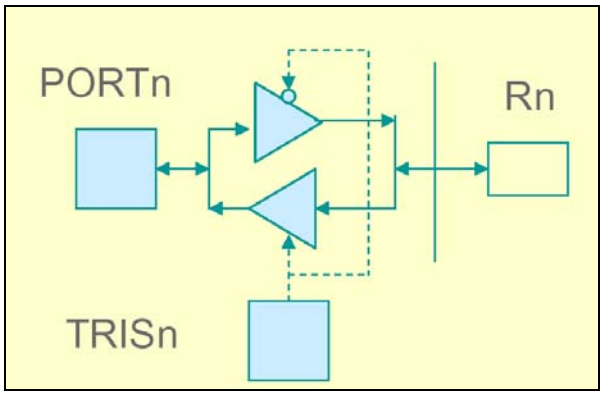
4. Osnovne vhodno/izhodne enote

Spoznali boste osnovne vhodno/izhodne enote – digitalne (binarne) ali logične vhode/izhode. Naučili se boste konfigurirati posamezne priključke vrat A, B in C ter programsko krmiliti izhode in ugotavljati stanja vhodov.

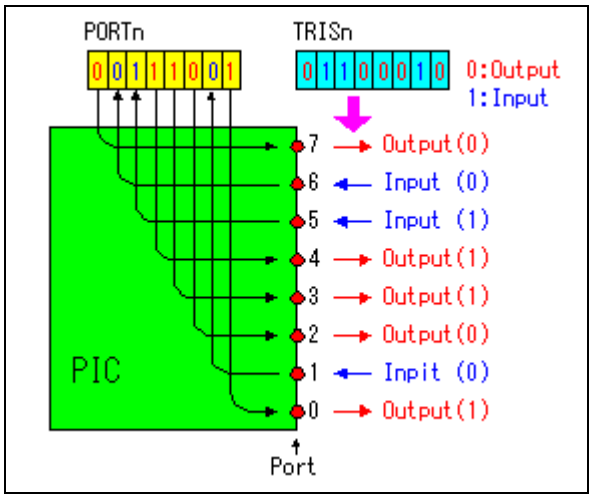
Mikrokrmilnik PIC16F876 (Slika 1-5) ima tri vhodno/izhodna vrata (angl.: port): **PORTA**, **PORTB** in **PORTC**, preko katerih so dostopni vhodno/izhodni signali vsebovanih vmesnikov. Vrednosti na priključkih lahko zajemamo oz. spreminjamo s čitanjem/vpisovanjem vrednosti istoimenskih registrov. Priključke je potrebno predhodno programsko konfigurirati (izbrati vhod/izhod) z nastavitvijo vrednosti v registrih **TRISA**, **TRISB** in **TRISC**. Ta postopek imenujemo tudi »inicializacija«.

4.1. Vrata PORTA, PORTB in PORTC

Osnovni način delovanja (Slika 4-1) **PIC mikrokrmilnikov** je, da se linije (priključki) vrat **PORTn**: **PORTA** (RA0 do RA5), **PORTB** (RB0 do RB7) in **PORTC** (RC0 do RC7) **uporabljajo kot digitalni (binarni, logični, preklopni) vhodi ali izhodi**. Določanje orientacije (Slika 4-2) ali smeri delovanja (**0: izhod** - Output, **1: vhod** - Input) poteka z vpisom bitne informacije v ustrezni register **TRISn**, stanje vhodne/izhodne linije pa določamo s čitanjem/vpisom v bitne celice pripadajočega registra **PORTn**.



Slika 4-1: Poenostavljena zgradba elementa vrat



Slika 4-2: Primer konfiguriranja priključkov vrat

Upoštevati je potrebno [8], da je **večina vhodno/izhodnih priključkov multipleksirana** (deljena raba) s preostalimi funkcijami mikrokrmilnika (analogni vhodi, Timer0,1,2, UART, idr.).

4.1.1. PORTA

PORTA ima 6 vhodno/izhodnih priključkov, katerih obnašanje (logični vhod ali izhod) nastavljamo z registrom **TRISA**. Postavitev ustreznega **bita TRISA na 1** bo iz priključka na **PORTA naredila vhod**, **postavitev na 0 pa izhod**. Priključek **RA4** je multipleksiran z vhodom časovnika Timer0. **Ostali priključki PORTA** so **multipleksirani z analognimi vhodi** oziroma z analognima referenčnima vhodoma VREF. Delovanje priključkov je določeno z nastavitvijo/brisanjem ustreznih bitov v registru **ADCON1**. Register **TRISA** določa smer delovanja RA priključkov tudi, ko so le-ti konfigurirani kot analogni vhodi, torej je treba zagotoviti, da so priključki tudi v tem primeru orientirani kot vhodi (ustrezni biti v **TRISA** so postavljeni na 1).

Ob vklopu se priključki konfigurirajo kot analogni vhodi in imajo vrednost 0.

Funkcije PORTA:

Ime	Bit#	Buffer	Funkcija
RA0/AN0	bit0	TTL	Vhod/izhod ali analogni vhod
RA1/AN1	bit1	TTL	Vhod/izhod ali analogni vhod
RA2/AN2	bit2	TTL	Vhod/izhod ali analogni vhod
RA3/AN3/VREF	bit3	TTL	Vhod/izhod ali analogni vhod ali VREF
RA4/TOCKI	bit4	ST	Vhod/izhod ali vhod za zunanji takt za Timer0 Izhod je z odprtim ponorom
RA5/$\overline{SS}$/AN4	bit5	TTL	Vhod/izhod ali analogni vhod

Legenda: TTL = TTL vhod, ST = vhod s Schmittovim Triggerjem

S **PORTA** povezani registri:

Naslov	Ime	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Vrednost ob POR, BOR	Vrednost ob drugih resetih
05h	**PORTA**	-	-	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	**TRISA**	-	-	**PORTA** register smeri podatkov						--11 1111	--11 1111
9Fh	**ADCON1**	ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000
1Fh	**ADCON0**	ADCS1	ADCS0	CHS2	CHS1	CHS0	Go/Done	-	ADON	000 00-0	0000 00-0

Legenda: x = neznan, u = nespremenjen, - = neuporabljene lokacije, beri kot '0', PORTA ne uporablja osenčenih lokacij

4.1.2. PORTB

PORTB ima 8 vhodno/izhodnih priključkov, katerih obnašanje (logični vhod ali izhod) nastavljamo z registrom **TRISB**. Postavitev ustreznega bita **TRISB** na 1 bo iz priključka na **PORTB** naredila vhod, postavitev na **0** pa **izhod**. Trije biti so multipleksirani s funkcijo za programiranje pri nizki napetosti (**RB3/PGM)** in signaloma za komunikacijo z **ICD** modulom: **RB6/PGC** in **RB7/PGD**). Priključek **RB0** je multipleksiran (podvojen) z zunanjim prekinitvenim vhodom **INT** (Interrupt).

Funkcije PORTB:

Ime	Bit#	Buffer	Funkcija
RB0/INT	bit0	TTL/ST	Vhod/izhod ali zunanji priključek za »interrupt«
RB1	bit1	TTL	Vhod/izhod
RB2	bit2	TTL	Vhod/izhod
RB3/PGM	bit3	TTL	Vhod/izhod ali priključek za programiranje pri nizki napetosti
RB4	bit4	TTL	Vhod/izhod
RB5	bit5	TTL	Vhod/izhod
RB6/PGC	bit6	TTL/ST	Vhod/izhod ali priključek za programiranje pri nizki napetosti
RB7/PGD	bit7	TTL/ST	Vhod/izhod ali priključek za programiranje pri nizki napetosti

Legenda: TTL = TTL vhod, ST = vhod s Schmittovim Triggerjem

S **PORTB** povezani registri:

Naslov	Ime	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Vrednost ob POR, BOR	Vrednost ob drugih resetih
06h, 106h	**PORTB**	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	**TRISB**	PORTB register smeri podatkov								1111 1111	1111 1111
81h, 181h	**OPTION_REG**	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legenda: x = neznan, u = nespremenjen, - = neuporabljene lokacije, beri kot '0', PORTB ne uporablja osenčenih lokacij

4.1.3. PORTC

PORTC ima 8 vhodno/izhodnih priključkov, katerih obnašanje (logični vhod ali izhod) nastavljamo z registrom **TRISC**. Postavitev ustreznega bita **TRISC** na **1** bo iz priključka na **PORTC** naredila **vhod**, postavitev na **0** pa **izhod**.

Funkcije PORTC:

Ime	Bit#	Buffer	Funkcija
RC0/T1OSO/T1CKI	bit0	ST	Vhod/izhod ali izhod oscilatorja za Timer1 / vhod takta za Timer1
RC1/T1OSI/CCP2	bit1	ST	Vhod/izhod ali vhod oscilatorja za Timer1 ali vhod Capture2 / izhod Compare 2 / izhod PWM2
RC2/CCP1	bit2	ST	Vhod/izhod ali vhod Capture1 / izhod Compare 1 / izhod PWM1
RC3/SCK/SCL	bit3	ST	Vhod/izhod ali sinhronski serijski CLOCK za SPI in I^2C
RC4/SDI/SDA	bit4	ST	Vhod/izhod ali podatkovni vhod SPI / I^2C
RC5/SDO	bit5	ST	Vhod/izhod ali sinhronski serijski podatkovni izhod
RC6/TX/CK	bit6	ST	Vhod/izhod ali USART asinhronski Transmit ali sinhronski clock
RC7/RX/DtT	bit7	ST	Vhod/izhod ali USART asinhronski Receive ali sinhronski podatki

Legenda: TTL = TTL vhod, ST = vhod s Schmittovim Triggerjem

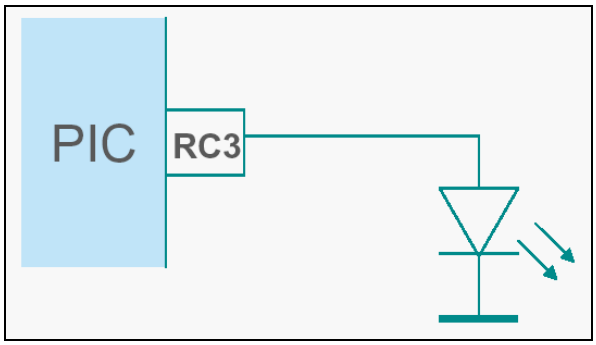
S **PORTC** povezani registri:

Naslov	Ime	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Vrednost ob POR, BOR	Vrednost ob drugih resetih
07h	**PORTC**	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	**TRISC**	PORTC register smeri podatkov								1111 1111	1111 1111

Legenda: x = neznan, u = nespremenjen

4.2. Preprosti zgledi programov za binarne digitalne vhode/izhode

Vzemimo **zgled: krmiljenje svetlobnega indikatorja** (LED dioda), ki je priključen na logični izhod **RC3** (Slika 4-3, dejansko je potrebno dodati zaporedni upor npr.: 330 Ω za omejitev toka na pribl. 10 mA) [3].



Slika 4-3: Priključitev LED diode na izhod RC3

Poglejmo si izsek programa (Program 4-1), ki opravi inicializacijo (**RC3** kot izhod) in zatem vklopi (**RC3** postavi na 1) oziroma izklopi LED diodo (**RC3** postavi na 0):

```
; Inicializacija: vrata PORTC (RC3 izhod) in postavitev zacetnega stanja na RC3 izhod
       movlw  b'10011111'    ;0x9f->W, priprava maskirne konstante za brisanje bitov:6,5
       andwf  STATUS,f       ;bitni AND, 0->RP1,RP0, izbira Bank0 zaradi dostopa do PORTA
       bcf    PORTC,3        ;zacetno stanje RC3=0, ker stanja ob RESET-u niso definirana!
       bsf    STATUS,RP0     ;1->RP0, kodna stran Bank1 zaradi dostopa do TRISC registra
       bcf    TRISC,3        ;0->PORTC.3, RC3: izhod
; Postavitev stanja na vrata PORTC, bit 3
       bcf    STATUS,RP0     ;kodna stran Bank0 zaradi dostopa do PORTC registra!
ponovi bsf    PORTC,3        ;1->RC3, vklopi LED
; Brisanje stanja na izhodu RC3
       bcf    PORTC,3        ;0->RC3, izklopi LED
       goto   ponovi         ;neskoncna zanka
```

Program 4-1: Vklop/izklop LED diode v zbirnem jeziku

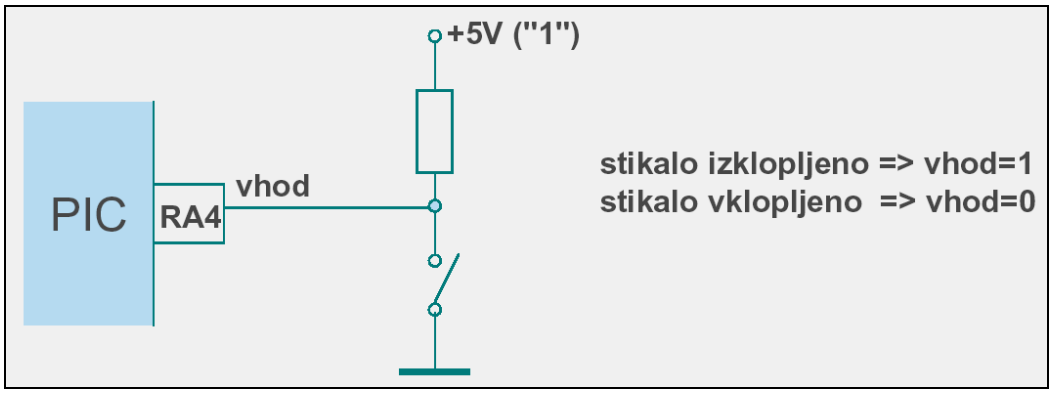
V zgornjem primeru **prvi trije ukazi niso nujno potrebni**, koristni pa so v primeru, ko stanja izhodov (odvisno od naprav, ki so nanje priključene) niso definirana ob prvem vklopu mikrokrmilnika.

Za ilustracijo si še poglejmo, kako je lahko kodiran **enak algoritem v C-jeziku** (Program 4-2) za prevajalnik HI-TEC PICC:

```
main()
{
// Inicializacija: vrata PORTC (RC3 izhod) in postavitev zacetnega stanja na RC3 izhod
PORTC=PORTC&0xf7;     //bitni AND med PORTC in '11110111', zacetno stanje RC3=0,
//ker stanja ob RESET-u niso definirana!
TRISC=0xF7;           //'11110111' -> TRISC, RC3 je izhod
       while(1)       //neskoncna zanka
       {              //zacetek zanke
// Postavitev stanja na vrata PORTC, bit 3
       RC3=1; //1->RC3, vklopi LED
// Brisanje stanja na izhodu RC3
       RC3=0;         //0->RC3, izklopi LED
       }              //konec zanke
}
```

Program 4-2: Vklop/izklop LED diode v C-jeziku

Vzemimo **zgled: program za ugotavljanje stanja stikala**, ki je priključeno na vhod **RA4** (Slika 4-4).



Slika 4-4: Priključitev stikala ali tipke na vhod RA4

```
Stikalo equ   0xF0          ;spremenljivka Stikalo v GPR registru na naslovu 0xF0
; Inicializacija: vrata PORTA (RA4 vhod)
       bsf    STATUS,RP0    ;1->RP0, kodna stran Bank1 zaradi dostopa do TRISA registra
       bsf    TRISA,4       ;1->PORTA.4, RA4: vhod
       bcf    STATUS,RP0    ;kodna stran Bank0 zaradi dostopa do PORTA registra!
; Ugotavljanje stanja PORTA, bit 4, rezultat v registru W
ponovi movf   PORTA,w       ;PORTA->W
       andlw  b'00010000'   ;bitni AND med W in 0x10, maskiranje bita 4, W=000x0000, x=st.
; ce je W=0, je stikalo sklenjeno!, ce je W razlicen od 0, je stikalo odprto!
       movwf  Stikalo       ;W->Stikalo, shrani stanje v spre. Stikalo
       goto   ponovi        ;neskoncna zanka
```

Program 4-3: Ugotavljanje stanja stikala v zbirnem jeziku

V algoritmu za ugotavljanje stanja stikala je potrebno upoštevati, da se **stikalo** smatra kot **idealno**. Dejansko prihaja pri mehanskih stikalih do **pojava »odskakovanja«**, zato je potrebno ustrezno ukrepati (angl.: debounce) [6], bodisi v vezju, bodisi v programskem algoritmu. Jedro zanke **`ponovi`** traja (Program 4-3) le okrog 1 μs, pojav odskakovanja (spreminjanje logičnega stanja med '0' in '1') pa od nekaj ms do nekaj 10 ms.

Za ilustracijo si še poglejmo, kako je lahko kodiran **enak algoritem v C-jeziku** (Program 4-4) za prevajalnik HI-TEC PICC:

```
main()
{
unsigned char Stikalo; //spremenljivka Stikalo je 8-bitno celo stevilo brez predznaka
// Inicializacija PORTA, RA4 kot vhod
TRISA=TRISA|0x10;     // 1 -> bit4, RA4 je vhod
       while(1)       //neskoncna zanka
       {      //zacetek zanke
// Ugotavljanje stanja PORTA, bit 4, rezultat v spremenljivki Stikalo
       Stikalo=RA4;  //RA4->Stikalo
// ce je Stikalo=0 je stikalo sklenjeno!, ce je Stikalo razlicen od 0, je stikalo odprto!
       }              //konec zanke
}
```

Program 4-4: Ugotavljanje stanja stikala v C- jeziku

Obsežnejši primer (Program 4-5) konfiguracije vrat in uporabe vhodov/izhodov:

```
; ********* Program za inicializacijo vrat A, B, C in brisanje RB4 ter postavitev RA5 **

       include <p16f876.inc>; vkljucitev datoteke z definicijami simbolov registrov
       list   p=16f876       ; izbira tipa cipa

; Inicializacija: vrata A (RA2, RA5 izhoda), B (vsi izhodi), C (vsi izhodi)
       clrf   PORTB  ; zacetno stanje vrat B =0, ker stanja ob RESET-u niso definirana!
       clrf   PORTC  ; zacetno stanje vrat C =0, ker stanja ob RESET-u niso definirana!

       bsf    STATUS,RP0     ; kodna stran Bank1 zaradi dostopa do TRISn registrov
       movlw  0x00           ; PORTB in PORTC - vsi izhodi
       movwf  TRISB          ; PORTB - RB(0:7) : izhodi
       movwf  TRISC          ; PORTC - RC(0:7) : izhodi
       movlw  B'00011011'    ; 0x1B ->W,  RA(2, 5): izhodi, RA(0, 1, 3, 4): vhodi
       movwf  TRISA          ; W->TRISA
                             ; za PORTA je potrebno v ADCON1 vpisati B'10000100' ali 0x84
       movlw  B'10000100'    ; B'10000100' -> W, 0x84
       movwf  ADCON1         ; W->ADCON1, desno poravnan zapis, RA(0,1,3): analogni vhodi
; Postavitev zacetnih stanj na vratih B in C
       bcf    STATUS,RP0     ; kodna stran Bank0 zaradi dostopa do PORTn registrov!
       movlw  B'11111111'    ; nastavitev stanja 0xFF na izhodih PORTC
       movwf  PORTC          ; na PORTC
       movlw  B'00001111'    ; nastavitev stanja 0x0F na izhodih PORTB
       movwf  PORTB          ; na PORTB
; Ponavljajoca se zanka
label1 movlw  0xEF           ; B'11101111'
       movwf  PORTB          ; zapiši na PORTB, RB4=0, ostali 1
       bsf    PORTA,5        ; postavi bit 5 PORTA, RA5=1
       goto   label1         ; izvajaj zanko, skoci na label1

       end                   ;psevdoukaz za zakljucek programske kode
```

Program 4-5: Popoln primer programa v zbirnem jeziku za logične signale

Popoln zgled programa za izvajanje na MPU-PIC16F876:

Primer programa v zbirnem jeziku

```
;-------------------------------------------------------------------------------
; Demo program za vklop LED diod: LD1 na RC0 in LD3 na RC3
;-------------------------------------------------------------------------------

            list        p=16f876  ;izbira tipa cipa
            include <p16f876.inc> ;vkljucitev datoteke z definicijami
                                  ;simbolov
;-------------------------------------------------------------------------
; Nastavitev vektorjev

            org         0x00      ;RESET vektor
            goto        Main      ;skok na glavni program Main
            org         0x05      ;zacetek programske kode
Main
;--------- Inicializacija registrov vhodno/izhodnih vmesnikov
            bsf         STATUS,RP0          ;banka 1
            movlw       0x06      ;v delovni register nalozi vrednost 0x06
            movwf       ADCON1    ;ne uporabljamo analognih vhodov
                                  ; (00000110)
            movlw       0x10      ;0 - izhod, 1 – vhod, b'00010000'
            movwf       TRISA     ;vsi pini porta A so izhodni, razen RA4

            clrf        TRISB     ;vsi pini porta B so izhodni
            clrf        TRISC     ;vsi pini porta C so izhodni
            bcf         STATUS,RP0          ;banka 0
            clrf        INTCON    ;prekinitev ne bomo uporabljali
            clrf        PORTC     ;izhode porta C postavi na 0

;-------- jedro programa
Zanka
            bsf         PORTC,0   ;vklop LED diode LD1 na RC0
            bsf         PORTC,3   ;vklop LED diode LD3 na RC3
            goto        Zanka     ;skok na zacetek

            end                   ;psevdo ukaz za konec programa
```

Primer programa v C-jeziku

```
/* Demo program za vklop LED diod: LD1 na RC0 in LD3 na RC3 */

#include <pic1687x.h> // vkljucitev datoteke z definicijami simbolov

void main(void)
{

/* --------- Inicializacija registrov vhodno/izhodnih vmesnikov */
ADCON1=0x06;   // ne uporabljamo analognih vhodov, 0b00000110
TRISA=0x10;  // vsi pini porta A so izhodni, razen RA4,
                // 0b00010000)
TRISB=0;       // vsi pini porta B so izhodni
TRISC=0;       // vsi pini porta C so izhodni
PORTC=0;       // izhode porta C postavi na 0
INTCON=0;      // prekinitev ne bomo uporabljali

/* -------- jedro programa */

            while (1)   // neskoncna zanka
            {
            RC0=1;     // vklop LED diode LD1 na RC0
            RC3=1;     // vklop LED diode LD3 na RC3
            }

}
```

Program 4-6: Popoln primer programa v zbirnem in C-jeziku za logične izhode

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja. Dodatna znanja iz programiranja v zbirnem jeziku so v: [2][4][6][7]. Za razumevanje programov, ki so kodirani v C-jeziku, je potrebno poznavanje osnov ANSI C strukturnega programiranja: [15][11][13][5]. Veliko rešitev praktičnih nalog z uporabo PIC mikrokrmilnikov je na spletnem portalu http://www.interq.or.jp/japan/se-inoue/e_pic6.htm.

Vprašanja za utrjevanje:

Napišite podprogram za inicializacijo vseh linij vrat PORTB kot izhodi.

Rešitev:

```
PBizh   bsf     STATUS,RP0      ; 1 -> RP0 , izbira Bank1 za dostop do TRISB
        clrf    TRISB           ; 0 -> TRISB, vsi izhodi
        bcf     STATUS,RP0      ; 0 -> RP0 , izbira Bank0 za dostop do PORTB
        return                  ; konec podprograma
```

1. *Napišite podprogram za inicializacijo vseh linij vrat PORTC kot vhodi.*
2. *Kakšen pomen imajo linije vrat PORTA ?*
3. *Napišite programček za ugotavljanje stanj na vhodih: RA0, RA1 in RA3 ?*
4. *Stanje stikala na vhodu RA4 prenesi (programsko) na izhod RC3.*
5. *Izhode na RC0 in RC1 preklapljaj izmenoma vsakih (pribl.) 100 ms.*