

Přesun bitu

Př.: přesune 0. bit portu A na 5. bit portu B:

```
btfsc  PORTA,0      ; IF zdrojový bit = 0 THEN přeskok  
bsf     PORTB,5      ; nastavení cílového bitu  
btfss  PORTA,0      ; IF zdrojový bit = 1 THEN přeskok  
bcf     PORTB,5      ; nulování cílového bitu
```

Záměnou instrukcí **bsf** a **bcf** se provede přesun
invertovaného bitu

Test skupiny bitů

Nejprve logickým součinem vymaskujeme nepotřebné bity a pak porovnáme s požadovanou hodnotou

```
MASK    EQU    b'00110111'
VALUE   EQU    b'00100101'

...
movf     reg,W      ; test hodnoty 0x25 v registru „reg“
andlw    MASK       ; log. součin s maskou – bin. 1 vybírají bit
xorlw    VALUE      ; nonekvivalence dá 0 na pozice shodných bitů,
                    ; při shodě všech bitů se nastaví příznak ZERO
btfsc    STATUS,Z; větvení programu
goto     ...
...
```

Maska i hodnota mohou být i proměnné (instrukce *lw nahradíme příslušnými ekvivalenty pro práci s registry *wf)

Test změny skupiny bitů

Testovaná skupina bitů je v registru reg

```
    movf    reg,W
    andlw   MASK
    movwf   oldstat    ; uložení výchozího stavu
    ...
Test: movf   reg,W      ; testovaná proměnná -> W
    andlw   MASK        ; maskování irelevantních bitů
    movwf   temp        ; záloha aktuálního stavu
    xorwf   oldstat,W    ; porovnání s minulým stavem
    btfsc   STATUS,Z     ; přeskok při neshodě stavů (Z = 0)
    goto    Test
    movf    temp,W
    movwf   oldstat      ; aktuální stav zapamatován
    ...
```

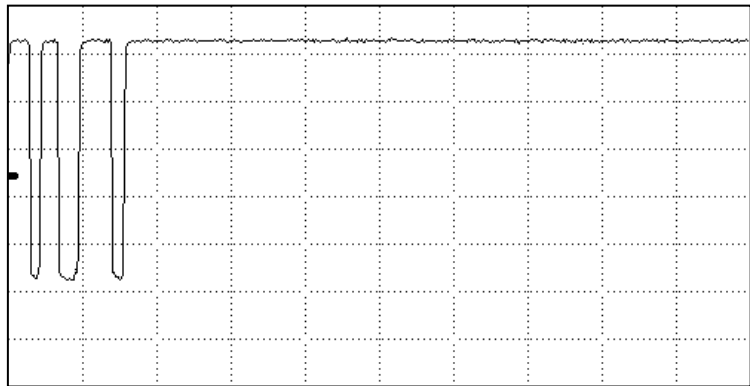
- pokud je výchozí stav znám, je možno `oldstat` nastavit přímo

Ošetření tlačítka

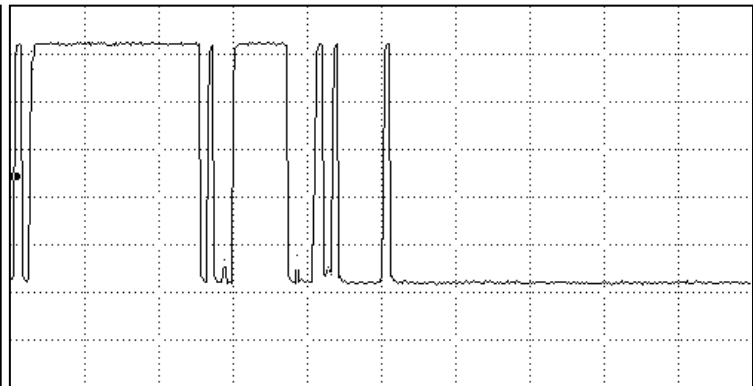
Mechanické kontakty při sepnutí i rozepnutí po dobu desetin až desítek ms generují parazitní impulsy

⇒ nutno ošetřit (vícekrát opakované čtení na stejný stav nebo test shody dvou čtení po dostatečné prodlevě)

sepnutí (100 $\mu\text{s}/\text{div}$)



rozepnutí (100 $\mu\text{s}/\text{div}$)



Vícekrát opakované testování

V registru **citac** je počet opakování NSCAN, vzhledem k rychlé smyčce např. 100

```
Scan:  movlw  NSCAN    ; počet testů do čítače
        movwf  citac
        movf   PORTA,W ; čtení portu s připojenými kontakty
        andlw  MASK    ; vymaskování nevýznamných bitů
        movwf  temp    ; uložení aktuálního stavu
Scan1:  movf   PORTA,W ; čtení portu s připojenými kontakty
        andlw  MASK
        xorwf  temp,W  ; porovnání s uloženým stavem
        btfss  STATUS,Z; přeskok při shodě stavů
        goto   Scan    ; začít znovu s plným počtem opakování
        decfsz citac,F ; dekrementace čítače a test na 0
        goto   Scan1   ; opakování testu
        ...           ; pokračování po splnění podmínek
```

Toto je příklad na testování skupiny tlačítek – pro jedno stačí např. instrukce **btfss**.

Test shody dvou čtení s prodlevou

Předpokládá podprogram Delay_ms

```
Scan: movf    PORTA,W    ; 1.čtení portu s připojeným tlačítkem
      andlw   MASK       ; vymaskování nepodstatných bitů
      movwf   temp       ; uložení aktuálního stavu
      movlw   SCANDELAY
      call    Delay_ms   ; prodleva
      movf    PORTA,W    ; 2.čtení portu s připojeným tlačítkem
      andlw   MASK
      xorwf   temp,W     ; porovnání s uloženým stavem
      btfss   STATUS,Z   ; přeskok při shodě stavů
      goto    Scan       ; skok na opakování testu
      ...               ; pokračování po splnění podmínky
```

Toto je příklad na testování skupiny tlačítek – pro jedno stačí např. instrukce **btfss**.

Výběr z tabulky konstant

Do tabulky je výhodné ukládat různé kódy (např. sedmsegmentové), konstanty zpoždění (např. generování kmitočtu), převodní charakteristiky (např. sinus), atd.

Řešení např. pomocí podprogramu a speciální instrukce `retlw`

; v hlavním programu

```
...  
movf    index,W    ; načte index nebo zde může být konst.  
call    Table      ; vyzvednutí položky do W  
...
```

```
Table: brw          ; PC = PC + W + 1  
    retlw  k0        ; sem skočí při W=0  
    retlw  k1        ; sem skočí při W=1  
    ...  
    retlw  kn
```

Operace s čísly

Porovnání dvou 8bitových čísel

```
movf    x1,W      ; první číslo -> W,  (x1, x2 jsou registry)
subwf   x2,W      ; odečtení x2-x1 (uložená čísla nedotčena)
btfs*   STATUS,Z  ; test na (ne)shodu podle Z (x1=x2 => Z=1)
goto    ...
btfs*   STATUS,C  ; test větší/menší (C=1 pouze při x1<x2)
goto    ...
```

Inkrementace 16bitového čísla (nmsb:nlb)

```
incfsz  nlb,F     ; inkrementace, přeskok při 0
goto    $+2       ; přeskok následující instrukce
incf    nmsb,F    ; inkrementace při přetečení nlb
...
```


Součet dvou 16bitových čísel

V registrech ah:al je jedno číslo, v bh:bl druhé číslo, výsledek uloží do bh:bl

```
movf    al,W      ; LoByte prvního čísla -> W
addwf   bl,F      ; přičtení k LoByte 2. č. (příp. nastavení C)
movf    ah,W      ; HiByte prvního čísla -> W
addwfc  bh,F      ; sečtení a uložení HiByte 2. čísla s W a C
```

- u rozdílu se zamění **addwf** za **subwf** a **addwfc** za **subwfb**
- u vícebytových čísel úsek programu analogicky pokračuje