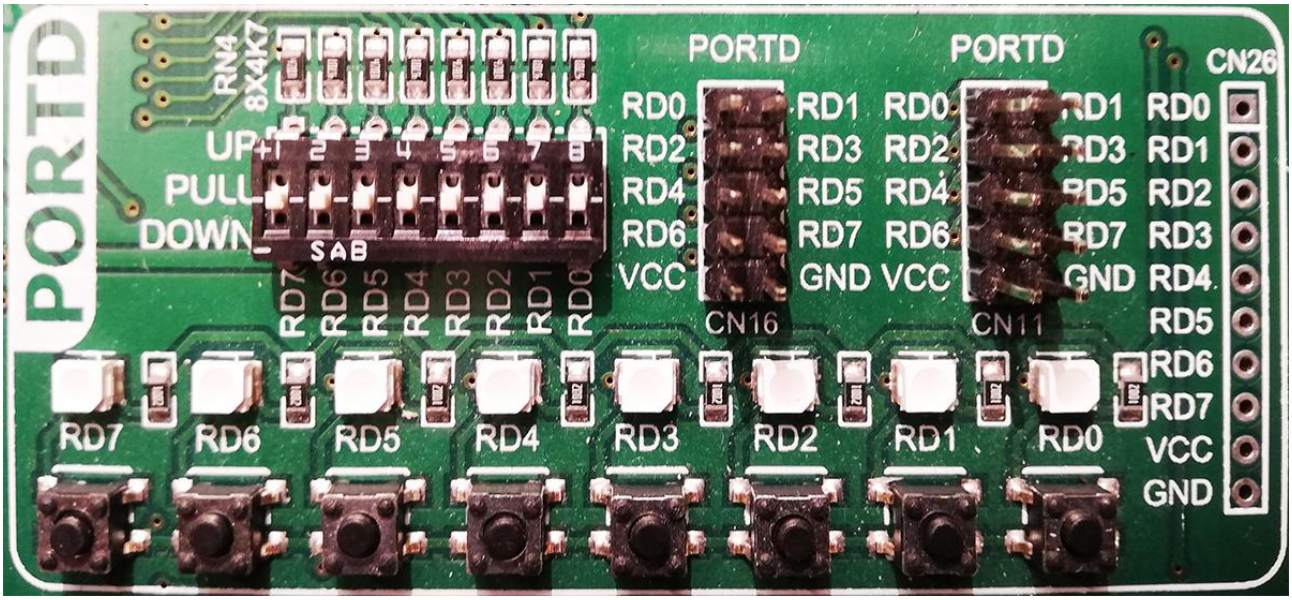
PIC18: leggere un ingresso digitale in assembly



Continuiamo in questa pagina ad esaminare l'uso di PORTx per leggere un segnale digitale proveniente dall'esterno

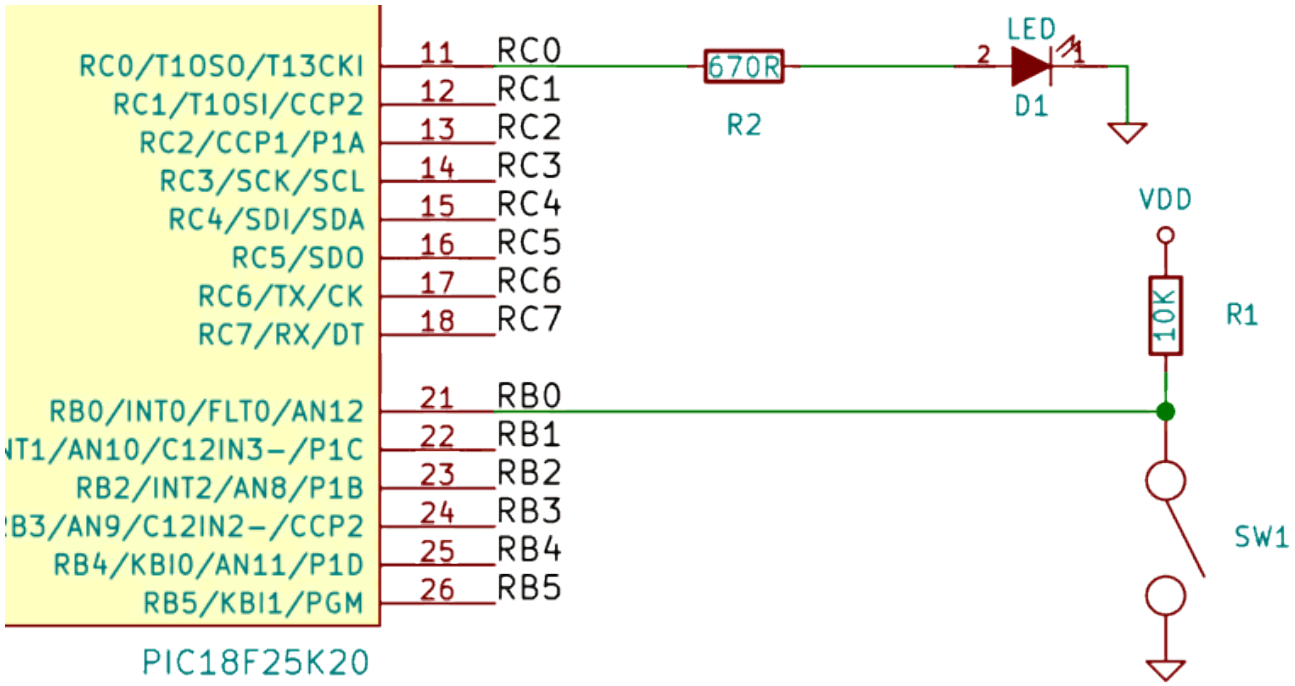
Leggiamo un bit

Per leggere lo stato di un segnale esterno al PIC18 occorre configurare un pin di una porta come ingresso digitale.

Se una porta è configurata come ingresso digitale:

- una tensione di circa 0 V viene letta come valore 0 logico
- una tensione pari a circa VDD viene letta come 1 logico
- tensioni intermedie vengono lette come 1 oppure 0, in modo piuttosto casuale, ovviamente in funzione di quale delle due precedenti situazioni è *più vicina*. Per un approfondimento, cercate sul foglio tecnico VIH oppure VIL, valori non coincidenti per tutti i pin. Analogamente producono risultati casuali i pin non collegati a nulla
- Infine, tensioni superiori a VDD oppure inferiori a GND possono essere distruttive per il PIC18

Realizziamo il circuito mostrato, collegando un interruttore al pin RB0 (che quindi andrà configurato come ingresso digitale) ed un LED al pin RC0 (che quindi andrà configurato come uscita digitale):



- Quando l'interruttore SW1 è chiuso, la tensione di ingresso a RB0 vale 0 V e quindi l'ingresso verrà letto come zero logico: il pulsante è infatti un cortocircuito verso massa.
- La resistenza R1 è la *resistenza di pull-up*: "tira su" la tensione di ingresso quando l'interruttore è aperto. In questo caso la tensione di ingresso è (circa) VDD e quindi verrà letto come uno logico. Si noti che, in assenza di tale resistenza il valore letto sarebbe casuale. Il valore della resistenza di pull-up è poco rilevante, in genere si sceglie di poche decine di KΩ.

Il programma di esempio ha la seguente struttura:

- Configura il pin RB0 come ingresso, mettendo a 1 il bit 0 del registro speciale TRISB.
  `bsf TRISB, 0 ;meglio scrivere "bsf TRISB, RB0`
- Configura il pin RC0 come uscita digitale, mettendo a 0 il bit 0 del registro TRISC. Nell'esempio in realtà tutti i pin di PORTC sono stati configurati come uscite digitali:
  `clrf TRISC`

- All'interno di un loop infinito, legge il valore del pin RB0 e, in base al suo valore, accende o spegne il LED. Per conoscere il valore dell'ingresso RB0, è necessario leggere il bit 0 del registro PORTB
  `btfss PORTB, 0 ; per motivi di leggibilità è forse meglio scrivere: btfss PORTB, RB0`

Pull-up interno

L'uso di pulsanti ed interruttori è molto frequente. Per questo i progettisti del PIC18 hanno deciso di includere in alcune porte la resistenza di pull-up (*weak internal pull-up*). In genere sono previste almeno per PORTB.

Queste resistenze sono in genere disattivate. L'attivazione deve avvenire in due passaggi, descritti nel Datasheet:

- Occorre, per ciascun pin, attivare individualmente la resistenza di pull-up, agendo sul registro `WPUB`:
  ```
  bsf WPUB, 0 ; attiva le resistenza di pull'up per RB0
  ```

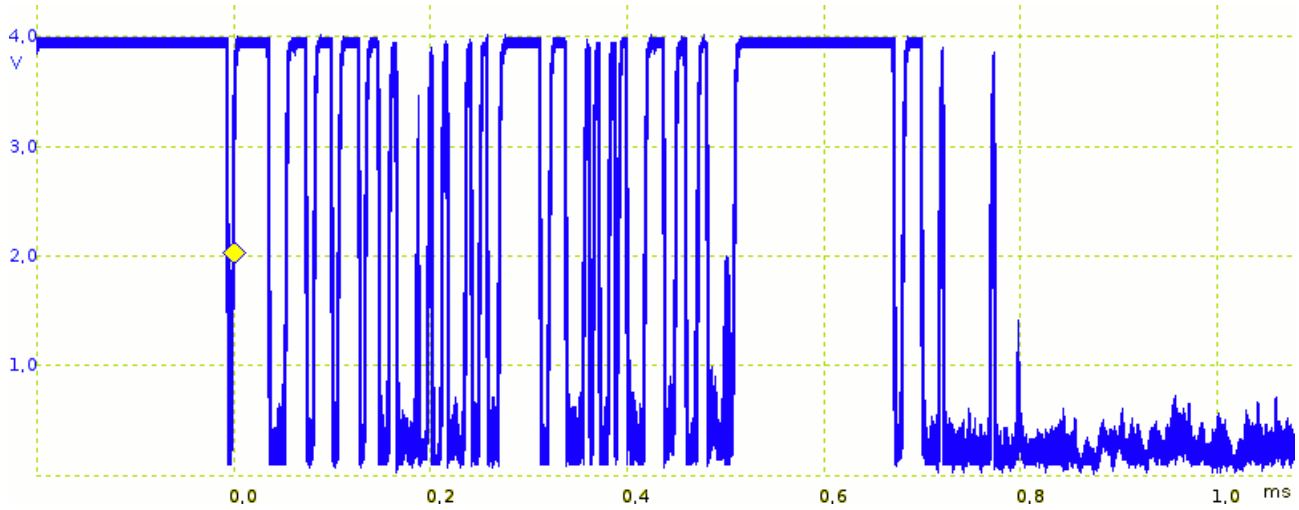
- Occorre abilitare globalmente tutte le resistenze di pull-up attivare, agento sul bit 7 del registro INTCON2
  ```
  bcf INTCON2, 7
  ```

Se viene attivata la resistenza interna di pull-up, non è più necessario utilizzare la resistenza esterna (R1 nello schema precedente)

Antirimbalzo

Uno dei fenomeni che si accompagna sempre alla presenza di interruttori meccanici è il *rimbalzo*, causato dallo scontro tra i due contatti metallici; ciò produce un andamento della tensione tutt'altro che ideale. In genere il fenomeno è presente solo (o quasi) in chiusura, mentre in apertura, almeno a bassa tensione, non si presenta in modo significativo.

L'immagine reale seguente mostra come un rimbalzo si presenta a livello elettrico: una serie di passaggi da 1 a 0 e viceversa, di durata breve a livello umano, meno di 1 ms in molti casi. Il numero di eventi è tendenzialmente casuale, spesso superiore a 10 o 20.



I rimbalzi non sono un problema se siamo interessati a sapere se un interruttore è aperto o chiuso. Diventano un problema se siamo interessati a contare quante volte un pulsante è stato premuto... Pensate ad una tastiera!

Tra i metodi antirimbalzo (*debounce*) possiamo proporre:

- Usare pulsanti migliori: aiuta, ma assolutamente non risolve
- Usare un deviatore (SPDT) ed un latch RS: è la soluzione ideale, ma costa, occupa spazio e richiede un deviatore. Questo metodo lo trovate descritto in tutti i libri di elettronica digitale, appena vengono introdotti i flip-flop RS
- Usare un gruppo RC: funziona, ma costa e occupa spazio
- Usare il software, la soluzione qui descritta

L'idea è quella di effettuare due letture del valore del pin inserendo un ritardo (1 o 2 ms) tra di esse. In questo modo eventuali rimbalzi vengono ignorati.