Ajánlott összefoglaló PIC kezdőknek

Az összefoglaló Istvánfi Béla és Braun Gábor munkája, kiegészítve a ChipCAD levlista tagjainak észrevételeivel.

Ahhoz, hogy a mikrovezérlőd működjön, az alábbiakra lesz szükséged::

1. Mindenekelőtt némi **olvasnivaló:**
a PIC katalógusa, a 16Cxx Reference Manual és az Application Note-ok. Ezek mind rajta vannak a Microchip CD-n, angolul. Magyar nyelven is létezik némi szakirodalom, pl. Tietze – Schenk: Analóg és Digitális áramkörök c. könyve, ebből is a Mikroszámítógépek c. fejezet. Ez nem a PIC-ekről szól, hanem általánosságban tisztáz néhány alapfogalmat. A PIC-ekről is van némi magyar nyelvű irodalom, ezek a ChipCad Kft-nél beszerezhetőek, pár száz forintért. A CD ingyenes. Ha ezeket áttanulmányozod, ismerni fogod a PIC mikrovezérlők lelkivilágát. Alább pár mondatban összefoglaljuk a lényeget:

A PIC mikrovezérlő család sokféle elemet tartalmaz, ezeknek rengeteg közös tulajdonsága van. Minden PIC-et hasonló, majdnem azonos nyelven lehet programozni. A különbség abban rejlik, hogy különféle PIC-ek más-más integrált hardver eszközöket tartalmaznak, mint például beépített A/D konverter, EEPROM adatmemória, stb.

Minden PIC mikrovezerlő beépítve tartalmaz egy RISC CPU-t, programmemóriát amely lehet PROM, EPROM, EEPROM, RAM adatmemóriát, 1 db 8 bites hardveres számlálót, watchdog és resetáramkört, valamint oszcillátort. A 16C5x-es család után megjelent kontrollerek tartalmaznak megszakításvezérlőt, az újabbak pedig beépített brown-out reset áramkört is. A fentiekkel bizonyára tisztában vagy, de azért foglaljuk össze a dolgokat:

RISC CPU: Reduced Instruction Set Central Processing Unit, azaz csökkentett utasításkészletű CPU. Valamikor a számítástechnikai őskorban két irányba szakadt szét a CPU-k fejlesztése. Az egyik irány, hogy minel több komplex utasítást, - akár szorzást is - valósítson meg a processzor (ez volt a CISC, Complex Instuction Set Cpu), a másik pedig, hogy kevés, egyszerű utasítást tudjon, de azt gyorsan. Ez lett a RISC. RISC esetében a bonyolultabb műveleteket (például szorzás) programból kell megcsinálni. Mivel a programok csupán 2%-ának van szüksége szorzásra, így nem baj, hogy nem kell minden esetben megfizetni a hardverbe integrált szorzót... Így lett a RISC egyszerű, ezért olcsó, és gyors.

A PIC fizikai felépítése lényegesen eltér például az INTEL típusú mikroprocesszoroktól (pl 80286, vagy a Motorola 68HC05-ös mikrovezérlő családja), mert a PIC-nek külön van a programmemóriája és külön az adatmemóriája. Ezt a konstrukciót Harward architektúrának hívják, szemben a 286-osok és HC05-ösök von Neumann architektúrájával, (Neumann János tiszteletére hívják így) ahol a program és az adatok ugyanannak a memóriának különböző címein találhatóak.

Programmemória: az a memória, amiben a végrehajtandó program van. Ez egy kissé fura dolog a RISC procikon, ugyanis ott 1 utasítás az 1 programhelyet foglal el, szemben pld. a 80x86-os procikkal, ahol 15 byte = 1 utasítás a 'csúcs'. Nos, a furcsaság oka az, hogy a programmemória nem 8, hanem 12, vagy 14, vagy 16 bites szervezésű. Így ez esetben nem program byte-okról, hanem program word-ökről beszélünk. 1 RISC utasítás = 1 'word'. Ha egy PIC-nek van 2048 word memóriája, abba 2048 utasításnyi programot tehetsz. Fizikailag minden PIC-nek van 'ablaktalan', egyszer programozható (vagy OTP = One Time Programmable) verziója, néhánynak létezik kvarcablakos, EPROMos kivitele. A kettő között csak a tokozás jelent különbséget.

A 16C84 volt az első olyan PIC típus, ami EEPROM memóriával került forgalomba, így olcsó volt (mert nem kell rá kvarcablak), és mégis könnyen újraprogramozható. Manapság már sokféle EEPROM-os PIC van. (Apróság: a 16C84 utáni összes EEPROM-os típusnak Flash EEPROM memóriája van.) A programmemória tipikus mérete 512-1024-2048-4096 word.

Adatmemória: RAM memória, melyben a program átmenetileg (amíg be van kapcsolva a PIC)

tárolhat adatokat, változókat, stb. A RAM 8 bit szélességű, byte szervezésű. A RAM memória első kb. 12 - 32 byte-ja 'foglalt' (ez típustól függ), utána típustól függően 40-50 byte memória áll rendelkezésre. Az első foglalt 12 - 32 byte-ban a PIC perifériáit (időzítő, számláló, A/D konverter) lehet elérni, úgy, mintha a memóriában 'látszanának'

Hardveres számláló / időzítő: egy 8 bites számláló, ami a következőket tudja:

- Az RA.4 lábon beállítható jelszintváltozásra növekedik az értéke, vagy
- a PIC órajele/4 frekvenciával növekszik az értéke

Mindkét esetben rendelkezésre áll egy beépített hardveres osztó, ami 2..256-os frekiosztást tud, kiválaszthatóan. Ez praktikus, ha viszonylag nagyobb frekvenciákat kell megmérni. Ha a számláló értékét az órajel növeli, akkor időzítőt kapsz..

Watchdog: 'Őrző-védő kutya', egy kis, független áramkör, ami a PIC-et reseteli, ha esetleg a benne futó program valami programhiba, de sokkal inkább tápzavar, ipari környezet, stb. hatására 'elszáll'. Ez úgy működik, hogy a watchdog egy kb. 20 ms-os időzítő, ami elindul. Ha 20 ms alatt nem történt CLRWDT utasítás a programban (a program 'lefagyott'), akkor reseteli a processzort. Ha megvolt a CLRWDT, akkor újraindul a 20 ms. A fentebb említett előosztóval a 20ms-os watchdog timeout értékét megnövelheted, max. 128-szorosára.

Reset áramkör: A tápfeszültség megjelenésekor automatikusan reseteli a PIC-et. (Régen, még a Z80, 8051-es időkben erre egy külön RC tagot kellett kívülről odaragasztani...) Néhány típusban van egy PWRT = PoWer staRtup Timer, ami a 'megfelelőnek' vélt reset után 200ms-ig még resetben tartja a PIC-et, a biztonság kedvéért. Nem szériatartozék, csak az újakban van: a brown-out reset áramkör. Ez akkor nagyszerű, ha a tápfeszültség fura módon piciket változik. Pl.: tipikusan el szoktam követni azt a hibát, hogy dióda, nagyElkó, stabkocka... Ez esetben a táp megszűntekor a PIC-en a táp lassan fog megszűnni, mert az elkó tartja a tápfeszt. A táp lassan lemászik 3V-ig, aztán valaki megint bekapcsolja... és ekkor a normál resetáramkör nem mindig / nem jól resetel. A brown-out reset ezt küszöböli ki. (Ha nincs benne BO reset, akkor külső resetkockával kell(ene) próbálkozni.)

Oszcillátor: A PIC-re kerámia rezonátort, kvarcot, RC oszcillátort, alacsony frekis kvarcot tudsz rátenni, az oszcillátor többi része a tokba van integrálva. Te mondod meg, hogy mid van, és az oszcillátor áramkör a tokon belül úgy rendezi el magát, hogy az adott konfiguráció működjön. A programod elején tudod meghatározni, hogy miféle oszcillátorod van. Az MPLAB help-ben olvasd el a __CONFIG direktívát. Ide kívánkozik a power management is: egy SLEEP nevű utasítás a PIC-et alvó állapotba helyezi: ekkor az oszcillátor megáll, és az áramfelvétel pár mA-re csökken a tetemes tipikus 4mA-ről. Az alvásból reset, watchdog timeout, vagy valami megszakítás tudja felébreszteni a PIC-et.

Megszakítás vezérlő: A 'megszakítás' azt jelenti, hogy a program normális, szekvenciális futása valamilyen külső hatás miatt átmenetileg felfüggesztődik, és a vezérlést egy külön rutin, a megszakításkezelő kapja meg. Miután a megszakításkezelő végzett, a program a futását - mintha semmi se történt volna - folytatja.

A külső hatások tipikusan: jelváltozás valamelyik bemeneten, a hardver számláló / időzítő 255-ről 0-ra váltása (túlcsordulása), vagy pld. az A/D konverter jelzése, hogy készen van a konverzió.

A megszakítás vezérlő feladata, hogy:
- legyenek megszakítások
- egyesével, vagy globálisan lehessen a megszakításokat engedélyezni, vagy letiltani.Kezdetben valószínűleg nem fogsz a megszakításokkal foglalkozni, de hamar elérkezik ennek is az ideje. Ha nem kell az interrupt, gondoskodj arról, hogy az INTCON regiszter 00-ban legyen. Ezzel minden megszakítást letiltottál.

2. Egy **mikrovezérlő,** amiben könnyen tudod cserélni a programot. Ez lehet egy ablakos eszköz is, de ez kicsit macerás, mert UV fénnyel (germicid lámpa ajánlott, és vigyázz, a fénye ne jusson a szemedbe !) kell törölni. Az ablakos eszközöknél kicsit olcsóbb a Flash memóriás PIC, ezt anélkül tudod újraprogramozni, hogy előtte fénnyel kellene törölnöd. Kezdőknek a PIC 16 F 84-et ajánljuk, ez egy 18 lábú IC, újraprogramozható és nem túl drága.

3. A mikrovezérlőt fel kell programozni, tehát kell egy **program**. Ezt Te fogod megírni. Kezdetnek azt javasoljuk, hogy az egyik I/O port valamelyik kivezetése legyen kimenet, és ennek segítségével villogtass egy LED-et. Aránylag egyszerű feladat, néhány sornyi programmal megoldható, mégis elegendő arra, hogy a programírás alapszabályait megtanuld. Ha valami nem
megy, nyugodtan kérj segítséget a ChipCad listán, de arra kérünk, hogy NE azt kérd, hogy valaki írjon neked egy ilyen - olyan programot. HANEM pl.: "itt ez a pár soros program, én írtam, és nem akar működni. Segítsetek már, mi lehet rossz! " és itt következik a néhány soros kis program, amit elkövettél. Így sokkal könnyebb segíteni (tehát hamarabb kapsz választ), és Te is többre jutsz vele.

4. A program megírásához kell egy **fejlesztőkörnyezet**. Ez ingyenes, MPLAB-nak hívják, letölthető a www.microchip.com -ról, vagy a Microchip CD-ről. A program 3.1-es vagy annál újabb Windows alatt fut. Van benne szövegszerkesztő, assembler (ez fordítja le a szöveges utasításokat gépi kódra), szimulátor, amivel ellenőrizheted, hogy a programod mit csinál. Más fejlesztőkörnyezet is létezik, pl a Parallax féle, és DOS alatt is lehet programot írni PIC-re, csak nem olyan kényelmes.

5. Ha megírtad **a programot**, akkor azt valahogyan **bele kell töltened a PIC-be**, ezt a folyamatot a zsargonban égetésnek hívjuk. Külön készüléket igényel, ami össze van kötve a PC-ddel, bele tudod tenni a PIC-et és felprogramozod. Többféle ilyen készülék van forgalomban, de magad is építhetsz egyet. A doksikat az Internetről össze lehet szedni hozzá.

6. A felprogramozott (beégetett) PIC-et **ki is kell próbálni**, ehhez egy kis hardvert kell építened. Ha elfogadod a 2. pontbéli LED villogtatást, akkor egy csupalyuk próbapanelre tégy egy 18 lábú IC foglalatot (hogy a PIC-et ki tudd venni programozás céljából), valamelyik I/O portra soros 220 -500 ohm-on keresztül köss egy LED-et. A PIC oszcillátora legolcsóbb esetben egy RC tag, a katalógusból ki tudod olvasni az értékeket. Kell még neki tápfeszültség: legyen mondjuk 5V, fontos, hogy stabilizált legyen! Ha ez nincs, egy 4,5V-os zseblámpaelem is megteszi. (A stabilizált tápfesz azért kell, mert zajos, brummos feszültségről nem fog jól működni a mikrovezérlő. A szárazelem nem zajos, tehát megfelel.) A hozzávalókat elektronikai alkatrészboltokban megkapod.

7. Kell némi kitartás is. Ne add fel az első kudarc után!

8. A PIC eléggé jól sikerült mikrovezérlő, stabilan működik és tartós, de ehhez néhány ökölszabályt be kell tartanod. Például:

- ne adj rá soha fordított tápfeszültséget!
- ne terheld túl a kimeneteket! (nincs rövidzárvédelem a kimeneteken!)
- a bemeneteket határozott potenciálra (0-ra vagy 1-re) kösd !
- vedd figyelembe azt, hogy bekapcsolás után a PIC minden I/O portja bemenet, és a programból kapcsolod át azokat kimenetté. Ha lóg a levegőben valamelyik kivezetés, az, amíg bemenet, összeszedhet zajokat, és lehet, hogy ettől rosszul működik majd az áramköröd. Határozott potenciálra viszont csak ellenálláson keresztül kösd (kb 500 ohm), mert ha a programban kimenetként állítasz be egy I/O vonalat, és 0-ba programozod, miközben az a tápfeszre van kötve, akkor tönkremehet a vezérlőd. Ezeket az ellenállásokat azért nem rajzoltuk rá az ábrákra, hogy a rajzok egyszerűek legyenek, csak a lényeget mutassák.

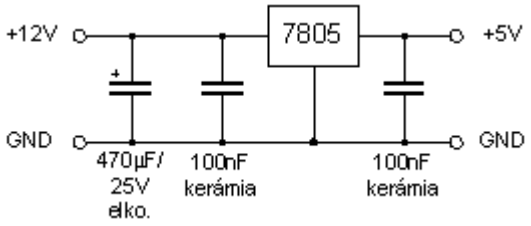
9. Ha mégse működik, az alábbiakat ellenőrizd:

- tápfesz van? megfelelő lábakra van kötve? A + és a - is be van kötve?
- oszcillátor jól van bekötve? Rezeg? (ezt tutira csak oszcilloszkóppal tudod megnézni)

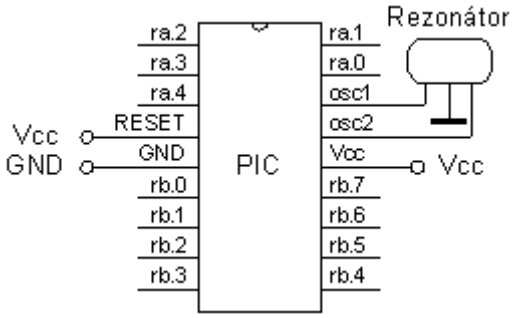
- arra a lábra kötötted a LED-et, amit a program kezel? Nincs-e fordítva a LED?
- Ha világít, de nem villog: nem túl gyors-e a villogás frekvenciája ahhoz, hogy szemmel is lásd?
- bemeneteket nem hagytad-e a levegőben lógni? Különösen a MCLR bemenet érzékeny erre.

10. Végül néhány alapvető ábra, segédanyag a gyakorlati felhasználáshoz :

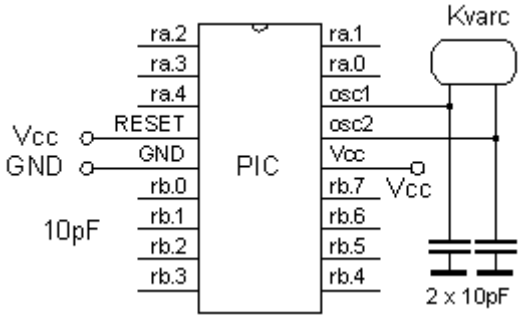
1. ábra : ajánlott táplálás



2a. ábra : alap bekötés 4MHz-es kerámia rezonátorral

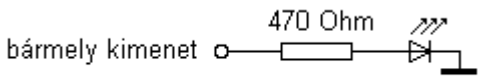


2b. ábra : alap bekötés kvarccal

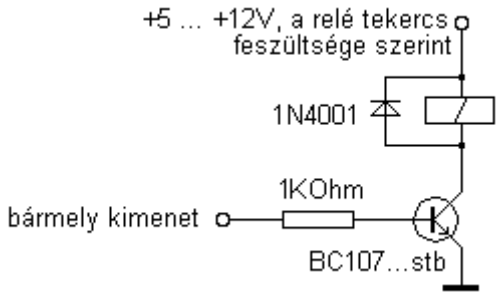


3. KIMENETEK

3a. ábra : LED a kimenetre

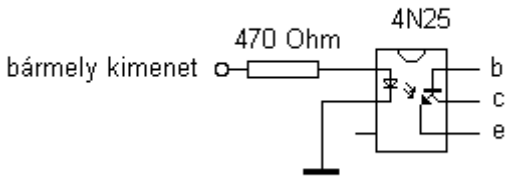


3b. ábra : relé a kimenetre



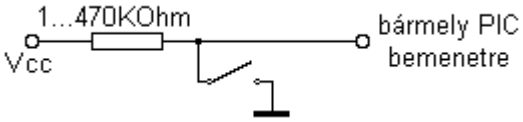
A relé tekercsével párhuzamosan kötött dióda alapvető fontosságú! Ne hadgyd ki! Hasonló módon tudsz kapcsolni bármely picike fogyasztót, lámpát, stb. Ne köss azért rá 230V-ot.. Ha induktiv fogyasztót használsz (tekercs, relé, motor), akkor a dióda mindenképpen FONTOS!

3c. ábra : Optocsatoló a kimenetre:

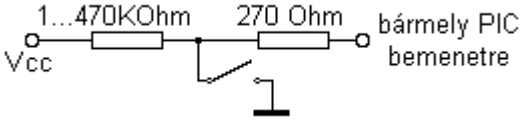


4. BEMENETEK

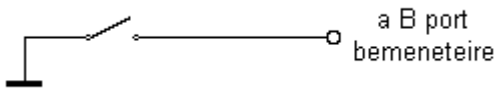
4a. egyszerű gombok:



4b. ábra : az előző gomb “bolondbiztos” változata, ami minden lehetséges hiba ellen védett



4c. ábra : vagy gomb egyszerűbben, kihasználva az RB porton beépitett felhuzóellenállásokat:



Ez a táblázat olyan, "különleges" assembly utasításokat tartalmaz, amit az MPLAB elfogad 16C5x és 16Cxx processzorokhoz. Fordításkor automatikusan helyettesíti ezeket az utasításokat az "Egyenértékű művelet" oszlopban levő, eredeti assembly kódokkal.

Mnemonic	Description	Leírás	Egyenértékű művelet	Státuszbit
ADDCF f,d	Add Carry to file	Carry-t hozzáadja egy regiszterhez	BTFSC 3,0 INCF f,d	Z
ADDDCF f,d	Add Digit Carry to file	Digit Carry-t hozzáadja egy regiszterhez	BTFSC 3,1 INCF f,d	Z
B k	Branch	Ugrás (Branch)	GOTO k	-
BC k	Branch on Carry	Ugrás, ha Carry=1	BTFSC 3,0 GOTO k	-
BDC k	Branch on Digit Carry	Ugrás, ha Digit Carry=1	BTFSC 3,1 GOTO k	-
BNC	Branch on No Carry	Ugrás, ha Carry=0	BTFSS 3,0 GOTO k	-
BNDC	Branch on No Digit Carry	Ugrás, ha Digit Carry=0	BTFSS 3,1 GOTO k	-
BNZ	Branch on No Zero	Ugrás, ha nem 0	BTFSS 3,2 GOTO k	-
BZ	Branch on Zero	Ugrás, ha 0	BTFSC 3,2 GOTO k	-
CLRC	Clear Carry	Carry törlése	BCF 3,0	-
CLRDC	Clear Digit Carry	Digit Carry törlése	BCF 3,1	-
CLRZ	Clear Zero	Zero bit törlése	BCF 3,2	-
LCALL	Long Call	Távoli hívás	BSF/BCF 0A,3 BSF/BCF 0A,4 CALL k	-
LGOTO	Long Goto	Távoli ugrás	BSF/BCF 0A,3 BSF/BCF 0A,4 GOTO k	-
MOVFW f	Move File to W	Az f regisztert W-be teszi	MOVF f,0	Z
NEGF f	Negate File	Az f regisztert negálja	COMF f,1 INCF f,d	Z
SETC	Set Carry	Carryt 1-be állítja	BSF 3,0	-
SETDC	Set Digit Carry	Digit Carryt 1-be állítja	BSF 3,1	-
SETZ	Set Zero	Zero bitet 1-be állítja	BSF 3,2	-
SKPC	Skip on Carry	Ha Carry=1, a következő utasítást átugorja	BTFSS 3,0	-
SKPDC	Skip on Digit Carry	HA Digit Carry=1, a következő utasítást átugorja	BTFSS 3,1	-
SKPNC	Skip on No Carry	Ha Carry=0, a következő utasítást átugorja	BTFSC 3,0	-
SKPNDC	Skip on No Digit Carry	Ha Digit Carry=0, a következő utasítást átugorja	BTFSC 3,1	-
SKPNZ	Skip on Non Zero	Ha a Zero bit=0, a következő utasítást átugorja	BTFSC 3,2	-
SKPZ	Skip on Zero	Ha a Zero bit=1, a következő utasítást átugorja	BTFSS 3,2	-
SUBCF f,d	Subtract Carry from File	Carry bitet kivonja egy regiszterből	BTFSC 3,0 DECF f,d	Z
SUBDCF f,d	Subtract Digit Carry from File	A Digit Carry bitet kivonja egy regiszterből	BTFSC 3,1 DECF f,d	Z
TSTF f	Test File	Regiszter ellenőrzése	MOVF f,1	Z