

# UN ESEMPIO DI INPUT DIGITALE


*In questo fascicolo sperimentiamo come sfruttare il linguaggio messo a disposizione dall'ambiente mikroC per interpretare un input digitale generato da un micropulsante esterno azionato dall'utente.*

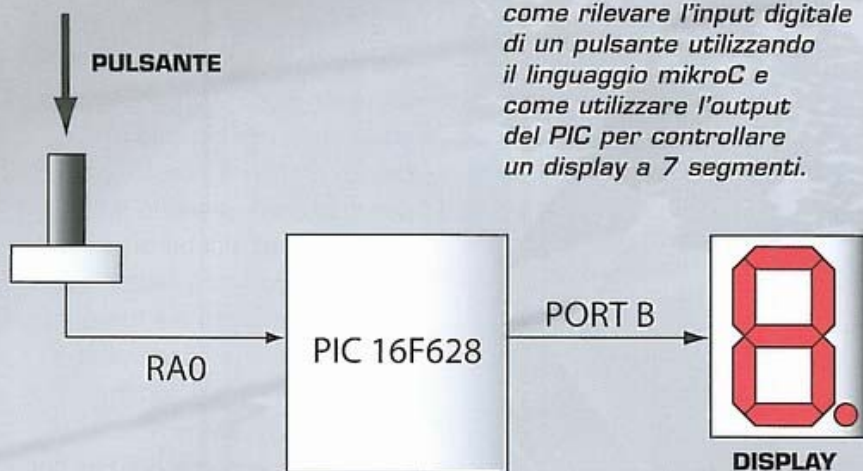
**N**el Workshop 61 abbiamo sperimentato su breadboard un primo semplice firmware orientato alla gestione dell'output digitale di alcuni pin del 16F628. In questo Workshop vediamo, invece, **come utilizzare la porta A del microcontrollore come ingresso logico** atto a rilevare il click di un **pulsante**. A ogni click il PIC incrementerà una variabile interna (con valore da 0 a 9), che verrà visualizzata per mezzo di un **display LED a 7 segmenti** (questo componente sarà analizzato più dettagliatamente nel prossimo fascicolo) controllato dal firmware stesso attraverso la porta B.

## L'INPUT DIGITALE >>>

Quando abbiamo introdotto i microcontrollori abbiamo osservato come i pin delle porte digitali possano operare sia come **ingressi**, sia come **uscite** logiche. Nei Workshop precedenti abbiamo già sperimentato l'impostazione delle porte dei PIC come uscite attraverso **l'inizializzazione dei registri TRIS. Vediamo ora come utilizzare gli stessi pin come linee di input digitale.** Ricordando quanto detto sui registri TRIS, la differenza più evidente con quanto fatto nell'esperimento precedente risiede **nel valore dei bit del registro TRISA e in particolare**

**nell'assegnamento del valore '1' alla cella di registro associata al bit 0 della porta A** (ricordiamo che **bit = 1: ingresso, bit = 0: uscita**). La prima istruzione di configurazione del PIC dovrà quindi riguardare proprio la porta A e in particolare il suo **registro di configurazione TRISA**, che dovrà essere **inizializzato al valore binario 0b00000001** (in questo modo impostiamo come ingresso solo il pin 0 della porta, mentre lasciamo come uscite tutti gli altri). Un'ulteriore osservazione, tuttavia, deve essere fatta in merito alla **struttura interna di questa porta**. Come puoi vedere dalla piedinatura del PIC che ti è stata presentata a pagina 6 del fascicolo 56, i pin della porta A, e in particolare il pin RA0, non hanno esclusivamente funzioni di ingressi digitali, ma **possono operare anche in modalità analogiche (ANO, AN1...)**. È quindi necessario indicare al microcontrollore quale tipo di ingresso si vuole attivare. Nel caso del 16F628 ciò è possibile andando ad agire sul **registro interno CMCON**,

 *L'esperimento di questo Workshop ti mostrerà come rilevare l'input digitale di un pulsante utilizzando il linguaggio mikroC e come utilizzare l'output del PIC per controllare un display a 7 segmenti.*



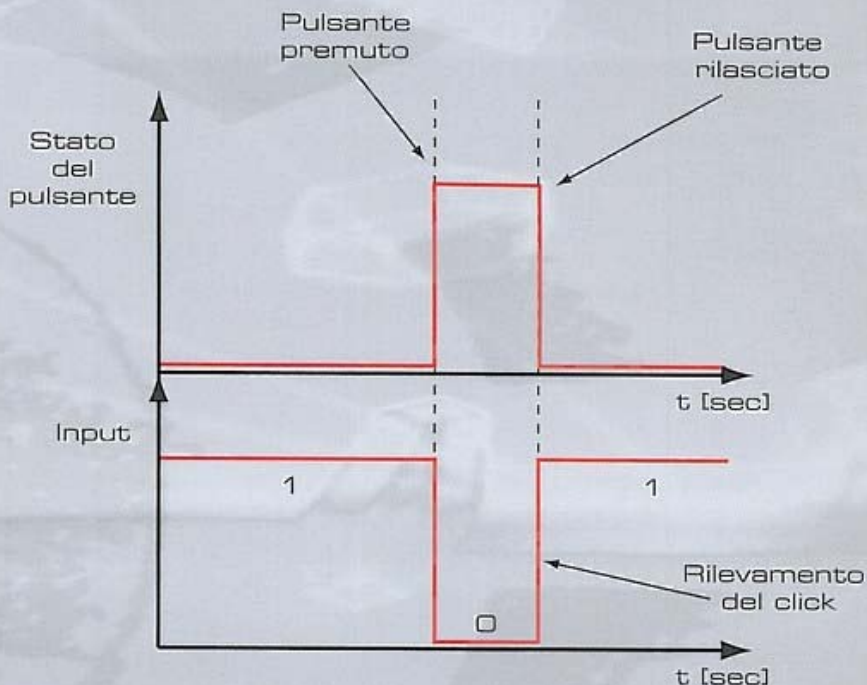
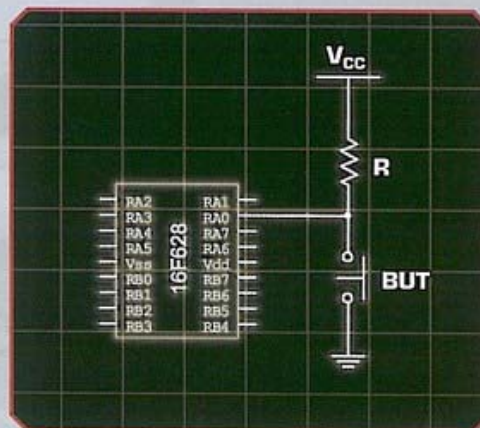


utilizzabile in mikroC per mezzo della omonima variabile (per una comprensione completa della funzione svolta da questo registro della famiglia 16F62X di PIC, ti rimandiamo al datasheet ufficiale del dispositivo, facilmente reperibile via Internet). In particolare, per l'esperimento odierno, ciò che dovremo fare sarà **disabilitare tutti i comparatori del PIC**, in modo da attivare la modalità di I/O logico della porta. Ciò può essere fatto **portando i tre bit meno significativi del registro CMCON a 1**, attraverso il semplice assegnamento **CMCON = 0b00000111;**. La porta A è, a questo punto, pronta e **il pin RAO può essere utilizzato come ingresso**. Ma come rilevare il valore in input sul pin? Semplicemente **leggendo lo stato del bit 0 della variabile di registro PORTA**. Le variabili PORT di mikroC, quindi, servono sia per impostare lo stato di uscita delle porte, sia a rilevare gli stati di ingresso (in funzione di come viene configurato il registro TRIS associato).

**CLICK! >>>**

Dedichiamoci ad analizzare il circuito di ingresso che utilizzeremo per interagire con il microcontrollore. La sua struttura è estremamente semplice: esso sarà costituito da **un resistore** e da **un micropulsante**, collegati opportunamente tra il 16F628, la **tensione di alimentazione** e la **massa**. I micropulsanti sono dispositivi elettromeccanici elementari utilizzatissimi in moltissimi apparati elettronici,

➤ *A destra, il circuito che permette di collegare il pulsante al PIC 16F628. Quando il pulsante (BUT) è aperto sul pin RAO abbiamo una tensione  $V_{IN}$  alta; a pulsante chiuso, invece, bassa.*



➤ *Sopra, la rappresentazione temporalizzata dell'evento 'click', ottenuto come sequenza dei due eventi elementari 'pressione' e 'rilascio' di un pulsante. In alto è mostrata l'evoluzione dello stato del pulsante, in basso lo stato elettrico dell'ingresso.*

anche domestici. Tecnicamente sono molto simili agli interruttori o ai microswitch che abbiamo usato in passato, con la differenza che **mentre gli switch sono caratterizzati da una struttura meccanica che li mantiene nello stato prescelto (aperto o chiuso), i micropulsanti permangono**

**nello stato di attivazione esclusivamente per la durata della pressione dell'utente**. Proprio da questa loro peculiarità nasce il concetto di 'click'. Il click è un evento tipicamente identificabile nei sistemi informatici, che si genera nel momento in cui si verifica la pressione di un



pulsante (reale o virtuale), seguita immediatamente dal suo rilascio (vedi schema nella pagina precedente). Osserva ora lo schema elettrico posto in alto a destra nella pagina precedente: in esso vedi rappresentato il circuito elettronico che utilizzeremo come sistema di interazione con l'utente. Come puoi osservare, **in condizioni di normalità il pulsante si comporta come un circuito aperto**. Ciò significa che **sul pin di ingresso del microcontrollore vi sarà una tensione pari a  $V_{CC}$** , ossia un **valore logico alto**. Se però chiudiamo l'interruttore agendo sul pulsante, non facciamo altro che **cortocircuitare a massa il pin di ingresso del PIC, portandolo di fatto a un valore logico basso**. A livello di firmware, quindi, l'evento 'click' può essere rilevato **monitorando ciclicamente il valore presente sull'ingresso RAO**, in attesa della pressione del pulsante, che porta il valore logico di input a 0. **Una volta intercettato questo evento,**

**il programma dovrà rimanere in attesa del rilascio del pulsante** in modo da generare l'evento 'click'. Nel box a fondo pagina puoi osservare una possibile sequenza di pseudocodice che implementa il processo appena descritto.

#### IL DISPLAY NUMERICO >>>

Come detto in precedenza, all'evento 'click' corrisponderà la variazione di una variabile intera interna al microcontrollore. Il valore di tale variabile viene visualizzato grazie a un dispositivo **chiamato display LED a 7 segmenti**, un semplice ed economico componente elettronico basato su diodi a emissione luminosa, che si attivano agendo opportunamente sui pin di controllo. Nel prossimo Workshop troverai un breve approfondimento che ti aiuterà a capire meglio la struttura interna di questi sistemi elettronici. Per poter procedere, però, è necessario comprenderne, anche se solo in modo approssimativo,

il funzionamento. Senza addentrarci eccessivamente in dettagli, per ora ti è sufficiente sapere che il display a 7 segmenti che utilizzeremo per il nostro esperimento avrà a disposizione **10 piedini, 8 dei quali associati agli anodi dei LED costituenti i segmenti luminosi del display (7 segmenti più un punto per i decimali)**. L'attivazione dei singoli elementi grafici luminosi avviene, come sempre, portando i LED in condizioni di **polarizzazione diretta**. In particolare, **i 7 segmenti saranno controllati dai primi 7 pin della porta B del 16F628, secondo la rete di collegamenti mostrata nello schema in alto nella pagina successiva**. Ogni visualizzazione grafica corrisponde a una ben precisa combinazione di bit della porta B. Introducendo l'esperimento abbiamo specificato che i click agiranno su una variabile, che potrà assumere **valori compresi tra 0 e 9 (inclusi)**. **Dobbiamo, di conseguenza, poter rappresentare le 10 cifre**

Il ciclo di controllo del click in pseudocodice

```
Esegui in loop
{
  ...
  if(1'ingresso RAO è basso) //viene rilevata la pressione del pulsante
  {
    while(RAO è basso) // finchè RAO è basso, rimango in loop
    {
    }

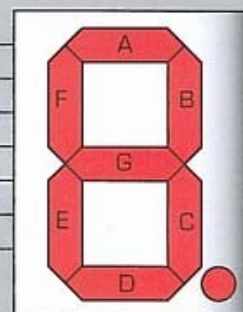
    //RAO è tornato alto, il pulsante è stato rilasciato.
    Esegui le funzioni associate al click;
    ...
  }
  Azioni da eseguire ciclicamente;
  ...
}
```



**in modo grafico.** Nella tabella a fondo pagina puoi vedere schematizzati i 10 possibili stati del sistema. Per ogni valore ti viene mostrata la rappresentazione grafica del numero per mezzo del display e gli stati di attivazione dei sette segmenti (l'indicizzazione dei singoli segmenti è riferita allo schema in alto a destra). Nell'ultima colonna vi è, infine, **un valore numerico:** tale valore corrisponde al **valore di output della porta B in grado di produrre l'effetto grafico desiderato**, considerando che **i segmenti, ordinati da A a G, verranno collegati uno a uno secondo un ordine crescente**



RB0	A
RB1	B
RB2	C
RB3	D
RB4	E
RB5	F
RB6	G



Lo schema dei collegamenti che utilizzeremo per pilotare il display dalla porta B del PIC 16F628.

con i piedini della porta B (RB0 con segmento A, RB1 con segmento B ecc.). In questa maniera ogni segmento luminoso è abbinabile a uno specifico bit della porta B e l'intero display può essere

controllato utilizzando un semplice numero intero a 8 bit.

**IL CODICE SORGENTE >>>**

Nella prossima pagina puoi vedere il sorgente del firmware, che realizza il nostro progetto.

Valore da visualizzare	DISPLAY	A 2 <sup>0</sup>	B 2 <sup>1</sup>	C 2 <sup>2</sup>	D 2 <sup>3</sup>	E 2 <sup>4</sup>	F 2 <sup>5</sup>	G 2 <sup>0</sup>	OUTPUT
0		1	1	1	1	1	1	0	63
1		0	1	1	0	0	0	0	6
2		1	1	0	1	1	0	1	91
3		1	1	1	1	0	0	1	79
4		0	1	1	0	0	1	1	102
5		1	0	1	1	0	1	1	109
6		0	0	1	1	1	1	1	125
7		1	1	1	0	0	0	0	7
8		1	1	1	1	1	1	1	127
9		1	1	1	1	0	1	1	111





Esempio



```
void main()
{
    /* variabile valore_attuale che funge da indice di scansione per l'output e che
       viene incrementata a ogni click*/
    int valore_attuale;

    /* il vettore configurazioni_LED viene inizializzato per contenere la serie di
       output per la porta B che permettono di visualizzare le cifre da 0 a 9 */
    int configurazioni_LED [] = {63, 6, 91, 79, 102, 109, 125, 7, 127, 111};

    /* inizializziamo a 0 l'indice che verrà incrementato con i click */
    valore_attuale = 0;

    /* impostazione della direzione delle porte A e B */
    TRISB = 0b00000000; /* porta B in output */
    TRISA = 0b00000001; /* pin 0 della porta A in input, gli altri in output */

    CMCON = 0x7; /* Disattivazione dei comparatori della porta A*/

    /* imposto l'output della porta B con la configurazione di LED indicizzata dalla
       variabile 'valore_attuale' */
    PORTB = configurazioni_LED[valore_attuale];

    /* ciclo infinito */
    while(1)
    {
        /* se il pin 0 della porta A (RA0) ha ingresso basso significa che è stato
           premuto il pulsante */
        if(PORTA.F0 == 0)
        {
            while(PORTA.F0 == 0) /* finchè il pulsante è premuto */
            {
                /* il pulsante è rilasciato e il valore logico di RA0 è tornato alto*/

                /* incremento il valore dell'indice */
                valore_attuale++;

                /* se il valore dell'indice è maggiore di 9, lo reinizializzo a 0*/
                if(valore_attuale>9) valore_attuale=0;
            }

            PORTB = configurazioni_LED[valore_attuale]; /* invio la configurazione dei LED
                indicizzata da 'valore_attuale' sulla porta B */
        }
        /* ripeti il ciclo */
    }
}
```