INTERRUPTS

TIMER 0

- **Exercise 3.1.** Write a program using the Timer 0 module and its interrupt on overflow to increase variable *CICLO_125us* every 125 µs.

- **Exercise 3.2.** Write a program using the Timer 0 module and its associate interrupt that increases the value of variable *CICLO_35ms* every 35 ms and shows its value in 8 LEDs connected to PORTC.

- **Exercise 3.3.** Write a program that uses the Timer0 module and its associate interrupt in order to generate in pins RC[3:0] of PORTC the following sequence of values. These values will be displayed in LEDs S[3:0].

pin	RC3	RC2	RC1	RC0
State1	1	1	0	0
State2	0	1	1	0
State3	0	0	1	1
State4	1	0	0	1

Initially, the value in RC[3:0] must be the associated to State1 in the Table. After a period of time, the value must change to the State2, and so on. Once the time in State4 has finished, RC[3:0] must go back to State1, cyclically repeating the sequence.

The period of time of each state is $N \times 50$ms, where N is the value in pins RB[7:3], the 5 most significant bits of PORTB. Additionally, if N is 0 the sequence is halted.

Set the PICSchool board interface connections as follows:

pin	RB7	RB6	RB5	RB4	RB3
sw	E3	E2	E1	E0	E7

Interrupt on edge on RB0/INT

- **Exercise 3.4.** Write a program that increments the value of PORTC when a push button connected to pin RB0 is released. PORTC must be cleared at initialization. In order to debounce (i.e. to filter spurious transitions out of the output signal) the push button, a delay loop (exercise 1.2) of 30 ms must be executed at the beginning of the Interrupt Service Routine, before clearing the INTF flag.

Interrupt on change RB[7:4]

- **Exercise 3.5.** Write a program that increments the value of PORTC when a button connected to pin RB4 is pressed, decrements the value of PORTC when a button connected to RB5 is released and resets the value of PORTC when a button connected to RB6 is pressed. In order to debounce the push buttons, a delay loop (exercise 1.2) of 30 ms must be executed at the beginning of the Interrupt Service Routine.

Managing multiple interrupts

- **Exercise 3.6.** Modify the sequence generator described in exercise 3.3 according to the following functionality:
  - The period of time at each state in RC[3:0] is *N* × 50 ms. However, unlike exercise 3.3, the value of *N* is controlled by two push buttons connected to RB6 and RB5: when button connected RB6 is pushed, *N* increases; when button connected RB5 is pushed, *N* decreases. The initial value of *N* is 7; its maximum value is limited to 15 and the minimum to 1. The value of *N* will be displayed in 4 LEDs (S[7:4] in PICSchool) connected to pins RA[3:0] in PORTA.
  - The generator can be set ON or OFF by releasing a button connected to RB7. Initially, the generator is OFF, i.e., the sequence is halted (with RC[3:0] in State1). When the button is released, the generator switches ON, i.e., the sequence starts. Each time the button is released, the generator changes its state.
  - The sequence can run both forward and backward, configurable by a switch connected to pin RB0: if RB0 is '1', the sequence runs as indicated in Table 1; if '0', the sequence must run reversed.

  Use Timer0 interrupt to generate the timing required. Use the interrupt on change in RB[7:4] to detect the pushing or releasing of push buttons. Use RB0 interrupt to detect the change of the switch that controls the direction.