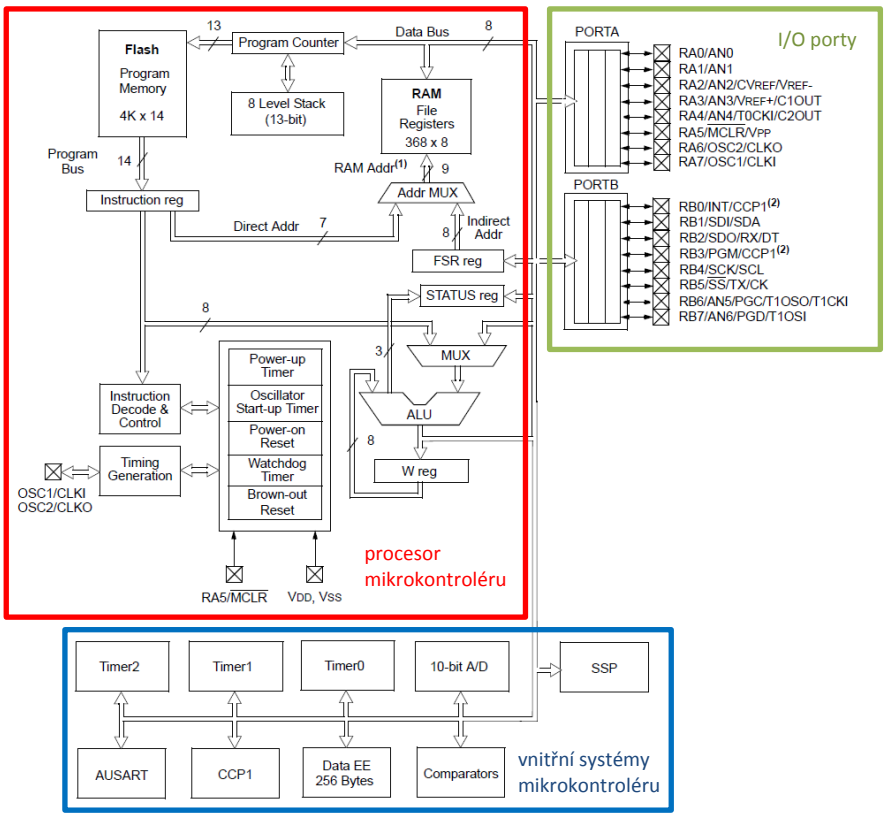
Paralelní I/O porty

Upozornění: Pokud programujeme procesor ve vyšších jazycích, jako například Basic, nebo C, nemáme přístup k vnitřním obvodům a sytémům, tedy do vnitřní struktury procesoru. Nemůžeme ji přímo využívat a proto se podstatně se zmenší operační rychlost mikrokontroléru. Nevíme přesně, jak dlouho trvá provádění jednotlivých příkazů, což výrazně snižuje možnosti využití procesoru pro řízení systémů v reálném čase.

Vnitřní struktura mikrokontroléru PIC16F88



Procesor mikrokontroléru je ve skutečnosti jádrem mikrokontroléru. Provádí například instrukce, obsahuje aritmeticko-logickou jednotku ALU, která provádí aritmetické a logické operace. Řídí běh programu, činnost podprogramů, obsahuje generátor hodin procesoru, realizuje systém přerušení, obsahuje paměť programu, paměť dat (univerzální registry), obvody ochrany dat i programu proti nedovolenému vnějšímu přístupu i hlídací časovač, který umožňuje bezpečný a spolehlivý běh programu. Těma procesor je velmi rozsáhlé téma. Podrobněji se s ním seznámíme postupně v průběhu několika let a naučíme se, jak jej efektivně využívat.

Vnitřní systémy mikrokontroléru nejsou u všech mikrokontrolérů stejné. Liší se v počtu modulů, jejich činností, schopnostech i druhu. Jednotlivými moduly se budeme posupně podrobně zabývat a naučíme se je využívat.

I/O porty mikrokontroléru – jsou vývody (piny) mikrokontroléru a jejich řídící obvody, které propojují vnitřní obvody mikrokontroléru s okolními systémy a umožňují obousměrnou komunikaci mikrokontroléru a jeho vnitřních systémů s okolními systémy, řízenými mikrokontrolerem.

Porovnáme-li mikrokontrolér s počítačem, lže říct, že **mikrokontrolér je vlastně počítač v jednom pouzdře** (celý počítač v jedné součástce). V počítačích je procesor jedna nebo více součástek, jednotlivé moduly vnitřních systémů mikrokontroléru jsou realizovány, jako celé desky, nebo několik obvodů v počítačích. I/O porty počítačů obvykle připojujeme zařízení, jako jsou například myši, tiskárny, měřící zařízení apod.

Procesor mikrokontroléru, vnitřní systémy i porty jsou vzájemně propojeny osmibitovou datovou sběrnicí (DataBus).

Postup při návrhu systémů řízených mikrokontrolérem

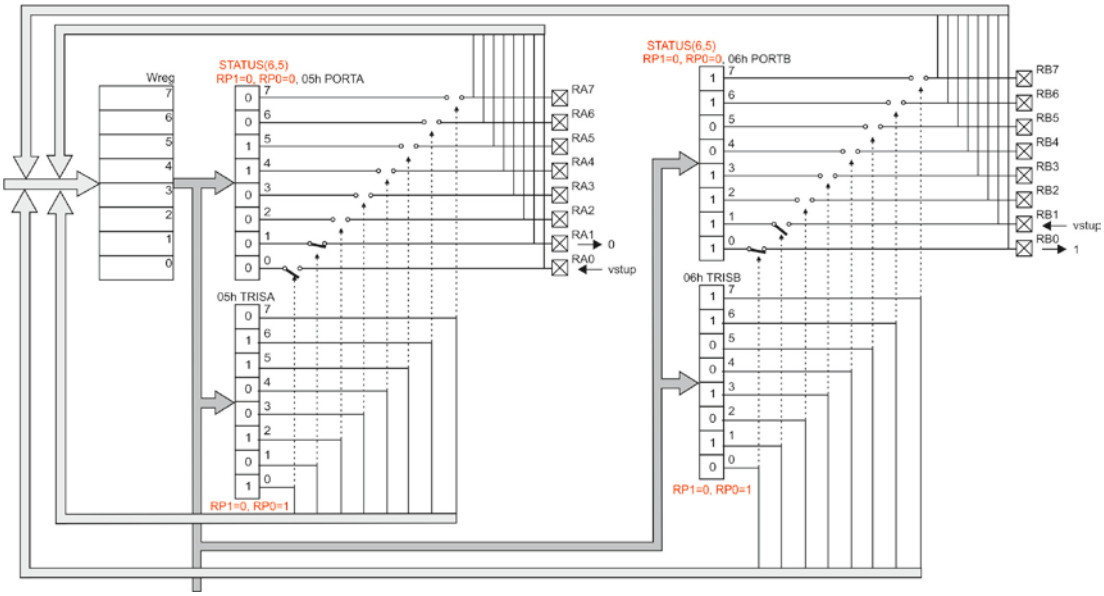
Jak jsme se již dozvěděli, mikrokontrolér kromě procesoru, obsahuje řadu velmi výkonných periferních systémů, jako jsou paralelní digitální I/O porty, různé čítače, časovače, komparátory, komunikační systémy, SPI, SSI a mnoho dalších. Každá aplikace, kterou budeme vyvíjet, vyžaduje použití jistých hardwarových obvodů, které budou dostatečně rychle provádět požadované činnosti. Podle toho vybíráme přesný typ mikrokontrléru, který je jimi vybaven. Důležitým hlediskem je cena výsledného systému a výkon systému. Obzvlášť jde-li o řízení systémů pracujících v reálném čase.

Při vývoji systémů tedy nejdříve, podle požadavků konstrukce zařízení, zvolíme typ mikrokontroléru, který obsahuje požadované obvody. Nyní budeme pracovat s mikrokontrolérem PIC16F88.

Aby se zvolený mikrokontrolér mohl použít, je třeba jej správně nakonfigurovat – nastavit tak, aby pracoval přesně jak je třeba.

Následujícím krokem je tedy konfigurace systémů mikrokontroléru. V našem případě budeme pracovat s digitálními vstupy/výstupy mikrokontroléru – tedy s jeho I/O porty.

Práce s registry



Na obrázku je zobrazeno zjednodušené propojení registrů Portu A a Portu B, které umožňují vstup a výstup da z/do mikrokontroléru. Nyní si podrobně ukážeme, jak vše pracuje.

Registr na adrese 05h (RP1=0, RP0=0 a adresa registru 05h) se nazývá PORTA. Tento registr, který je nazývám portem (bránou) je zajímavý tím, že vše co do něj zapíšeme se oběví na vývodech RA7/0, jsou-li nastaveny jako výstupy. Podobně obsah registru na adrese 06h (RP1=0, RP0=0 a adresa registru 06h) PORTB se oběví na vývodech RB7/0, jsou-li nastaveny jako výstupy.

Zda daný vývod bude vstupem či výstupem určuje obsah registrů TRISA a TRISB. Je-li příslušný bit těchto registrů v 1 (H), bude příslušný vývod nastaven jako vstup, bude-li v 0 (L), bude nastaven jako výstup.

Úkol: Dokreslete do obrázku podle hodnot v TRIS registrech polohu spínačů, připojujících PINy k registrům a tím i konfigurují vstupy/výstupy. Dále tam pak doplňe k výstupním pinům hodnoty, podle stavu bitů portů. Označ také piny, nastavené jako vstupy.

Adresa registru říká instrukcím, pracujícím s registry, s kterým registrem se má pracovat.

Poznámka: *Je to podobné, jako když vy chcete někomu zatelefonovat. Musíte znát jeho telefonní číslo, nebo by se dalo říct adresu telefonu. Celá adresa telefonu se skládá z několika čísel. První část říká stát, druhá pak operátora a třetí je vlastní číslo uživatele. Například telefonní číslo +420 602 575 148 – červeně je číslo státu (+420 nebo 00420 je Česká republika, +421 nebo 00421 je Slovenská republika....) modře je trojčíslí, které udává operátora (každý operátor má svá trojčíslí a nemusí být po sobě jdoucí čísla). Černé číslice jsou číslem konkrétního telefonu.*

Adresa registru je tvořena 9 bity (dvojkovými číslicemi). Registry tedy mají adresu od 000h (0 0000 0000) po adresu 18fh (1 1111 1111). Při práci s těmito adresami pro volbu registru se mohou používat celé adresy jen při takzvaném nepřímém adresování, kterým se zatím nebudeme zabývat.

Při **přímém adresování** – tedy při předávání adres registrů přímo instrukcím, tak jsme zvyklí programovat doposud, je třeba přímou adresu každého registru vytvořit následujícím způsobem:

7 spodních bitů adresy používáme přímo v instrukcích (00h až 7fh). Horní dva bity musíme před použitím instrukce s adresou registru nastavit do bitů 6 a 5 registru STATUS – bity RP1 a RP0. Tedy osmý bit adresy registru musí být před provedením instrukce nastaven do RP0 a devátý bit adresy registru musí být před provedením instrukce nastaven v RP1 registru STATUS.

TRISA registr je na adrese 85h (RP1=0, RP0=1 a adresa registru 05h) a TRISB registr je na adrese 86h (RP1=0, RP0=1 a adresa registru 06h). Chceme-li zapsat data do registru, musíme zvolit adresu požadovaného registru (nejprve nastavit bit RP0 – 5. bit registru STATUS) a pak použít v instrukci adresu registru (00h až 7fh). Říkáme tomuto přímé adresování registru. RP0 je při přímém adresování osmým bit adresy registru.

Upozornění! Existuje pouze 6 registrů, které když adresujeme, nezáleží na nastavení bitů RP1 a 0. Jedním z nich je **registr STATUS**. Bez ohledu na to, použijeme-li pro jeho adresaci adresu 03h, 83h, 103h nebo 183h, vždy se pracuje s tím samým registrem. Tedy při **práci s registrem STATUS do instrukcí používáme adresu 03h bez ohledu na to, jak jsou nastaveny jeho bity RP1 a RP0!**

Poznámka: *Univerzální registry – paměť RAM mikrokontroléru PIC16F88. Ukládáme do nich hodnoty proměnných apod. nalezneme na těchto adresách:*

20h až 7Fh (RP1=0,RP0=0 adresa 20h až 7Fh) prvních 96 bytů.
A0h až EFh (RP1=0,RP0=1 adresa 20h až 7Fh) dalších 80 bytů.
110h až 16Fh (RP1=1,RP0=0 adresa 10h až 7Fh) dalších 96 bytů.
190h až 1EFh (RP1=1,RP0=1 adresa 10h až 7Fh) dalších 96 bytů.

Inicializace portů v assembleru

Po zapnutí napájení (Power On Reset) jsou piny PORTA<4:0> nastaveny jako analogové vstupy a čteny jako 0. Chceme-li tedy celý PORTA používat jako digitální I/O port, musíme vynulováním registru 9Bh ANSEL (RP0=1 adresa 1Bh) tyto analogové vstupy přepnou do režimu digitální I/O. Toto není třeba, programujeme-li v WJEC assembleru PICAXE18M2!

Po zapnutí napájení je automaticky nastaveno RP1 = 0 a RP0 = 0 registru STATUS.

```
CLRF    PORTA           ;vynuluje výstupy I/0 portů do 0
CLRF    PORTB
BSF     STATUS,RP0      ;nastaví se RP0 do 1
CLRF    ANSEL           ;přepne výstupy portu A do režimu digitální I/O – ne při práci s PICAXE!
MOVLW   0xFF            ;celý port A nastaví do režimu vstup
MOVWF   PORTA
MOVLW   0x00            ;všechny piny portu B nastaví jako výstupy
MOVWF   PORTB
BCF     STATUS,RP0      ;nastaví se RP0 do 0
```

Popis použitých instrukcí

CLRF	*Clear f*	*Vynuluj registr*

CLRF *f*

00h do (f) f je adresa registru (dolních 7 bitů adresy, horní 2 bity jsou v bitech RP1 a RP0 registru STATUS)
STATUS (bit 2) Z bit nastaví do 1

BSF	*Bit Set f*	*Nastav bit registru*

BSF *f,b*

1 do (f<b>) nastaví jedničku do bitu b (bit 0 až 7) registru na adrese f (dolních 7 bitů adresy, horní 2 bity jsou v bitech RP1 a RP0 registru 03h - STATUS)

BCF	*Bit Clear f*	*Vynuluj bit registru*

BCF *f,b*

1 do (f<b>) nastaví nulu do bitu b (bit 0 až 7) registru na adrese f (dolních 7 bitů adresy, horní 2 bity jsou v bitech RP1 a RP0 registru 03h - STATUS)

MOVLW	*Move Literal to W*	*Ulož konstantu do W registru*

MOVLW *k*

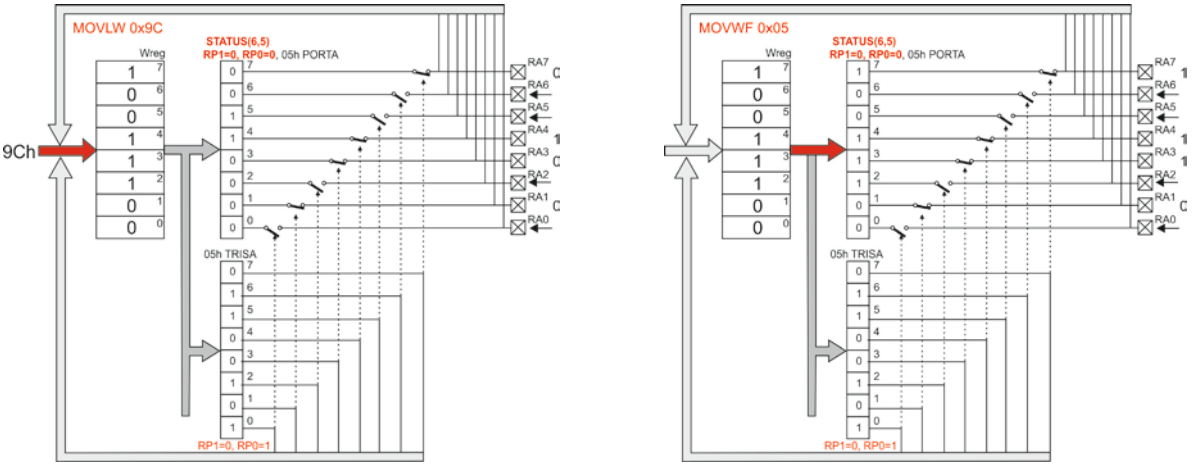
k do (W) uloží konstantu do W registru
k – konstanta 00h až FFh

MOVWF	*Move W to F*	*Ulož obsah registru W do registru o adrese F*

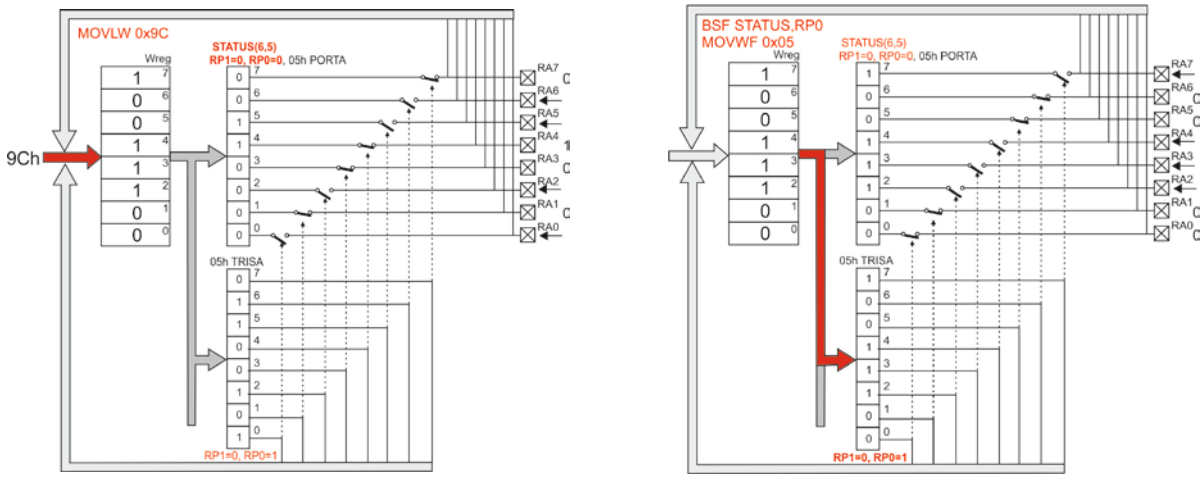
MOVWF *f*

(W) do (f) přesune (zkopíruje) obsah W registru do registru f (dolních 7 bitů adresy, horní 2 bity jsou v bitech RP1 a RP0 registru 03h - STATUS)

Poznámka: Potřebujeme-li uložit konstantu (hodnotu) do nějakého registru, musíme ji nejprve uložit do W registru a až potom z W registru do požadovaného registru.



Zápis konstanty do portu A



Nastavení I/O portu A