Připojení tlačítka na vstup řídícího počítače

Interní Pull-Up] **FIGURE 4-4**.

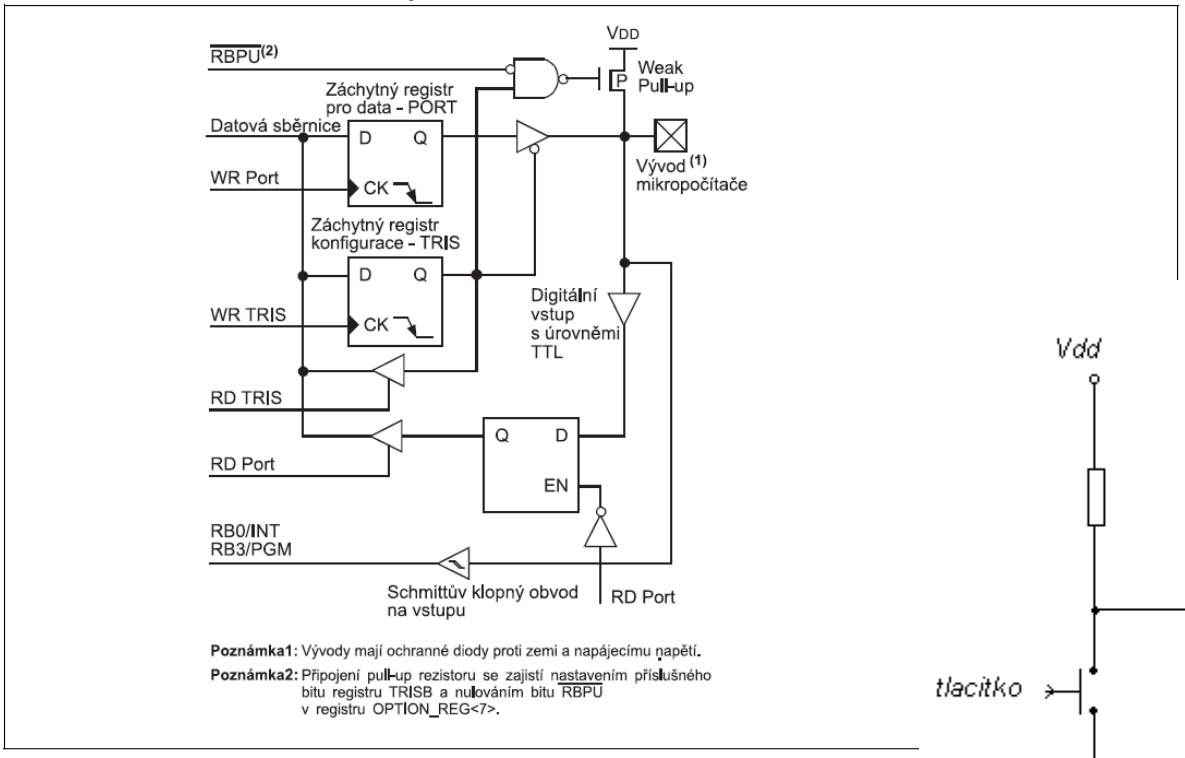
SW obsluha (bezzákmitové tlačítko). Možno ukázat v programu. [tl.zip]

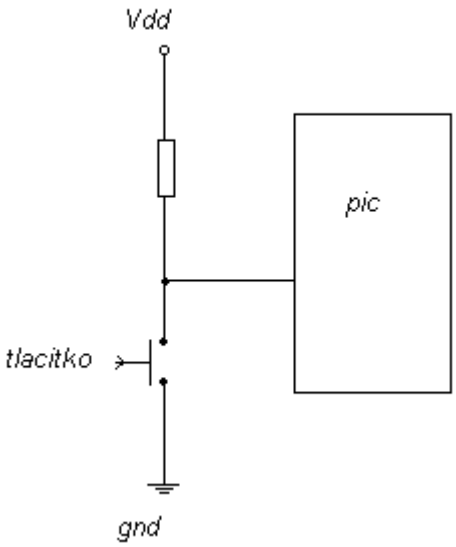
Když zbude čas, tak pokračovat otázkou č. .

Interní Pull-Up:

Každý vývod brány PortB je vybaven možností připojit ve vstupním režimu vnitřní "pull-up" rezistor na vstup. Tím dojde k přidržení vstupního nezatíženého vývodu ve stuvu log.1. Nulováním řídícího bitu $\overline{\text{RBPU}}$ v registru OPTION_REG<7> lze připojit na všechny vstupní vývody brány PortB rezistory pull-up. Přepnutím do výstupního režimu dojde automaticky k odpojení vnitřního pull-up rezistoru. Po signálu reset jsou všechny pull-up rezistory zakázány.

Obrázek 4-4: Blokové schéma vývodů RB3-RB0





Externi pull-up:

Funkce pull-upu:

hradlo na vstupu pinu reaguje na všechna indukovaná napětí v jeho okolí, takže nejen na rušení okolních spotřebičů, ale i vlastní činnost procesoru. Obvykle se to řeší Pull-Up nebo Pull-down rezistorem (záleží na tom, jestli tlačítko připojuje na zem nebo na zdroj). Jeho velikost je obvykle v kiloohmech (10 až 100 k - záleží na aplikaci, popř. bývá uvedeno v datasheetu - nesmí zatžovat obvody pinu) a slouží k tomu, aby zabezpečil přívod žádané hodnoty napětí pro žádanou logickou úroveň v čase, kdy tlačítko není stisknuté. Stiskem tlačítka se napěťové poměry na pinu změní, protože odpor tlačítka je nulový.

Program:

```
/*** Bezzakmitove tlacitko ***/

#include "tl.h"

#define LED1   PIN_A4   // Ledky
#define LED2   PIN_A5
#define TL     PIN_B0   // Tlacitko
#define JITTER 25       // Doba na ustaleni kontaktu tlacitka [ms]

void main()
{
  int1 n;     // Promenna na pamatovani predchoziho stavu

  port_b_pullups(TRUE);
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
```

```
//   setup_psp(PSP_DISABLED);  // PIC16F876A tuhle periferii nema
   setup_spi(SPI_SS_DISABLED);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);

   n=TRUE;     // Inicializace promenne

   while(TRUE)
   {
     while(input(TL));       // Cekame na stisk tlacitka

     n=~n;                  // Zmenime stav
     output_bit(LED1,n);     // Rozsvitime/zhasneme LEDky
     output_bit(LED2,~n);

     delay_ms(JITTER);       // Cekani na ustaleni kontaktu
     while(!input(TL));      // Cekame na pusteni tlacitka
     delay_ms(JITTER);       // Cekani na ustaleni kontaktu
    }}
```

Jelikoz tlačitko není idealni tak při stisknuti zakmitava a tzn ze při jednom stisku tlacit sepne rozepne sepne a rozepne dokud se neustálí v sepnutem stavu. Takto muze tlačítko v velmi krátkých programech nechtene několikrát za sebou sepnout a spustit tak program vícekrát. Proto se jak vidite v programu pouzije zpozdení aby bylo před další smyckou tlačitko ustalene v logicke 1 nebo 0.

Další priklad SW odstraneni je:

```
while(input(TL));       // Cekame na stisk tlačítka
 {
delay_ms(JITTER);       // Cekani na ustaleni kontaktu
while(input(TL));       // kontrolujeme jestli ma po ustaleni porad sepnuty stav
 }
```

Pokud se vam to zda kratky mate smulu protože toho vic neni a musite zacit mluvit o otazce č.3 jak uz je v zadaní napsano.