Vstupně výstupní linky

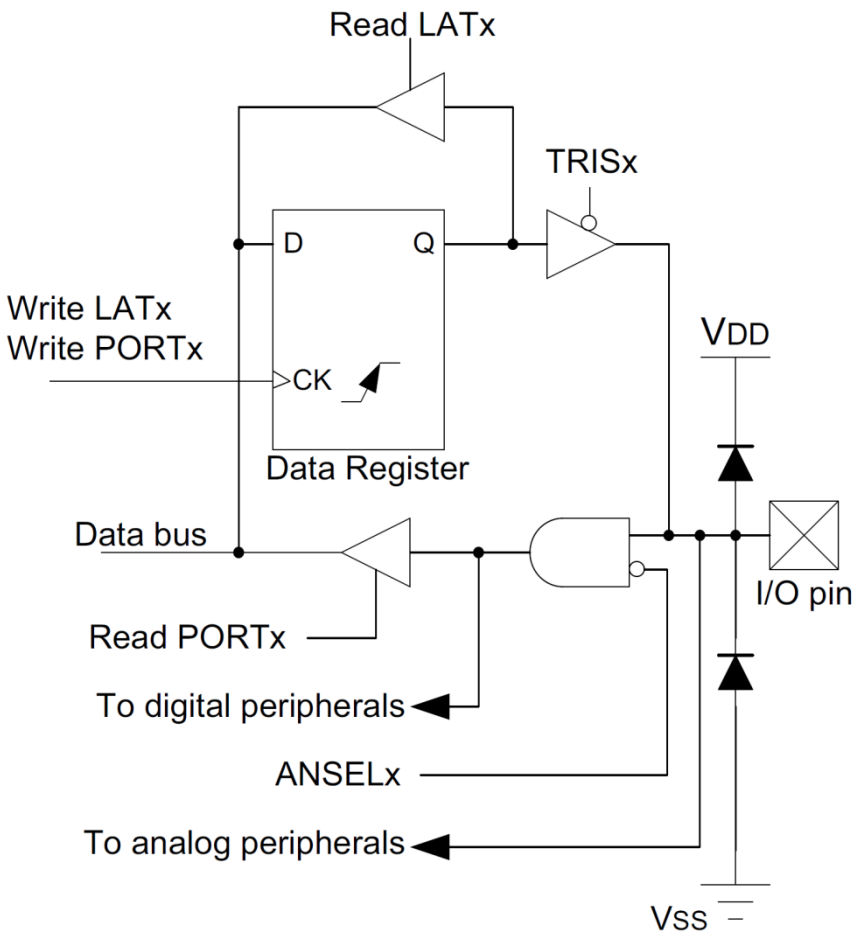
Obousměrné, 3-stavové výstupy

Registr TRISx řídí směr vývodu bitem na příslušné pozici
- 0 … 0ut, výstup
- 1 … 1n, vstup (HiZ)

Čtení registru PORTx vrací bity podle úrovně na vývodech

Čtení registru LATx vrací bity výstupního bufferu

Zápis bitů do registrů PORTx či LATx nastavuje odpovídající úroveň na vývodech



Asembler MPASM

Překladač z jazyka symbolických adres do strojového kódu

Vyjádření čísel v číselných soustavách:

desítková	123	(přepínání bank)
hexadecimální	0x7B	(adresy)
binární	01110101B	(nast. registrů po bitech)
ASCII znaky	‘A’	

Doporučení pro označování symbolů:

bity, konstanty	velká písmena	CY, MAX
registry	malá písmena	citac, y
návěští	počáteční velké	Skok, Funkce1
	dvojtečka za návěštím	(Skok: goto Skok)

Komentář ;toto je komentář

Základní pseudoinstrukce

- **PROCESSOR** *typ_procesoru*

  `;informace překladači, pro jaký procesor je kód`

- **#include** "*soubor*"

  `;vloží soubor s def., podprg., knihovnami...`

- **#define** *název registr*, *bit*

  `;definuje název pro bit registru`

- *název* **EQU** *hodnota* `;definice konstanty`

  ORG *hodnota*

  `;nastaví adresu pro uložení následujícího kódu`

- **END** `;konec kódu`

- **goto $-1** `;skok zpět o 1 instrukci`

Instrukce přesunu dat

MOVF f,d	obsah registru f přesune do W (je-li d=0) nebo zpět do f (je-li d=1, slouží k testování); registr f ~ 0 až 7Fh (127)
MOVLW k	registr W je naplněn konstantou k (8 bitů)
MOVWF f	obsah registru W se přesune do registru f
RLF f,d	rotuje obsah f doleva přes bit C, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
RRF f,d	rotuje obsah f doprava přes bit C, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
SWAPF f,d	zamění spodní a horní 4 bity registru f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)

Aritmetické a logické operace

ADDLW k	sečte registr W s konstantou k, výsledek uloží do registru W
ADDWF f,d	sečte W s registrem f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
ANDLW k	logický součin W a k, výsledek uloží do W
ANDWF f,d	logický součin W a f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
DECF f,d	odečte jedničku od f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
INCF f,d	přičte jedničku k f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
COMF f,d	jedničkový doplněk f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)

Aritmetické a logické operace

IORLW k	logický součet W a k, výsledek se uloží do W
IORWF f,d	logický součet W a f, výsledek se uloží do W (je-li d=0) nebo do f (je-li d=1)
SUBLW k	odečte W od konstanty k, výsledek uloží do W (řeší se přičtením dvojkového doplňku W; je-li výsledek ≥ 0, pak C=1)
SUBWF f,d	odečte W od registru f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
XORLW k	nonekvivalence W a k, výsledek uloží do W
XORWF f,d	nonekvivalence W a f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)

Instrukce nulování a nastavení

BCF f,b	vynuluje bit b v registru f
BSF f,b	nastaví do log.1 bit b v registru f
CLRF f	vynuluje obsah registru f
CLRW	vynuluje obsah registru W
CLRWDT	nuluje čítač WDT a jeho předděličku (je-li k WDT připojená)

Instrukce skoků v programu

BTFSC f,b	je-li bit b v registru f v log.0, následující instrukce se neprovede
BTFSS f,b	je-li bit b v registru f v log.1, následující instrukce se neprovede
DECFSZ f,d	odečte jedničku od obsahu registru f a výsledek uloží do W (je-li d=0) nebo do f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede
INCFSZ f,d	přičte jedničku k obsahu registru f a výsledek uloží do W (je-li d=0) nebo do f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede
GOTO a	adresa „a“ se uloží na spodních 11 bitů PC (zbývající 2 b se doplní z 3. a 4. bitu PCLATH) program pokračuje na adrese PC

Podprogramy a přerušení

CALL a	návratovou adresu PC+1 uloží do zásobníku a pokračuje do podprogramu
RETURN	naplní PC ze zásobníku (návrat z podprogramu)
RETLW k	naplní PC ze zásobníku a registr W naplní osmibitovou konstantou k
RETFIE	naplní PC ze zásobníku a povolí se přerušení nastavením bitu GIE do log.1 (návrat z přerušení)

Zvláštní instrukce

NOP	neprovede nic
SLEEP	procesor přejde do stavu „spánku“

Většina instrukcí trvá jeden strojový cyklus

Dva cykly zaberou vždy instrukce skoků CALL, GOTO, RETURN, RETLW, RETFIE a nastane-li skok také instrukce BTFSC, BTFSS, DECFSZ, INCFSZ

RESET mikrořadiče

RESET znamená nový start procesoru a může být vyvolán z těchto důvodů:

- zapnutí napájení
- nulování vstupního signálu !MCLR (RESET)
- přetečení WDT
- volání instrukce RESET

Při resetu je vynulován čítač instrukcí (program začíná na adrese 0000h), PCLATH je nastaven na 0, všechny porty jsou nastaveny jako analogové(!) vstupní, všechna přerušení jsou zakázána, zápis do EEPROM je zakázán.

Konfigurační slova

Procesor obsahuje *konfigurační slova*, která jsou přístupná pouze během programování procesoru a slouží k nastavení různých režimů, příkaz CONFIG

MPLABX má pro jejich nastavení jednoduché a intuitivní rozhraní s automatickým generováním kódu:

Window –> PIC memory views –> Configuration bits

Volba zdroje taktu, Watchdog Timer, Power-up Timer, Brown-out, Code & Data Protection…

Příklad – V/V operace

```
;RS-KO s BT1 a BT2 vstupy a LED1 jako výstupem
      PROCESSOR 16F1508

#define pinSET          PORTA,4
#define pinRESET        PORTA,5
#define LED             PORTC,5

; window -> TMW -> conf. Bits
; CONFIG1
CONFIG  FOSC = INTOSC               ; Oscillator Selection Bits (INTOSC oscillator: I/O function on CLKIN pin)
CONFIG  WDTE = OFF    ; Watchdog Timer Enable (WDT disabled)
CONFIG  PWRTE = OFF   ; Power-up Timer Enable (PWRT disabled)
CONFIG  MCLRE = ON    ; MCLR Pin Function Select (MCLR/VPP pin function is MCLR)
CONFIG  CP = OFF      ; Flash Program Memory Code Protection (Program memory code protection is disabled)
CONFIG  BOREN = ON    ; Brown-out Reset Enable (Brown-out Reset enabled)
CONFIG  CLKOUTEN = OFF              ; Clock Out Enable (CLKOUT function is disabled. I/O or oscillator function
                      ;on the CLKOUT pin)
CONFIG  IESO = ON     ; Internal/External Switchover Mode (Internal/External Switchover Mode is enabled)
CONFIG  FCMEN = ON    ; Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor is enabled)
; CONFIG2
CONFIG  WRT = OFF     ; Flash Memory Self-Write Protection (Write protection off)
CONFIG  STVREN = ON   ; Stack Overflow/Underflow Reset Enable (Stack Overflow or Underflow will cause a Reset)
CONFIG  BORV = LO     ; Brown-out Reset Voltage Selection (Brown-out Reset Voltage (Vbor), low trip point
                      ;selected.)
CONFIG  LPBOR = OFF   ; Low-Power Brown Out Reset (Low-Power BOR is disabled)
CONFIG  LVP = ON      ; Low-Voltage Programming Enable (Low-voltage programming enabled)
```

Příklad – V/V operace

```
#include <xc.inc>
;*****************************************************************************
PSECT PROGMEM0,delta=2, abs
RESETVEC:
   ORG        0
   PAGESEL    Start
   GOTO       Start

   ORG        4
   nop
   retfie
Start:
   movlb      1              ; 1 go to bank 1 because osccon is there
   movlw      00100000B      ; 62.5kHz (page 59)
   movwf      OSCCON         ; set clocks
   call       Config_IOs     ; subroutine to set ports
   movlb      0              ; go to bank 0
Main:
   btfsc      pinSET         ; if 1 -> go to next line, else skip
   bsf        LED            ; set to 1
   btfsc      pinRESET
   bcf        LED
   goto       Main

   #include    "Config_IOs.inc"
END
```