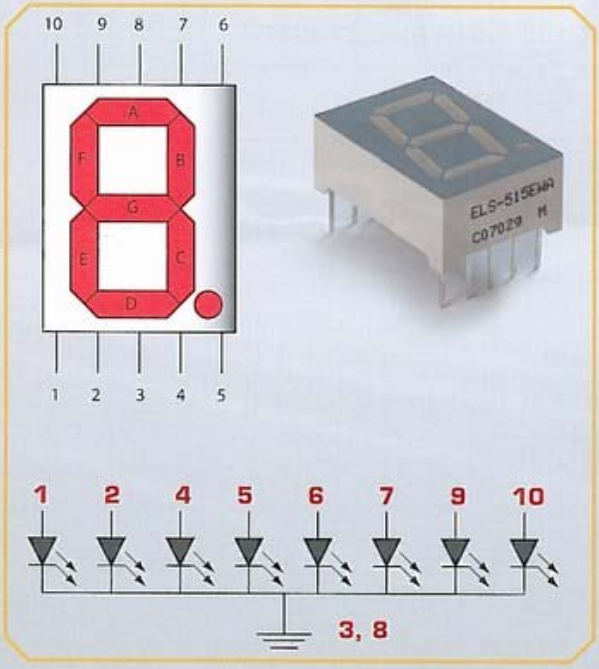
SPERIMENTIAMO L'INPUT DIGITALE DEL PIC 16F628

Dopo aver analizzato il progetto nel fascicolo precedente, realizziamo il circuito che ci consentirà di testare il nostro firmware.

Eccoci giunti alla fase di sperimentazione del progetto visto nel fascicolo 62. Per poter collaudare il firmware dovrai prima di tutto **creare un nuovo progetto mikroC, compilare il sorgente presentato nel fascicolo predente** e successivamente **caricare il binario creato nel 16F628.** Prima di passare allo StepbyStep vero e proprio, però, ti proponiamo un **FOCUS ON sui display a 7 segmenti,** che ti aiuterà a comprendere più a fondo come utilizzare questo tipo di dispositivi.

FOCUS ON

I DISPLAY A 7 SEGMENTI



I display a 7 segmenti sono dispositivi di output costituiti da 7 diodi a emissione luminosa disposti in modo tale da rendere visualizzabili (mediante l'accensione controllata) numeri, lettere e simboli. Il loro uso è molto diffuso sia nelle applicazioni domestiche (ad esempio in orologi e radiosveglie), sia in applicazioni più 'tecniche' (pannelli per la lettura dati ecc.). Vediamo come operano. Come detto, ogni display a 7 segmenti è basato generalmente su 7 LED (o 8 se il display permette di visualizzare anche il 'punto' separatore per i decimali), che devono essere comandati per mezzo di altrettanti pin. Nello schema a sinistra puoi vedere una disposizione tipica dei segmenti luminosi di un display. Ma, come abbiamo visto nel fascicolo 12, ogni LED richiede per il suo funzionamento l'utilizzo di due contatti (anodo e catodo): quindi quanti sono in realtà i pin di questi componenti?

Ovviamente, se tutti i LED fossero completamente indipendenti ogni display avrebbe una quantità di pin molto elevata, caratteristica che li renderebbe poco pratici per via delle numerose connessioni richieste. Si adottano, invece, due tipi di architetture (analoghe a quanto già visto nel **FOCUS ON** del fascicolo 12), dette rispettivamente a catodo e ad anodo comune. In queste due tipologie di configurazioni tutti i diodi presenti nel display sono inseriti saldando assieme i catodi o gli anodi di tutti i diodi presenti (da qui il termine 'comune'). Considera che la differenza di funzionamento tra queste due classi di elementi è sostanziale: nei display a catodo comune, infatti, uno o due pin (in base al modello) sono direttamente saldati ai catodi dei LED e richiedono di essere collegati alla massa del circuito. Gli altri 7 (o 8) pin sono, invece, gli anodi dei diodi: nel momento in cui nell'anodo viene immessa corrente (polarizzando direttamente il LED con una tensione sufficientemente alta) il segmento corrispondente si accende emettendo luce. Nel caso dei display ad anodo comune, al contrario, si ha l'anodo da collegare alla tensione Vcc e i 7 (o 8) catodi che devono essere impiegati per 'prelevare' corrente cosa possibile con un transistor configurato a emettitore comune o assorbendo corrente (corrente di sink) attraverso un circuito digitale. Creando un parallelo con il caso precedente, quindi, mentre nei dispositivi a catodo comune l'accensione dei segmenti si ottiene portando al livello elettrico alto gli anodi, nel caso di display ad anodo comune, per attivare i singoli segmenti il circuito di controllo deve portare bassi i catodi. Ovviamente, anche in questo caso è necessaria una serie di resistori che limitino la corrente passante nei diodi. Nella tabella sottostante, puoi osservare la mappatura dei pin del display a catodo comune che utilizzeremo nel corso dello StepbyStep (modello ELS-515EWA). La numerazione dei pin rispetta quella mostrata nella pagina precedente.

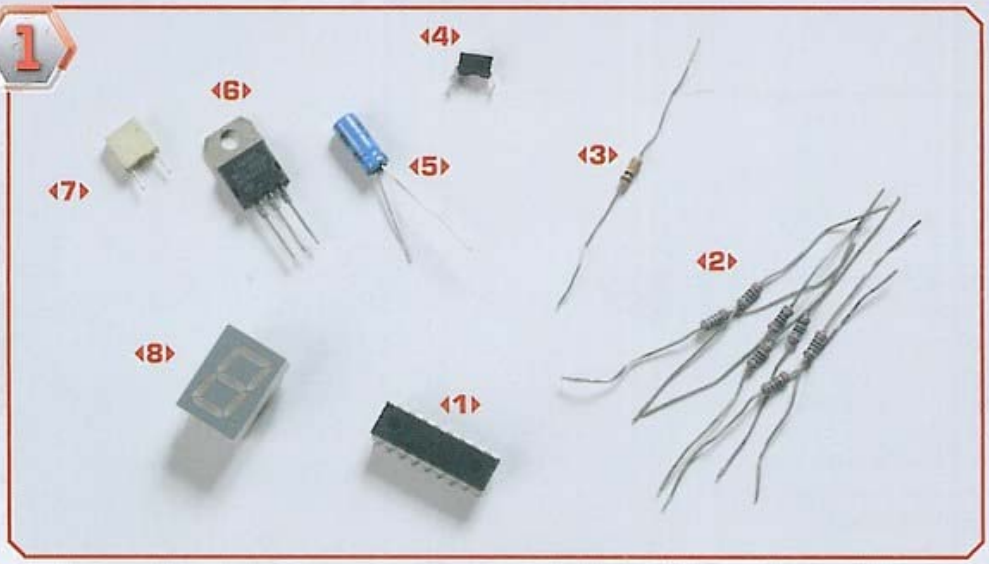
PIN	FUNZIONE
1	E
2	D
3	Catodo Comune (massa)
4	C
5	Punto
6	B
7	A
8	Catodo Comune (massa)
9	F
10	G

STEPbySTEP

L'INPUT DEI PIC E I DISPLAY A 7 SEGMENTI

Testiamo a questo punto il firmware analizzato nelle pagine precedenti. Per proseguire dovrai copiare il codice sorgente mostrato nel fascicolo precedente all'interno di mikroC e compilarlo, utilizzando poi il file '.hex' ottenuto per programmare il PIC 16F628.

1



‹1› **PIC 16F628 programmato con il firmware presentato nel fascicolo precedente (PIC)**

‹2› **7 resistori da 220 ohm (R_L)**

‹3› **un resistore da 10 kohm (R_{IN})**

‹4› **un micropulsante (aperto in stato di riposo) (BUT)**

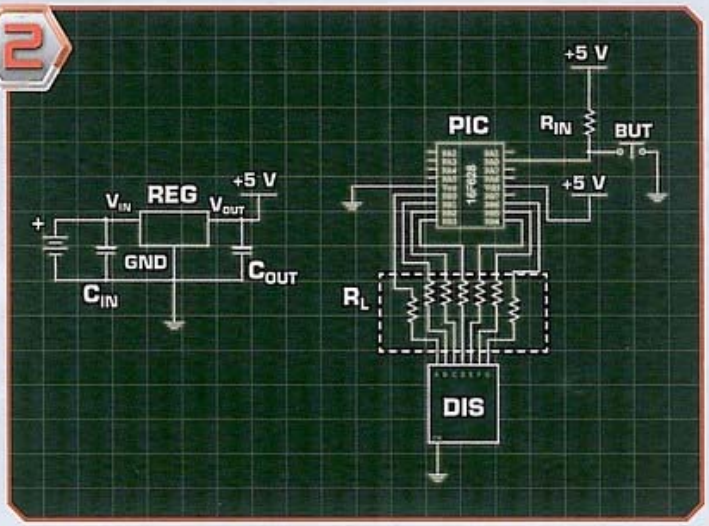
‹5› **un condensatore da 22 µF (C_{OUT})**

‹6› **un regolatore 2940-5.0 (REG)**

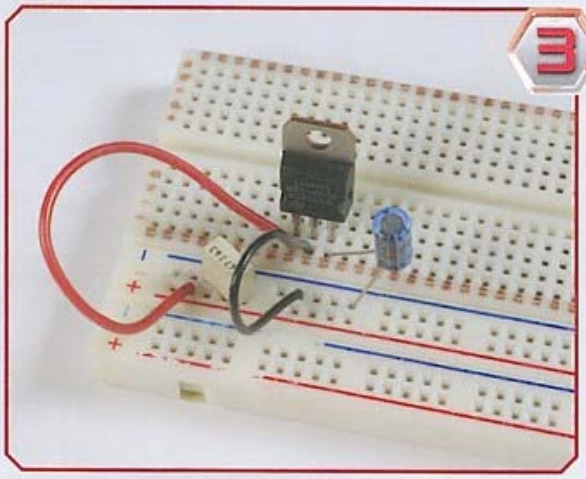
‹7› **un condensatore da 0,47 µF (C_{IN})**

‹8› **un display a 7 segmenti a catodo comune (in questo Workshop è stato impiegato il modello ELS-515-EWA) (DIS)**

2

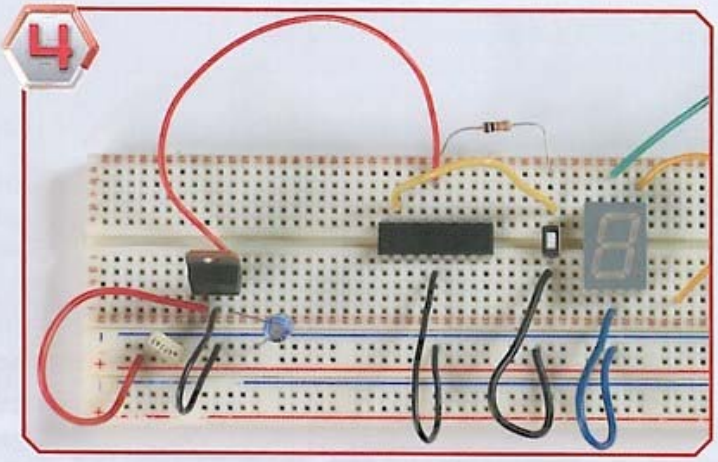


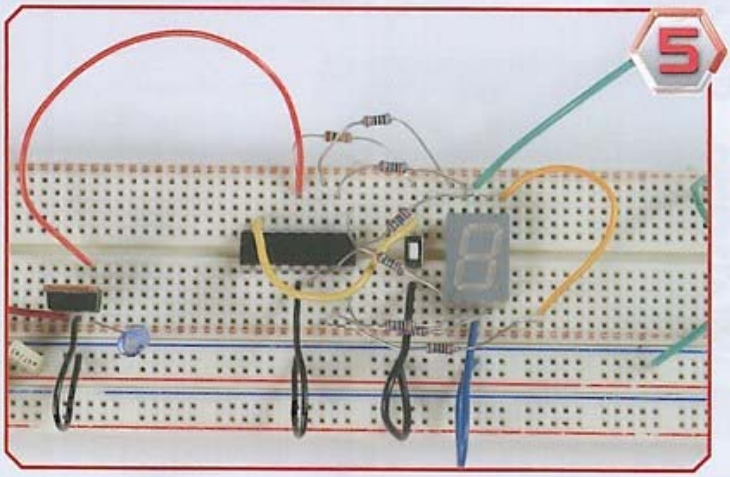
Nello schema a lato puoi osservare il circuito di test. Utilizza il regolatore 2940 per ottenere i 5 V dal pacco batterie, in modo da alimentare il PIC 16F628. Nel montaggio ricordati di verificare la piedinatura del display a 7 segmenti che hai a disposizione, in quanto potrebbe cambiare lievemente da quella mostrata nel FOCUS ON.



3 Inizia il montaggio partendo dal **blocco di alimentazione** (costituito dal **2940-5.0** e dai **due condensatori**), in modo da ottenere il circuito che produrrà i 5 V necessari all'alimentazione del PIC. Inserisci il **16F628** sulla scanalatura centrale della breadboard.

4 Aggiungi al circuito il **display a 7 segmenti** e il **micropulsante** con il **resistore R_{IN} da 10 kohm** posto in ingresso al pin **RA0** del PIC. Ricorda che **il pulsante è un dispositivo elettromeccanico simmetrico**, quindi non è necessario rispettare un verso particolare per il suo inserimento.

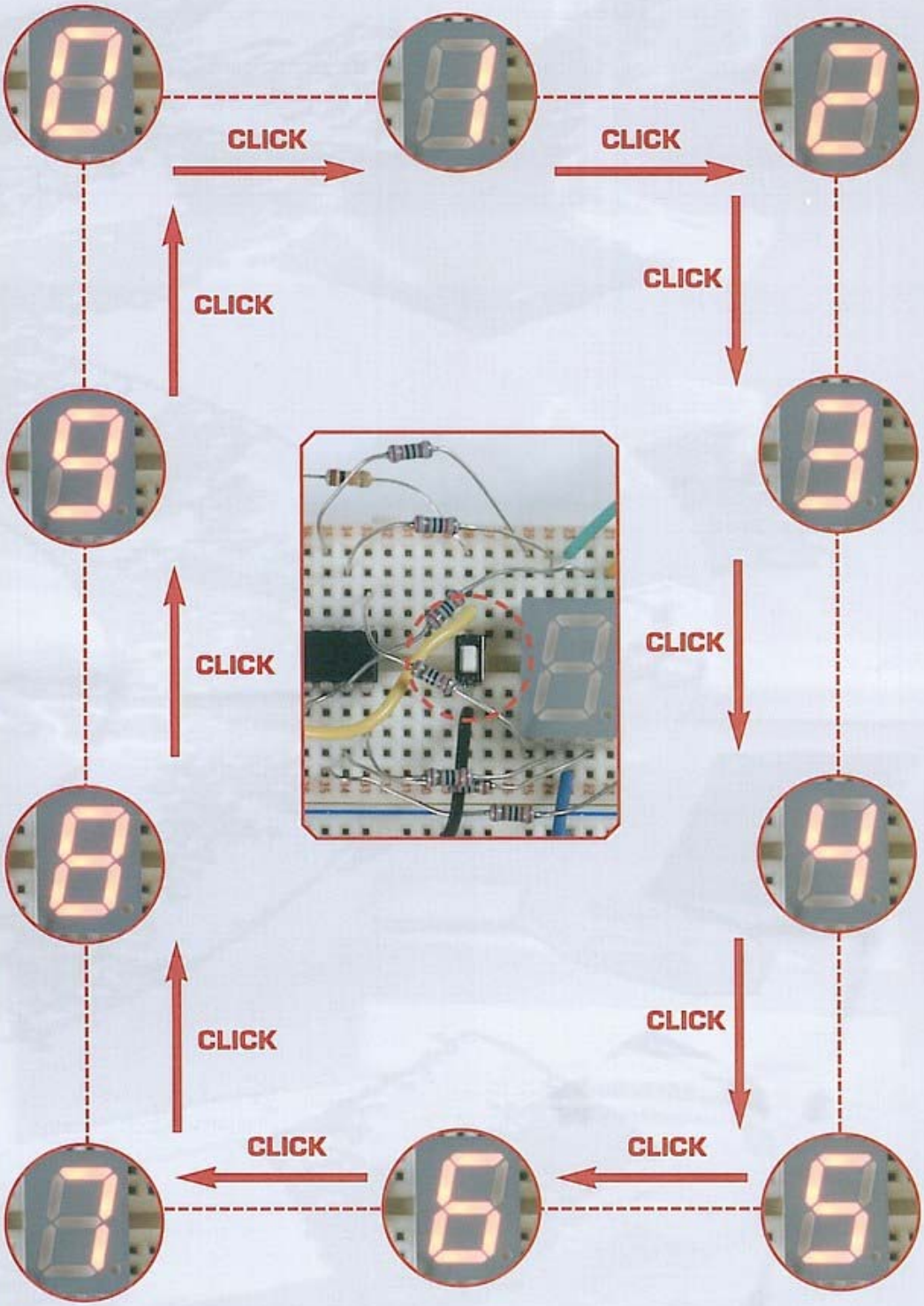




5 Completa infine il circuito, **stabilendo i collegamenti tra il display a 7 segmenti e i pin della porta B del PIC per mezzo dei resistori da 220 ohm.** Se tutto sarà stato eseguito correttamente, all'accensione del circuito **vedrai apparire sul display la cifra '0'.** Ora puoi interagire con il PIC **cliccando ripetutamente il micropulsante.** Come detto nel fascicolo precedente, **a ogni click del bottone la cifra presente sul display verrà incrementata di una unità** passando da 0 a 1, 2 ecc. fino a giungere a 9, per poi far riprendere il conteggio dall'inizio.

6 In questa pagina puoi vedere l'evoluzione ciclica del circuito nei suoi dieci stati di visualizzazione. Questo è il comportamento che dovrai attenderti durante il collaudo del sistema.

STATO INIZIALE



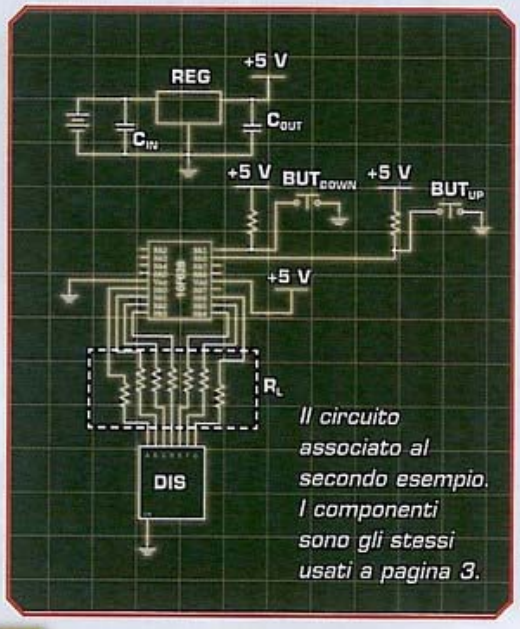
UN INPUT A DUE PULSANTI

Per concludere questo Workshop ti presentiamo una variante del firmware che hai appena avuto la possibilità di sperimentare. In questo nuovo programma è stato inserito il supporto per un secondo pulsante di input (da collegare al pin RA1, come mostrato nello schema della pagina successiva), che ti permetterà di 'decrementare' la variabile di accumulazione 'valore_attuale', rendendo così il conteggio 'bidirezionale'.

Inserire un secondo pulsante per il decremento

```
void main()
{

  /* variabile valore attuale che funge da indice di scansione
  e 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. I valori numerici
     contenuti nel vettore sono quelli presentati nella tabella
     del fascicolo precedente.  */

  int configurazioni_LED [] = {63, 6, 91, 79, 102, 109, 125, 7, 127,
111};

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

  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;
    }


      /* se il pin 1 della porta A (RA0) ha ingresso
        basso significa che è stato  premuto il pulsante */
    if(PORTA.F1 == 0)
    {
        while(PORTA.F1 == 0)    /* finchè il pulsante è premuto */
        {
        }
            /* il pulsante è rilasciato e il valore logico
              di RA1 è tornato alto*/

            /* decremento il valore dell'indice */
            valore_attuale--;

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


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



Il circuito associato al secondo esempio. I componenti sono gli stessi usati a pagina 3.

ACCEDERE AI PIN DEI MICROCONTROLLORI

L'ambiente mikroC permette di accedere in modo dedicato ai singoli pin delle porte dei microcontrollori. Per far ciò è sufficiente far seguire il nome della porta da un punto e dall'identificativo FX, dove X rappresenta il numero del bit desiderato, da 0 a 9. Ad esempio, volendo accedere al pin numero 4 della porta B, sarà sufficiente scrivere PORTB.F4. Quando si accede ai singoli pin, ovviamente, è possibile leggere e scrivere solamente valori binari '0' o '1'.