PIC mikrovezérlők (16F87x)- segédlet

Utasításkészlet: (főbb utasítások)

No operation		**NOP**	Szám mínusz w		**SUBLW** K
F két felének felcserélése	bájtos utasítás	**SWAPF** F,D	Megegyező vagy szám és w		**IORLW** K
Clear w		**CLRW**	Kizáró vagy szám és w		**XORLW** K
Clear fájlregiszter		**CLRF** cím	Szám&w		**ANDLW** K
Inrement f		**INCF** F,D	Cy+f (mac.)	bites utasítások	**ADDCF** F,D
Decrement f		**DECF** F,D	Dc+f (mac.)		**ADDDCF** F,D
Complement f		**COMF** F,D	Ugrás k címre, ha cy=1 (mac.)		**BC** címke
Increment f - követk. ut. Átlép, ha az eredmény=0		**INCFSZ** F,D	Ugrás k címre, ha dc=1 (mac.)		**BDC** címke
Decrement f- követk. ut. Átlép, ha az eredmény=0		**DECFSZ** F,D	Ugrás k címre, ha cy=0 (mac.)		**BNC** címke
Move f (honnan hova) általános		**MOVF** F,D	Ugrás k címre, ha dc=0 (mac.)		**BNDC** cimke
Move w to f		**MOVWF** F	Ugrás k címre, ha z=0 (mac.)		**BNZ** cimke
Move L to W (szám a W-be)		**MOVLW** K	Ugrás k címre, ha z=1 (mac.)		**BZ** cimke
Move f to w (mac.)		**MOVFW** F	Z törlése		**CLRZ**
F mínusz w (substract)		**SUBWF** F,D	Cy törlése		**CLCR**
F+w		**ADDWF** F,D	Dc törlése		**CLRDC**
Rotate right f		**RRF** F,D	Z=1		**SETZ**
Rotate left f		**RLF** F,D	Cy=1		**SETC**
W&f		**ANDWF** F,D	Dc=1		**SETDC**
Megegyező vagy w és f		**IORWF** F,D	Átlép, ha z=0		**SKPNZ**
Kizáró vagy w és f		**XORWF** F,D	Átlép, ha z=1		**SKPZ**
Go standby (watchdog, vagy intr ébreszti)		**SLEEP**	Átlép, ha cy=0		**SKPNC**
Clear watchdog timer		**CLRWDT**	Átlép, ha cy=1		**SKPC**
Return, számot a w-be		**RETLW** K	Átlép, ha dc=0		**SKPNDC**
Return-szubrutinból (nem intr)		**RETURN**	Átlép, ha dc=1		**SKPDC**
Return from interrupt		**RETFIE**	Szám+w		**ADDLW** K
Call szubrutin		**CALL** cimke	Bit clear f		**BCF** F,B
Lapozás (direktíva, programmemóriában)		**PAGESEL** cimke	Bit set f		**BSF** F,B
Direkt bankváltás (direktíva) regiszterfájlban		**BANKSEL** cimke	Bit test f átlép, ha 0		**BTFSC** F,B
Indirekt bankváltás (direktíva) regiszterfájlban		**BANKiSEL** cimke	Bit test f átlép, ha 1		**BTFSS** F,B
Hosszú ugrás (mac.)		**LGOTO** cimke	Sima ugrás		**GOTO** cimke
Hosszú szubrutinhívás (mac.)		**LCALL** cimke			

Jelölések:

F: Fájlregiszter címe, vagy cimkéje

D: Művelet eredménye hova kerül: 0=W 1-F

K: 8bites szám vagy karakter ``-jelek között

B: hanyadik bitje a regiszternek

Bármilyen cím helyett lehet cimkét megadni (f,b)

Bármilyen regisztert, vagy regiszterbitet el lehet nevezni, a #define név cím, vagy #define nev cím,bitsorszám alapján

Bármely számot el lehet nevezni(név equ k)(8bites,nem cím)

Számok: default: hexa (0x). bináris: b'10110010'

Stack-be mentés:Az FSR-t használjuk **ram** stack pointernek Ennek kezdőértéket adunk, majd mentés után inkrementájuk.

W a working register (akkumulátor)

A mac-jelüek többutasításból álló makrók (MPASM)

Indirekt címzés: Ha cím helyett azt adjuk meg, hogy INDF, Akkor az a cím=az FSR (címlatch) -ben tárolt címmel (regiszterfájlra vonatkozó címzés) Úgy is felfogható, hogy Annak a regiszternek a neve INDF, amit az FSR éppen meg-Címez. Az FSR-be 8bites adatot írunk. Lehet inkrementálni

Szubrutinok kezdőcímei csak a lapok első felén lehetnek, de A szubrutinok átnyúlhatnak a másik félre is. A PCL-írásos Ugrásokra is igaz. (régi, NEM 16f87x, 12 bites PIC-eknél)

Lapozás: vagy pagesel-lel, vagy ami ugyanez, a PCLATH-Írásával is megoldható (pc felső 5 bitje, ennek a 3. 4. bitje.) Lapzásos direktívák, és összetett MASM utasítások esetén a skip-ekre vigyázni kell, mert ezek több út.-ból állnak.

Változó: nevet adunk egy regiszternek, vagy bitnek

Reset vektor:00h intrvektor:04h letöltő progi kilépővek-Tora:03h->ide goto utasítást! (egyszavas:nem lóg át 04h-ra)

Az első 2 bank váltás nélkül címezhető, de csak indirekten!

Fontosabb direktívák:

Konstans definiálás		**CONSTANT**	Konfigurációs beállítások		**__CONFIG**
Bit, bájt neve		**#DEFINE** név cím	Szöveg tárolása programmemóriában		**DA**
vége		**END**	Bájt definiálása		**DB**
Konstans (számot, vagy címet helyettesít névvel)		név **EQU** szám	Szó (16bit) definiálása		**DW**
programkezdet		**ORG** kezdőcím	Táblázat megadása		**DT**
Processzor		**PROCESSOR**	Aktuális címet helyettesíti (inkrement)		**$**
Alapértelmezett szám		**RADIX**	Szimbólumhoz értéket (változó)		név **SET** szám

A perifériák általában default le vannak tiltva, a reset által, Kivétel az A/D, ami engedélyezve van az a porttal közös lábon. Csak azt kell inicializálni, amit használunk, kivéve néhány regisztert. A megszakítások is le vannak tiltva a resettel.

Watchdog: 8bites számláló, kapcsolódik hozzá 8bit előosztó, és 8bit utóosztó. Ha eléri FFFFFFh-t akkor resetel, megelőzése:clrwdt-utasítás, vagy letiltás a tok programozásakor: WDT_OFF

CONFIG:A program elejére kell a konfigurációs sor, ami a programbetöltéskor beíródik. Itt kell megadni az oszcillátor típusát, vagy hogy kell-ewatchdog…**__CONFIG BEÁLLÍTÁSOK**

Az egyes beállításokat szóközzel, és&-jellel választjuk el. Pl:: __CONFIG _CP_OFF & _WDT_OFF… Általános beállítások: Oszcillátortípus: _RC_OSC (RC), _LP_OSC (alacsony freq. kristály), _XT_OSC (kristály vagy kerámia rezonátor), _HS_OSC (high speed-kristály) Watchdog: _WDT_ON/_WDT_OFF Kódvédelem: _CP_OFF Power on timer: _PWRTE_ON Brown out reset (tápfesz leesik): _BODEN_ON Ezen felül lehetnek még: (PIC16F877) Programmemóriába írás engedély: _WRT_ENABLE_ON 5V-os írás eng.: _LVP_ON Debug mód eng.: _DEBUG_ON Adatmem.-védelem engedélyezése: _CPD_ON Tok azonosító: Ezt a config előtt kell. Ez 4db hexa szám, ez azonosítja az ic-t. **_ _IDLOCS H`1234´ A config a nonitorprogramban van!**

Főbb perifériák, regiszterek, és inicializálás:

●**Státusz regiszter (STATUS) (03h, 83h)** 7.bit:IRP bankválasztás indirekt címnél 5.-6.:RP bankválasztás 4.:TO watchdog time out 3.:PD power down sleep:0 2.:Z Zero 1.:DC-félátvitel 0.:C Carry

●**Megszakítások engedélyezése: INTCON** (főbb megszakítások, általános megsz. Bizonyos bitjeivel engedélyezünk, bizonyos bitjei pedig jelzik, hogy ki kérte a megszakítást)**, PIE1, PIE2** (perifériáktól jövő IT-k. pl. A/D, SSP) **–**regiszterekkel. **(1-ENGED/van, 0-TILT/nincs)** A **PIR1, PIR2**-jelzik a megszakításkérőt. Ellenőrizni kell, mert csak egy megszakítási vektorcím van.

INTCON-reg.**(0Bh,8Bh)**: Az alsó 3 bit jelző, a többi engedélyező bit. (**A megszakítást jelző biteket törölni kell!**)

7.bit:GIE glob. intr. En. 6.bit: EEIE EE-írás kész IT 5.bit: TOIE Tmr0-IT-je 4.bit:INTE RB0/INT-en jövő intr. 3.bit:RBIE Bport-it-eng. 2.bit:TOIF Tmr0-it-jelz. 1.:INTF RB0-int-jelz. 0.:Rb-it-jelz

●**I/O-Portok:** (resetre mind bemenet lesz) Azt, hogy az adott port-bit bemenet, vagy kimenet, a **TRISA** (A-port), **TRISB** (B-port)… regiszterekbe kell betölteni. (0=out)A portok kös lábon vannak más perifériákkal, pl A/D (ra), komparátor, megszakításbemenet(rb). Az ide tartozó más perifériákat le kell tiltani, ha portként akarjuk használni a lábakat. Ezek általában le vannak tiltva a resettel, de pl (PIC16F871-alapján) az a porton lévő A/D converternek nincs, azt az **ADCON1**-re kiadott beállítással lehet tiltani (adcon1,3..adcon1,0=011x általában) Resettel az egyszerre dig. Bem., és analóg bem. Is. A B port egyes lábait szokták megszakításra is használni (élérzékeny). Ha az intcon,3 engedélyezve van. Ekkor ellenőrizni kell, hogy melyik Bport-láb=1, mert úgy tudjuk kiválasztani a megfelelő szubrutint. Ezután töröljük a megsz. Jelző bitet, az INTCON-ban. Ezek a megszakítások default le vannak tiltva. C-port:I/O, usart, compare capture, pwm., számláló kivezetések. Az I/O-n kívül minden le van tiltva defaultból. **Trisc**: C irány D port: I/O, parallel slave port. Azt, hogy melyik, az E port **TRISE** regiszterében állítjuk be. **TRISD**: D iránya. E port: 3 I/O vonal, A/D, Port D vezérlőjelei. I/O, vagyA/D akkor, ha TRISE-ben az állítjuk be.A TRISE szolgál a 3 e vonal irányát megadni, és a PPIO státusz, és vezérlőjeleinek az írására, olvasására. Default:I/O.

●**A/D átalakító:** Az A, és E port lábai lehetnek a bemenetei. Inic.: a TRIS regiszterben beállítjuk, hogy a használni kívánt láb bemenet legyen (default is ez). Az **ADCON0**,6-7 bitekkel az órajelet beállítjuk: 00-1MHz-es kvarcnál, 01-5MHz, 10-20MHz, 11-RC oszcillátor használatakor AZ ADCON0,0 1-be állításával bekapcsoljuk az a/d átalakítót. Az **ADCON1**-benbeállítjuk (**ADCON2**) hogy melyik portláb legyen analóg, melyik digitális bemenet, mi legyen a Vref (alsó 4 bit) beállítjuk a kimeneti formátumot: 7. bit: legyen 1! (ha mindenhova analógot akarunk: CLRF ADCON1 majd BSF ADCON1,7). A konverziót indítani kell, időbe telik a működése. A folyamat LÉPÉSEI: 1:kiválasztjuk a bemeneti csatornát (ADCON0,5-3), 2: mintavételi időt várunk (12µs) 3: indítjuk a konverziót a **GO**-bit 1-be állításával (ADCON0,2) 4: Biz. idő múlva bekerül az eredmény (10bit) a **ADRESH,** és **ADRESL** regisztерekbe (2 és 8bit) 5: kiolvasás. Hogy mikor kerül az eredmény a 2 regiszterbe? 3mód van: a) várunk anyit, amenyi biztos elég (22µs). b) figyeljük, hogy a ADCON0,GO-bit mikor lesz=0. akkor lehetlesz kiolvasni c) engedélyezzük a globális, és a perifériamegszakítást, és ha kész a konverzió, akkor az megszakítást okoz. Engedélyezése: **PIE1,ADIE** ellenőrzése, törlése: **PIR1,ADIF**

●**Számlálók, időzítők:** Ezek a TMR0 (8bit, minden PICben benne van), TMR1 (16bit), és TMR2 (8bit). Ezekhez tartozhatnak elő,-és utóosztók. Ezek túlcsordulása megszakítást okozhat. A TMR0-hoz, és a WDT-hez tartozik az **OPTION_REG** regiszter. Ebben kell beállítani az előosztó-hozzárendelést(ha TMR0-hoz rendeljük, akkor mindkettő kap belőle), annak mértékét (mind a WDT-nek, mind a másiknak), az órajelkiválasztást(belső clk, vagy Icláb: T0CKI), megszakítás élének a kiválasztása, felhúzó ellenállás engedélyezése. A **TMR0** regiszter mutatja a számlálás helyzetét (R/W). A TMR1 számláló: A bemenete vagy belső oszcillátor jele/4, vagy külső:T1CKI,(vagy külső kvarc). Sleep alatt is megy, ébreszti belőle a processzort (megszakítást okozhat). Mivel 16 bites, ezért az aktuális értéket a **TMR1L**, és a **TMR1H** tartalmazza (R/W) Inicializálása: **T1CON** (5-6bit: előosztás 11-1:8…00-1:1, 3bit: külső oszcillátor engedélyezése ha külső kvarccal akarjuk 2.bit: 0-aszinkron 1-szinkron 1.bit: forrás1-külső jel, 0-osc/4 0.bit:1-on 0-off) A megszakítás engedélyezése a **PIE1,TMR1IE** regiszterben. Figyelése: **PIR1,TMR1IF** A TMR2-t használjuk a soros adatátviteli egység ütemadójaként. A period (**PR2**-kimenet periódusideje) regiszterbe írunk egy adatot, és ez ha megegyezik TMR2 értékével, akkor okozhat megszakítást egy utóosztás után. A TMR2-nek is van túlcsordulás kimenete. IT engedélyezése: **PIE1,TMR2IE.** Inicializálás: **T2CON** (engedélyezés, utóosztás, előosztás).

●**Komparátor:** 2db komparátort tartalmaz, ezeket több módon lehet összekepcsolni, amit a **CMCON** (a bemenetek hova kapcsolódnak, kimenet milyen) reg.-ben adunk meg. A bemenetei az RA-ra kapcsolódnak. Van hozzá referencia feszültség modul (beállítása:**VRCON**:on/off,Vref) Ha a komparátor kimenete megváltozik, az megszakítást okozhat. Akkor változik, ha a 2 bemenet átvált.

●**Capture/compare modul**. Két ilyen egység van. Mindkettő használja a 16 bit tárolására a **CCPR1H, CCPR1L**, a másik a **CCPR2H**, és a **CCPR2L**-regisztereket. Inic.:**CCP1CON, CCP2CON**. (beállítás:él, osztás,PWM-mód) Bemenetei: CCP1, CCP2. Capture mód: CCPx lábon a szintváltozásra a TMR1 értékét eltárolja. Compare mód: a TMR1 értéke ha megegyezik egy eltárolt értékkel, akkor kiad egy jelet, és megszakítást generál. PWM-mód: változtatható kitöltési tényezőjű négyszögjelet ad. A periódusidőt a TMR2 periódusregisztere (PR2) adja Akitöltési tényezővel arányos számot a CCPR1L-be, és CCPR2L-be kell beírni. 10bitesnél a CCPxCON,4-5 az alsó 2 bit. 1bites DAC-ként használható, ha a kimenetre integráló tagot teszünk, vagy motorvezérlésre is jó.

●**SSP modul (szinkron soros busz):** I2C busz, és SPI busz. IC-k közötti kommunikáció. Az adatcsere regisztere az **SSPBUF** a beállítását pedig az **SSPCON**, az álapotot az **SSPSTAT** jelzi.

●**USART:**Szinkron, aszinkron soros port. Regiszterei: **RCSTA** (vezérlő, és státusz regiszter. Van ahol 2 másik van helyette), **TXREG** (adatkiküldés), **RCREG** (vett adat), **TXSTA** (adó beállítása), **SPBRG** (baud rate beállítás), **TRISC** (C portot használja, ezért itt a portláb irányát meg kell adni.), **PIE1** (itt kell engedélyezni az usart megszakítását, ami szerves része az adatforgalmazásnak).

●**EEPROM:** Lehet belső adat-eeprom. Konfigurálásuk: **EECON1, EECON2**. Az adatmemória 256*8bites, ezért a használatához 2 8bites reiszter kell: **EEADR**-a címnek, és **EEDATA** az adatnak. Elérhető a programmemória is a működés közben. Ennek a címe is, és a tartalma is szélesebb, ezért ahhoz kell még 2 segédregiszter:**EEADRH** (cím felső), **EEDATAH** (adat felső része).

Gyakorlati tudnivalók

Programfejlesztés MPLAB-bal:

Az MPLAB projekt rendszerben működik, ami azt jelenti, hogy a program együtt kezeli, és tárolja a fejlesztéshez tartozó fájlokat, pl forrásszöveg, lefordított hex-fájl. Ezek „összetartozását" a project fájl tartalmazza (.PJT). Ha több forrásfájl van (node), akkor azokat a **linker**-rel össze kell kapcsolni, így jön létre a végleges, beégethető HEX fájl. Általában 1 darab forrásfájlt használunk.

Első lépésben létrehozzuk az új ASM fájlt: **FILE>NEW**. Rákérdez, hogy akarunk-e hozzá új project fájlt az új ASM-hez. Nem akarunk. Elmentjük az asm-fájlt: **FILE>SAVE AS**, valami.ASM. Bezárjuk a fájlt. Ezután az új project létrehozása következik: **PROJECT MENÜ NEW PROJECT**. Nevet adunk neki, **OK**, megjelenik egy ablak. Abban az **ADD NODE** gomra kattintva ki kell választani az előzőleg létrehozott ASM fájlt, majd a processzortípust (kontrollertípust) kiválasztani a change gombbal. Itt lehet megválasztani a szimulációhoz az órajelet, a fejlesztés módját (más eszköz híján: MPLAB SIM simulator). **APPLY-OK-OK**… amíg el nem tőnnek a beállítási ablakok. Ezután **FILE>OPEN**: megnyitjuk az ASM fájlt. Elkezdjük írni a programot majd amikor úgy érezzük, hogy már lehet tesztelni, akkor **PROJECT>BUILD ALL** (ezzel lefordítjuk). Ezután lehet teszteni (erről később). Ha nem jó, (szinte biztos) akkor leállítjuk a tesztelést, és átszerkesztjük a programot. Ezután újabb fordítás, tesztelés, amíg jó nem lesz. Ha kész és jó, akkor elmentjük a projectet: **PROJECT>SAVE PROJECT**. Ezután elindítjuk a letöltő programot, kiválasztjuk benne a projrct_neve.HEX-fájlt, és betöltjük a PIC-be.

Tesztelés(DEBUG): Tesztelés kor a programot léptetjük, vagy futtatjuk, ha kell reseteljük (DEBUG menü, vagy a színes gombok használatával) A kontroller bármely regiszterének a pillanatnyi tartalmát folyamatosan figyelhetjük: **WINDOW>NEW WATCH WINDOW**, vagy szemüveges arc ikon. Itt megadjuk, hogy mely regisztereket akarjuk figyelni, és hogyan. A programfutás részidejeit lehet mérni a **WINDOW>STOPWATCH** segítségével. **WINDOW>MODIFY**: léptetések között át lehet írni egy-egy regiszter tartalmát. Breakpoint: Ha a programot folyamatosan futtatjuk, nem léptetjük (zöld lámpa indítja, piros leállítja), akkor bizonyos általunk megadott helyeken (jobb gombbal oda kattintunk a forrásszövegbe, ahova a töréspontot akarjuk) meg lehet akasztani a programot, ha odaér. Onnan tovább a zöld lámpa-gombbal megy. Stimulus: Lehetséges a portlábakra bemenő (csak digitális) jeleket adni, a **DEBUG MENÜ SIMULATOR STIMULUS**-pontában kiválasztott módon. Ennek 4 féle módja van: 1. Aszinkron stimulus, ekkor egy kis ablak jelenik meg, amin gombok vannak, ezeket lehet az egyes portlábakhoz rendelni, különböző módokon (on/off, impulzus, pzitív logika, negatív…) Ezt úgy tehetjük meg, hogy jobb gombbal rákattintunk a gombra). 2. Pin stimulus: Egy szövegfájlban adhatjuk meg, hogy melyik ciklus után milyen legyen a gerjesztés. 3. Clock stimulus: Lábhoz négyszögjelet rendelhetünk. 4.Register stimulus: Szövegfájlban megadott értéket kehet egy adott regiszterbe tölteni, adott helyen.

Programozás:

A gyári nevű regisztereknek is EQU-val kell nevet adni. (nevek nagy betűvel!) Ha előre konfigurált, letöltőt tartalmazó tokot használunk, akkor nem kell a __CONFIG a programba, és a programunk a 03h címről kezdődjön! Ha megszakítást használunk, akkor a z első utasításunkkal ugorjuk át a reset vektort, ami a következő, 04h címen van. A monitorprogram a 4. lapon, és az első 3 sorban van. Az END az egész program után kell, nem csak a főprogram után. Ha egy regiszterbe írni akarunk, de a szimulátor azt mutatja, hogy mégsem történik írás, akkor az azért van, mert rossz bankban vagyunk, váltani kell! Az adtlapon megadott ramméret csak a RAM-ra vonatkozik, nem az egész regisztерfájlra, tehát ahol csak 128 bájt van benne, ott is van 4db bank. Taszkkezelés: Ha több különböző funkciójú programrészt akarunk egyszerre futtatni, akkor érdemes a taszkkezelést alkalmazni, ami azt jelenti, hogy van egy ütemező program, ami az egyes funkciókat ellátó részek működését meghatározott időszeletekre engedélyezi, beleértve azok főprogram, és megszakítási-részeit is. Hogy a programban használhassuk a gyári neveket: INCLUDE "P16F877.INC"-kell! címke után utasítás álljon, ne direktíva! Számmegadás: default: decimális, hexa: 0x5B vagy 55H, bináris: B'01010011' A bitek sorszáma: „a 0. bit az első, és a 7. az utolsó!!!!!" Szubrutin elején célszerű elmenteni a W, és a STATUS tartalmát. A szubrutin egymésbaágyazásnak man maximált értéke: 8 szint a pic16F87x-nél.

Néhány alap programrészlet (rutin):

Megszakítási szubrutin, és ugrási tábla:

```
        Org  04
        Goto szubr   ; ez akkor jó, ha csak
        …            ;a B porton jönnek
szubr:  btfss  rb,0  ;0. bit ellenőrzése
        goto  egy    ;nem a 0. volt, tovább
        goto  int0   ;a 0. volt, kiszolgálás
egy:    btfss  rb,1  ;1. bit ellenőrzése
        goto  ketto  ;nem az 1. volt, tovább
        goto  int1   ;az 1. volt, kiszolgálás
ketto:  btfss  rb,2
        …
hat:    btfsc  rb,6
        goto  int6   ;a  6. nem 0! A 6. volt
        goto  int7   ;a 6.=0 nem a 6. volt,
                     ;így  csak a 7. lehet
Kiszolgálás végén CLRF  RB, és retfie kell!
```

Késleltetés szoftverrel

```
Kesl1:  movwf  ment
        Movlw  01
Vissza1:addlw  01
        Nop
        Nop
        Bnz    vissza1
;ide lehet ujabb keslel-
;tető hívást (más nevűt)
        Movfw  ment
        return
```

Késleltetés tmr0-val

```
Ke2:movfw opt_reg
    Movwf  ment1  ;ment
    Movlw  0c7
    Movwf opt_reg
    movfw intcon
    Movwf  ment2
    Movlw  0a0
    Clrf tmr0
    Movwf intcon
    Clrf  al
Visz:movf  al,0
    Addlw  0
    Bz  visz
    Movfw ment1
    movwf  opt_reg
    movfw  ment2
    movwf  intcon
    return
;ha tulcsordul, akkor subrut. Hív
 subr: movlw 10
       movwf  al
       retfie
```

```
movlw B'01000001'  ;a/d inic
movwf ADCON0
banksel ADCON1
clrf ADCON1
bsf ADCON1,7
```

2 bcd számjegy
1 bájtba (a,b>b)

```
  swapf  a,0
  addwf  b,1
```

IF(A=B) {…}

```
Movf  a,0
Subwf  b,0
Bz  igaz
```

IF(A>B) {…}

```
movf A,0
subwf B,0
btfss STATUS,C
goto IGAZ
```

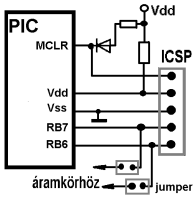
BINÁRISBÓL(PACKED)BCD 1 bájtból másfélbájt (a fél a százas)

```
BINBCD: CLRF   HUNDREDS  ;SZÁZASOK
        SWAPF  BIN, W
        ADDWF  BIN, W
        ANDLW  00001111B
        SKPNDC
        ADDLW  0X16
        SKPNDC
        ADDLW  0X06
        ADDLW  0X06
        SKPDC
        ADDLW  -0X06
        BTFSC  BIN,4
        ADDLW  0X16 - 1 + 0X6
        SKPDC
        ADDLW  -0X06
        BTFSC  BIN, 5
        ADDLW  0X30
        ADDLW  0X60
        BTFSC  BIN, 7
        ADDLW  0X20
        ADDLW  0X60
        RLF    HUNDREDS, F
        BTFSS  HUNDREDS, 0
        ADDLW  -0X60
        MOVWF  TENS_AND_ONES    ;EGYESEK,T.
        BTFSC  BIN,7
        INCF   HUNDREDS, F
```

Az MPLAB-bal való fejlesztés után betölthetjük a programot a PIC-be, vagy az ICD-modulba. Otthoni betöltés: LDRKEY-jel.

LDRKEY: Ehhez előre konfigurált tok kell. PIC16F87x FLASH-EEPROM-os mikrovezérlők programozására használható. Ha nem jó a program, egyszerűen új programot tudunk tölteni a helyére az LDRKEY-jel. Ha a RESET alatt a nyomógombot nyomva tartjuk,majd először a RESET gombot, majd azután a nyomógombot engedjük el, a led kétszeri felvillanása jelzi, hogy lehetséges a programot a PIC-be tölteni. Ezután elkezdődik a betöltés. Ha sikeres volt a betöltés, akkor a program elindul. Hiba esetén a LED folyamatosan villog, amit az újbóli betöltés kezdete állít meg. A letöltő a PC soros portjára csatlakozik. A letöltéskor a PIC-nek már a saját környezetében kell lennie!

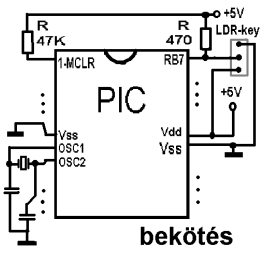
Használata: A letöltőt csatlakoztassuk a kontroller lábához. Bekapcsoláskor (az egész áramkör **bekapcsolásakor) nyomva kell tartani a gombot, majd felengedni.** Ekkor a LED kettőt villan, majd el kell indítani a letöltést a számítógépen. HA a LED **HÁRMAT VILLAN**, akkor kész a letöltés. Ha folyamatosan világít a LED, akkor hibás a letöltés, újra kell kezdeni. Bekötés: a letöltőnek 3 kivezetése van. A középsőt a PIC **RB7**-jelű kivezetéséhez kell csatlakoztatni, a másik kettőt pedig a **KÉT TÁPVONALRA**: Az a kivezetés, amelyik a 4,7KΩ-os ellenálláshoz csatlakozik, az megy a +tápvonalra, a maradék a földre. A letöltő a PC soros portját használja. A letöltő által használt, a PIC-be előre beégetett monitorprogram a programmemória első 3 bájtját, és utolsó 256 bájtját foglalja el, tehát a programunkban ezeket a területeket ne használjuk! A programunk a 03h címtől kezdődjön!



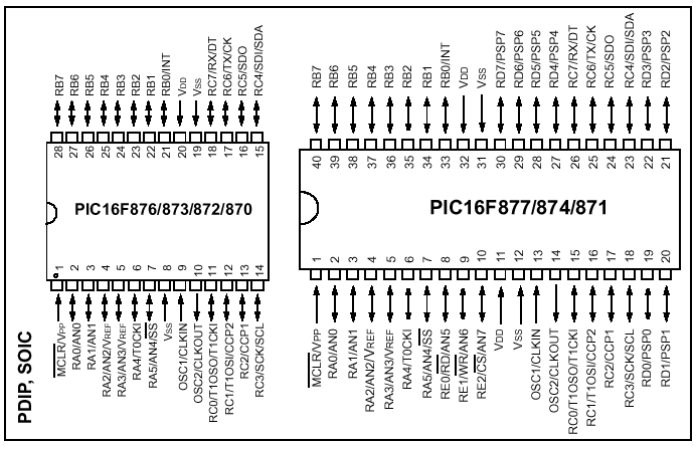
ICSP: (in circuit serial programming) Ehhez nek kell előre beégetett betöltőprogram, viszont drága programozó eszköz igen. Ennek van 1 5 pólusú csatlakozója, amit az alkalmazás paneljére kell csatlakoztatni. Ilyenlor kell már a _CONFIG a programba, és a program is a 00h címtől kezdődjön. A programozás és a normál működés elválasztásához jumper is kell az áramkörbe.

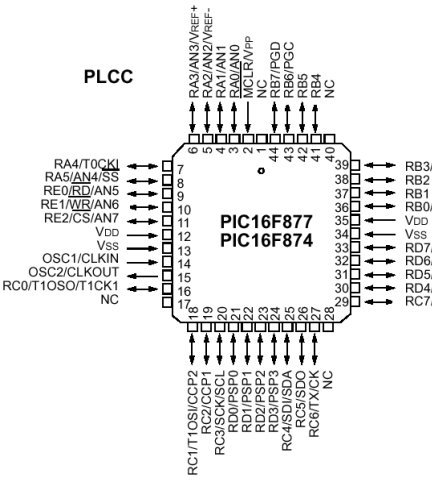
BEKÖTÉS: A **MCLR** -lábat fel kell kötni a + tápra egy 47K-s ellenálláson keresztül az áramkörben. Ezen kívül be kell kötni az **OSC1**, és **OSC2** lábakat, valamint pozitív feszültség kerüljön, de le lehessen húzni azt a letöltő nyomógombjával, (tehát legyen felhúzó ellenállás, és bemenet választó kapcsoló, vagy jumper) mert ez jelenti azt, hogy normál működés van. (Vss=gnd Vdd=+Utáp. Vpp=programozó fesz (12V tok prog., nem kell az ldrkey-hez). Vref-analóg bemenet, ezzel adjuk meg, az AN bem. Maximális értékét) A digitális bemenetekre kell pár száz ohmos soros ellenállás, az analóg bemenetekre 5-10kΩ-os ellenállás kell.

Kondenzátorok értéke		
mode	f	C
LP	32KHz	33pF
	200KHz	15 pF
XT	200K	47-56p
	1MHz	15 pF
	4MHz	15 pF
HS	4 MHz	15 pF
	8 MHz	15 pF
	20 MHz	33 pF



Néhány PIC bekötése:





Készítette: Nagy István (buenos@freemail.hu)