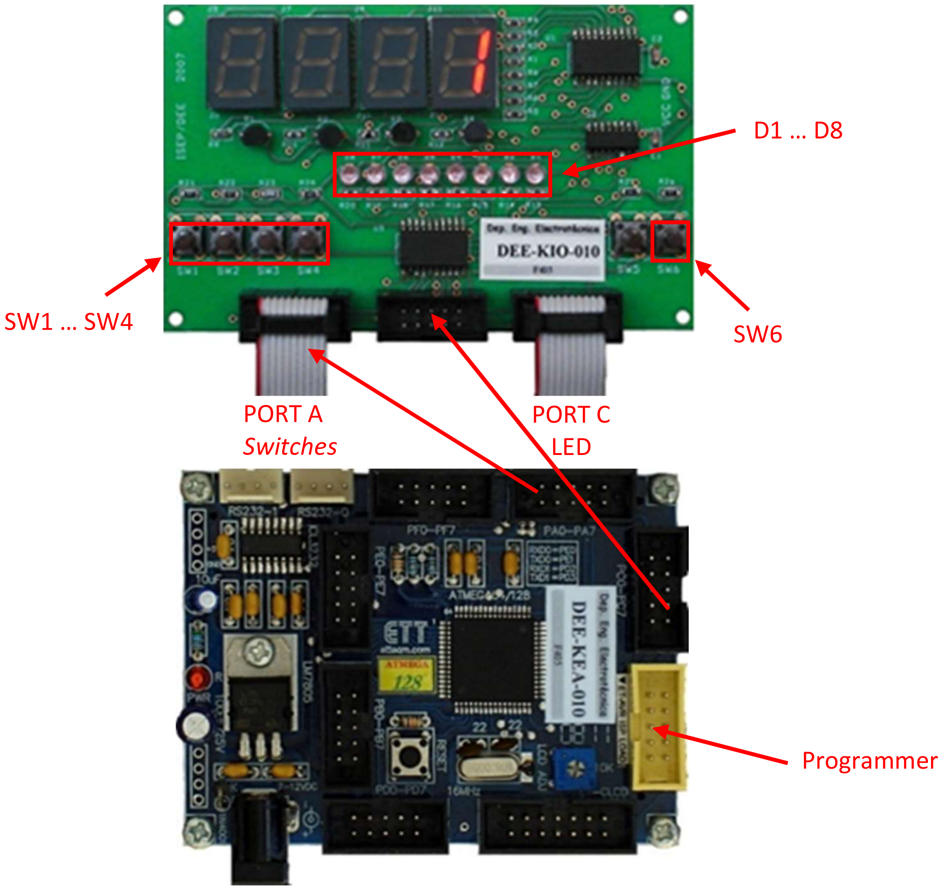
Objective: implementation of simple data input/output operations, usage of timers/counters and external interrupts.

Requirements: knowledge of the µC architecture, memory architecture and I/O ports, knowledge of Timers/Counters (TC0 in particular) and interrupts.

Operation 1: control the state of the LEDs D1…D8, using the switches SW. The LEDs state should obey the following table:

Active switch	LEDs ON
SW1	D4, D5
SW2	D3, D6
SW3	D2, D7
SW4	D1, D8
SW6	(All LEDs OFF)

Necessary hardware



Switches port

Pin no.	Function	Pin no.	Function
1	SW1 (PA.0)	2	SW2 (PA.1)
3	SW3 (PA.2)	4	SW4 (PA.3)
5	SW5 (PA.4)	6	SW6 (PA.5)
7	MUX.0	8	MUX.1
9	Vcc	10	Ground

Note: according to the hardware of the I/O board, the default (OFF) state of the switches corresponds to the logical value 1, the active (ON) state corresponds to the logical value 0.

LEDs port

Pin no.	Function	Pin no.	Function
1	D1 (PC.0)	2	D2 (PC.1)
3	D3 (PC.2)	4	D4 (PC.3)
5	D5 (PC.4)	6	D6 (PC.5)
7	D7 (PC.6)	8	D8 (PC.7)
9	Vcc	10	Ground

Note: according to the hardware of the I/O board, to turn ON a LED the respective pin number should have the logical value 0, to turn OFF a LED use the logical value 1.

Software implementation

- Use **Assembly programming language**

Operation 2:

starting with all 8 LEDs (**D1 … D8**) turned OFF, it is intended that, when pressing switch **SW1**, the LEDs are sequentially activated starting with LED **D1(D1->D8)**. The time between events starts at **2 s** and diminishes by **300 ms** as each LED is activated. When all the LEDs are activated, and after a **3 s** delay, the LEDS must be sequentially turned OFF in the reverse order (**D8->D1**). In this case, the time between events must be **1 s**. Pressing switch **SW6** stops the sequence of events.

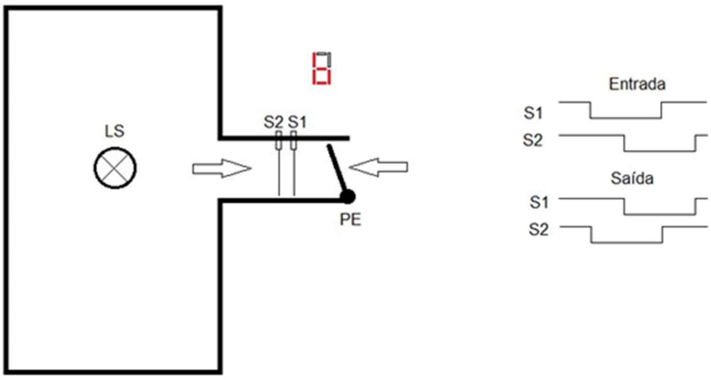
Software implementation

- Use **Assembly programming language**

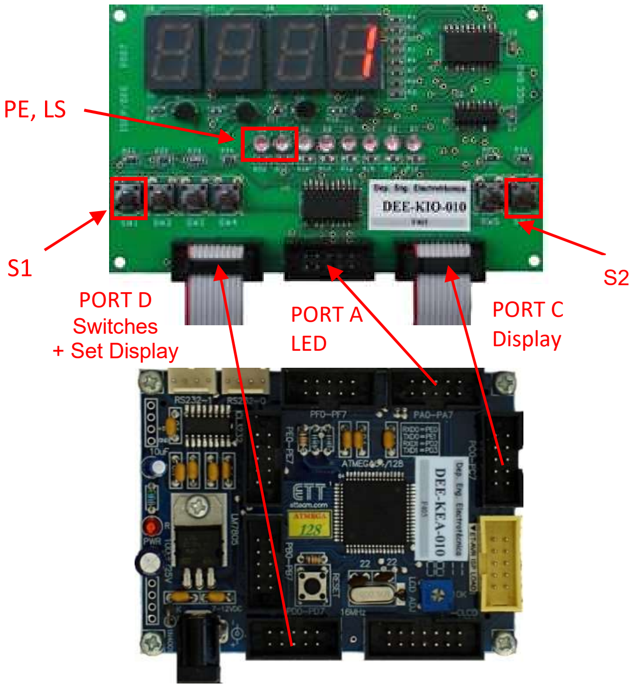
Operation 3: simulate the access control system of a room with maximum lotation of **9** people. The entrance and exit of people is made using the same location and the passage of people is detected by sensors **S1** and **S2**. Every time the lotation reaches the maximum value, the door **PE (D8)** must be closed to prevent more people coming in. The display on the right must show the **number of vacant places** in the room. The room light **LS (D7)** must be turned OFF everytime the room is empty.

The sensors **S1** and **S2** are placed with a phase shift of 90º to allow the detection of people entering and leaving. At the entrance, people are detected by the sensor **S1**, first, followed by sensor **S2**, while at the exit, first the sensor **S2** is activated, followed by sensor **S1**.

To determine the value of sensors **S1** and **S2**, for each sensor, two readings must be made, with a time interval of **1 ms**, and both readings need to give the same logical value for the reading to be considered valid.



Necessary hardware



Switches port

Pin no.	Function	Pin no.	Function
1	**S1 (PD.0)**	2	SW2 (PD.1)
3	SW3 (PD.2)	4	SW4 (PD.3)
5	SW5(PD.4)	6	**S2 (PD.5)**
7	**MUX.0 (PD.6)**	8	**MUX.1 (PD.7)**
9	Vcc	10	Ground

Note: PD.0 .. PD.5 should be programmed as data input to obtain the information from the switches. PD.6 e PD.7 should be programmed as data output and both pins should be updated with the logical value 1 to select the display placed on the right side.

Displays port

Pin no.	Function	Pin no.	Function
1	Seg a (PC.0)	2	Seg b (PC.1)
3	Seg c (PC.2)	4	Seg d (PC.3)
5	Seg e (PC.4)	6	Seg f (PC.5)
7	Seg g (PC.6)	8	DP (PC.7)
9	Vcc	10	Ground

Note: to turn ON one segment of the seven segment display, the respective pin number should have the logical value 0, to turn OFF one segment use the logical value 1.

Segment table

Digit	DP	Seg g	Seg f	Seg e	Seg d	Seg c	Seg b	Seg a	PORT C
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	0	1	0x99
5	1	0	0	1	0	0	1	0	0x92
6	1	0	0	0	0	0	1	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	0	0	0	0	0x90

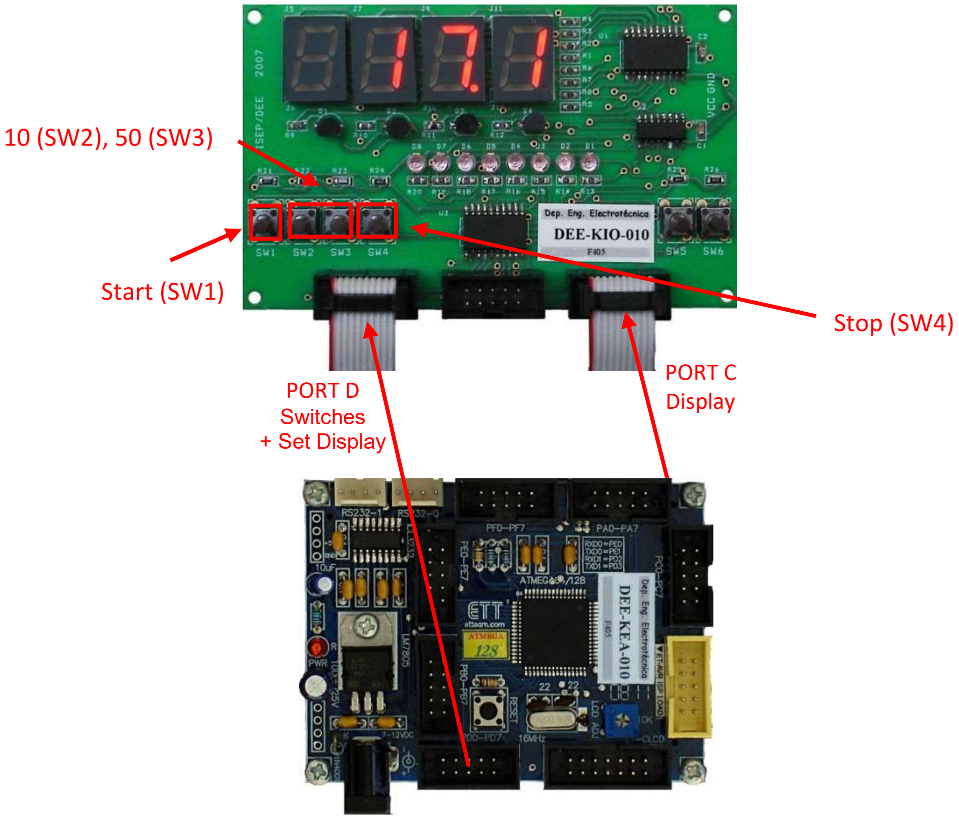
Software implementation

- Use **Assembly programming language**

Operation 4:

using one of the 7 segment displays, an electronic dice is to be implemented, by showing a sequence of **6 digits**. Every time the ***start*** switch is pressed, the display must show the digit **"1"** to the digit **"6"** at a **20 ms (50 Hz)** rate. When digit **"6"** is reached, the display must restart with digit **"1"**. Activating the ***stop*** switch, during the dice/roulette operation, the sequence must be halted and the current digit shown blinking at a frequency of **1 Hz**. After **5 seconds** the roulette must finish it's operation and the display must show the digit without blinking. At the initial state, the display must be off with no digit showing.

Necessary hardware



Suggestion: use the TC0 in mode 2 ("Clear Timer on Compare") to generate a time base of 2 ms.

Software implementation

- Use **Assembly programming language**

Operation 5: using **2 seven segment displays**, it is intended to create a game similar to the rolling of 2 dices. The winner of the game is the player that achieves the larger number os points. Using the code from operation 4, the software must be altered to implement the game using **2 seven segment displays**, considering that **display 0** increments and **display 1** decrements the value of the digits in the roulette sequence. The game starts when switch ***start*** is pressed, with the **2 displays** working in simultaneous. When the switch ***stop*** is activated, the **display 0** stops rolling but **display 1** continues to roll. The time interval Δ**t**, in seconds, between activating the ***start*** switch and the ***stop*** switch must be saved in a register (maximum value of 255 s). The **display 1** must stop rolling only after Δ**t/2** seconds have passed from the moment the ***stop*** switch is activated. At the end, the ***displays*** **0** and **1** must be shown blinking at a frequency of **1 Hz** during a **5 second** interval. The switches **SW2** (10 Hz) and **SW3** (50 Hz) are used to select the frequency of the roulette operation in both displays. This frequency can be altered at any time during the game.

Software implementation

- Use **Assembly programming language**