EÖTVÖS LORÁND TUDOMÁNYEGYETEM

DIPLOMAMUNKA

TANÁRI KÉPZÉS SZAKDOLGOZATA

Mikrovezérlők oktatási célú alkalmazása

Készítette:
Ocskó Szabina
Informatika – technika tanár
ELTE

Témavezető:
Dr. Bárkai János
adjunktus
BME

Belső konzulens:
Dr. Schiller István
adjunktus
ELTE

Budapest, 2011

Tartalomjegyzék

Tartalomjegyzék 2
Bevezetés 4
Szakdolgozatom előzményei és a témaválasztás pedagógiai szempontjai 5
Történeti áttekintés 6
A mikrovezérlők világa 6
Minden így kezdődött… 6
Mikroprocesszor kontra mikrovezérlő 10
Beágyazott rendszerek 11
A mikrovezérlő és a PC közötti különbségek 12
A mikrovezérlő komponensei 12
A PIC egyes részei: 12
Programmemória 13
Fájlregiszterek 13
I/ O portok 13
A/ D konverterek 13
Soros port 13
Időzítők 13
Egyéb perifériák 13
A Timer modul 14
Capture és Compare modul (Kiolvasó és összehasonlító modul) 14
Soros interfész 15
Az EUSART használata 16
Aszinkron mód 16
Aszinkron módú adat küldés a PIC Tx lábán 16
Aszinkron módú adat fogadás a PIC Rx lábán 16
Szinkron mód 16
Az MSSP (Master Synchronous Serial Port) használata 17
SPI üzemmód 17
I²C üzemmód 18
Architektúra 19
Harvard architektúra 19
Az adatmemóriától eltérő szélességű utasításkód 19
Egyszavas utasítások 19
Az utasítások gyakran egyetlen belső ciklus alatt végrehajtódnak 19
Átfedéses utasítás végrehajtás 20
Csökkentett utasításkészlet 20
Számrendszerek 21
A számok világa 21
Bináris számrendszer 22
Hexadecimális számrendszer 23
BCD kód 24
Számrendszerek közti konverzió 24
Bináris szám decimálisra alakítása 24
Logikai áramkörök 25
ÉS kapu 25
VAGY kapu 26
NEM kapu 26

KIZÁRÓ VAGY kapu 27
Ismerkedés a PIC mikrovezérlőkkel 27
Kapcsolás megoldása relével 32
Kapcsolás megoldása bipoláris tranzisztorral 33
Kapcsolás megoldása FET-tel 34
A PIC mikrovezérlő különböző szériái 37
PIC 10F széria 37
PIC 12F széria 37
PIC14 széria 38
PIC16F széria 38
PIC18 széria 38
PIC24 széria 38
PIC-ek összehasonlító táblázata 39
Fejlesztőkörnyezet 40
MPLAB 40
Részei 40
PASM assembler 41
MPLINK linker 41
Hibakereső 41
Telepítés 41
Mintapéldák MPLAB-bal 42
Mintapélda PIC12F683-mal 43
Mintaprogram C nyelven 49
Az MPLAB SIM 56
A disassembly lista 63
A program módosítása 66
Megvalósítás PICBASIC nyelven 68
A PIC16F1939 használata 70
A PIC 18-a sorozat. 71
Minta Panelok 75
Léptetőmotor 80
A léptetőmotor működési elve vázlatosan 80
A léptetőmotor tekercseinek gyakorlati megvalósítása 83
A szervo motor 84
Nyomtatott áramkör és kapcsolási rajz tervezése 86
EAGLE (Easily Applicable Graphical Layout Editor) 86
A program bemutatása 88
Új alkatrész rajzolása 92
Egy kész nyomtatott áramkör 97
Összefoglalás 98
Irodalomjegyzék 1

Bevezetés

Másodéves hallgató koromban felvettem a Budapesti Műszaki és Gazdaságtudományi Egyetem (BME) Vegyészmérnöki és Biomérnöki Karon (VBK) vendéghallgatóként a számítástechnika III. című tárgy mikrovezérlőkkel foglalkozó kurzusát. Megtetszett a mikrovezérlők alkalmazása. A következő félévekben segítettem a tárgyat tartó tanárnak – későbbi témavezetőmnek – az órák tartásában. Ezeken az órákon tapasztaltam, hogy a hallgatók könnyen megtanulják az alapvető alkalmazásokat, ha egyszerűen magyarázzuk el nekik. Ezért döntöttem ilyen témájú diplomamunka megírása mellet. A Mikrovezérlők alkalmazásáról sokezer oldalnyi anyag áll rendelkezésre. Közöttük vannak olyanok is, amiket kezdőknek tanulásra szántak. Ezek egy-egy szűkebb témát igyekeznek részletesen bemutatni, többnyire egyetlen mikrovezérlő típusra. Sajnos döntő többségük a lefedett szűk témában is elavult, a bemutatott alkatrészek korszerűtlenek és drágák. Szűk témakörben elavult ismereteket drágán szerezni nem tűnik jó ötletnek. Diplomamunkám célja a jelenlegi legmodernebb eszközöket bemutatni olyan szinten, hogy az interneten fellelhető kézikönyvek, leírások érthetővé váljanak kezdő számára is. A Microchip PIC10-es szériájától kezdve a PIC-24-es szériáig terjedő mikrovezérlőket választottam, mert ezek kezdő számára is egyszerűen, olcsó eszközökkel használhatók. Természetesen nem lehetett célom a részletes bemutatás, mert akkor sokezer oldal terjedelmű lett volna a diplomamunkám. Inkább kezdő lökést, inspirációt kívántam adni és az irodalom tengerében eligazítással szolgálni. Be kívántam mutatni a kezdő számára ingyenesen elérhető szoftver eszközöket, mint az MPLAB fejlesztői környezet, valamint a PICBASIC és a C nyelv ingyenesen is használható fordítói. Célom volt néhány egyszerűen megvalósítható konkrét áramköri megoldást megadni mintegy első projektként. Befejezésként egy elkészíthető vagy legyártatható kísérleti panel tervezése és a panel tervezés lépéseinek bemutatása volt a célom.

Szakdolgozatom előzményei és a témaválasztás pedagógiai szempontjai

A látogatott órákon láttam, hogy a hallgatók számára a kezdő lépések a nehezek. Ha ezeken sikerül őket átsegítenünk, akkor utána már gördülékenyen megy. Úgy kell elmagyarázni, hogy a lehető legkevesebb elmélettel a lehető leggyorsabban eljussanak az első működő alkalmazásig, azaz az első sikerélményig. Azt tapasztaltam, hogy a nehezebben érthető részek részletes elmagyarázása helyett célszerűbb az mondani, hogy „ezt így írjuk be ide, majd később megtanuljuk miért". Amikor az első áramkörük működik, utána az érdeklődőbbek az egyes részleteknek önállóan is utána tudnak nézni. Témavezetőmmel többször beszélgetünk arról, hogy jó lenne az órák tapasztalatait írásba foglalni. Tehát vannak, akiket ez a téma érdekel, de nem tuják önállóan vagy kevés tanári segítséggel elkezdeni. Így jött az ötlet, hogy írjak belőle egy diplomamunkát. Első lépésben ki kellet választanom a „célközönséget". Úgy gondolom, hogy az egyszerűbb mikrovezérlők alkalmazása nemcsak a nem elektronika szakos egyetemistáknak, hanem a középiskolásoknak, esetleg az érdeklődőbb általános iskolásoknak is érthető és érdekes, hasznos téma lehet. Témavezetőm mesélte, hogy az ő gyermekkorában az elektronika iránt érdeklődők detektoros rádiót fabrikáltak. Ismereteiket folyamatosan bővítve, egyre bonyolultabb áramköröket készítettek. Jó és olcsó könyvekből, folyóiratokból folyamatos önképzéssel jutottak az ismeretek birtokába. Külső segítséget akkor kértek, ha szükség volt rá. A detektoros rádió építése ma már természetesen nem tartozik az érdekes témák közé. Viszont ugyanilyen érdekes téma lehetne napjainkban a mikrovezérlők alkalmazása. A mikrovezérlők olcsón kaphatók, a hozzá kapcsolódó irodalom bőséges. A világhálón rengeteg jó és ingyenes leírás található, de ezek összegyűjtése kezdő számára nehézkes. Az ismereteket összegyűjtötten tartalmazó irodalom drága és sokszor elavult. A magyarországi oktatásban jelentős szereplő Kónya László, mind a tényleges oktatásban[1], mind az ismertető irodalom írásában[2]. Kónya elsősorban a felsőfokú szakemberképzésben vesz részt. A középiskolás mikrovezérlő oktatásban kiváló tananyagot készített Juhász Róbert[3]. A külföldi irodalom is bőséges. A Microchip honlapján nagyon sok információ ingyen elérhető[4]. Magyarországon Chipcad[5] a Microchip disztribútora, gyakorlatilag minden megjelent áramkörük kapható.

Történeti áttekintés

A mikrovezérlők világa

A mikrovezérlők területén elért mai állapot az integrált áramkörök technológiájának fejlesztésével indult. Ez a fejlődés lehetővé tette több ezer tranzisztor tárolását egyetlen chipben. Ez volt az előfeltétele a mikroprocesszor gyártásának és az első, számítógépekhez való egységek - memória, bemeneti / kimeneti vonalak, időzítők, stb. - hozzáadhatóságának. Az áramköri sűrűség további növelése következtében létrejött egy olyan integrált áramkör, mely tartalmazza a processzort és a perifériákat is. Tehát létrejött az első egychipes mikroszámítógép, későbbi nevén mikrovezérlő.

Minden így kezdődött…

Az interneten szános helyen fellelhetők a mikroprocesszorok, mikrovezérlők fejlődését bemutató írások, ezek egyike Milan Verle: PIC Microcontrollers című könyve[6]. Én itt a leggyakoribb leírásokat foglaltam össze, kivonatoltam a fontosnak tartott dolgokat. 1969-ben a BUSICOM japán mérnökeinek egy csapata az USA-ba érkezett. A feladatuk az volt, hogy zsebszámológépekhez kellett integrált áramkört tervezniük. A kérést elküldték az INTEL vállalathoz, ahol Marcian Hoff volt a project felelőse. Mivel volt számítógépes tapasztalata a PDP8 terén, a javasolt terv helyett ő alapvetően eltérő ötlettel állt elő. Ez az ötlet arról szólt, hogy maga az integrált áramkör tárolja a programot. Ez azt jelenti, hogy maga a konfiguráció egyszerűsödött, de sokkal több memóriát igényelt, mint a japán mérnökök megoldása. Kis idő múlva, bár a japán mérnökök igyekeztek megtalálni egy egyszerűbb megoldást, Marcian ötlete nyert és megszületett az első mikroprocesszor. Hogy az ötletből késztermék legyen, az Intel a legnagyobb segítséget Federico Faggintól kapta. Kilenc hónappal az Intelhez érkezése után sikerült az eredeti koncepciójából kifejlesztenie egy terméket. 1971-ben az Intel megszerezte a jogot az integrált áramkör értékesítéséhez. Ezt megelőzően az Intel megvásárolta a licencet BUSICOM társaságtól, amelynek fogalma sem volt róla mekkora kincs ez és mekkora jövő elé néz. Még ebben az évben, egy 4004 névre keresztelt mikroprocesszor jelent meg a piacon. Ez az első mikroprocesszor 4 bites volt, 6000 művelet / másodperc sebességgel. Nem sokkal később az amerikai CTC cég

együttműködést ajánlott az Intelnek és a Texas Instrumentsnek, hogy 8-bites mikroprocesszorokat alkalmazzon a terminálokon. Annak ellenére, hogy a CTC végül feladta ezt a projektet, az Intel és a Texas Instruments tovább dolgozott a mikroprocesszoron és 1972 áprilisában megjelent a piacon az első 8 bites processzor, ami 8008 névre hallgatott. 16Kb memóriát tudott megcímezni. 45 utasítása volt és 300000 művelet/ másodperc sebességgel dolgozott. Ez a mikroprocesszor volt az elődje az összes mai mikroprocesszornak. Az Intel folyamatosan fejlesztett és 1974 áprilisában megjelentette a 8080 nevű 8 bites processzorát. 64 Kb-nyi memóriájával és 75 utasításával 360 $-ba került. Ez akkor nagy összegnek számított, mégis igen sokat sikerült eladni belőle.

Egy másik amerikai cég - a Motorola - hamar rájött mi történik a piacokon, így ő is megjelentette a 8-bites mikroprocesszorát, a 6800-at. A vezető tervező Chuck Peddle volt. Eltekintve a processzortól, a Motorola volt az első cég, amely egyéb perifériális célú áramköröket is előállított, mint pl. a 6820 és 6850. Akkoriban egyre több vállalat ismerte fel a mikroprocesszorok növekvő jelentőségét és elkezdte a saját fejlesztéseit.

Az 1975-ös USA-ban tartott WESCON kiállításon a mikroprocesszorok történelmének egy fontos eseménye kapott helyet. A MOS Technology bejelentette, hogy a 6501 és 6502-es processzorát 25 $-ért lehet megvásárolni, ami iránt a vevők azonnal érdeklődni kezdtek. Hatalmas szenzáció volt, mert sokan csalásnak hitték, mivel a konkurencia a 8080-at és a 6800-as mikroprocesszort még mindig 179$-ért adta. A kiállítás első napján, a versenytársak válaszul - mind a Motorola, mind az Intel - lecsökkentették a mikroprocesszoraik árát 69.95 $-ra. A Motorola megvádolta a MOS Technologyt és Chuck Pedde-t, hogy plagizálta a védett 6800-at. Ezért a MOS Technology felhagyott a 6501-es gyártásával, de a 6502- t továbbra gyártotta. Ez 8 bites processzor volt. 56 utasítással rendelkezett és képes volt 64Kb-nyi memória közvetlen címzésére. Az alacsony ár miatt a 6502 nagyon népszerű lett, így nagyon sok mikroszámítógépre ezt telepítették. Ilyen számítógépek voltak például KIM-1, Apple I, Apple II, Atari, Commodore, Acorn, Oric, Galeb, Orao, Ultra és sok más. Hamarosan kiderült, hogy több gyártó kezdte el gyártani a 6502-t (Rockwell, Sznertek, GTE, NCR, Ricoh, Commodore). A fellendülés évében, 1982-ben, 15 millió processzort adtak el. Alapvetően ez a három típus uralta a piacot. Kissé különböztek egymástól, mindnek volt a másikhoz képest előnye is, hátránya is.

Azonban mások sem akartak lemaradni, kimaradni. Frederico Faggin otthagyta az Intelt és saját céget alapított Zilog Inc. néven. 1976-ban bejelentette a Z80-at, Úgy döntött, hogy az új processzorának kompatibilisnek kell lennie a 8080-nal, azaz képes kellett legyen elvégezni az összes 8080-ra írt programot. Azon kívül, hogy sok más funkcióval is bővült, a Z80 volt a legerősebb mikroprocesszor abban az időben. Képes volt címezni 64 Kb memóriát, 176 utasítással bírt, néhány regiszterrel, frissíthető dinamikus RAM memóriával. Megelégedett egyetlen áramellátással. A Z80 nagy siker volt és szinte mindenki lecserélte a 8080-at Z80-ra. Hirtelen a Z80 lett kereskedelmi szempontból a legsikeresebb 8 bites mikroprocesszor abban a időben. A Zilog mellett más új gyártók is hamarosan megjelentek, mint a Mostek, NEC, SHARP és SGS. A Z80 volt a szíve sok számítógépnek, mint például: ZX80, ZX81, Spectrum, Partner, RS703, Z-3 és Galaxy.

1976-ban az Intel előállt egy továbbfejlesztett 8 bites processzorral, a 8085-tel. Később az Intel már a 8088 típusú mikroprocesszorait gyártotta. Sokak szerint nem ez volt a kor legjobb processzora. Mind a Motorola új 68000-sét, mind a Zilog Z8000-sét jobbnak tartották. A kisszámítógépek gyártására az egyre nagyobb titkolódzás volt a jellemző. A tanszéken ahol a diplomamunkámat készítettem, akkoriban volt egy Wang típusú asztali gép. Semmit nem árult el a belsejéről a gyártó cég. A tanszékiek csak a gép korszerűtlenné válása és leselejtezése után a bontás során tudták meg még a processzor típusát is.

Még mindig sokan váll-váll mellet haladtak a piacon. A nagyszámítógépes területen akkoriban az IBM döntő fölénye megkérdőjelezhetetlen volt. Úgy emlegették a céget, hogy a “Nagy Kék”. Jó minőségű, de nagyon drága rendszereket gyártott. A piaci fölényük kissé kényelmessé is tette őket. Nem ismerték fel időben a kisszámítógépek jövőbeni lehetőségét. Amikor felismerték már késő volt: a piacon annyian tolongtak különböző teljesítményű és áru asztali számítógépekkel, hogy új belépőként reménytelennek látszott betörni. Ekkor jött az IBM nagy ötlete: megjelentetett egy saját kisszámítógépet, teljesen nyílt architektúrával. Ez volt az IBM PC. Mindent leírtak róla: áramköri szintű kapcsolási rajzot, a ROM-ban lévő Basic fordító részleteit, stb. A siker hatalmas lett. A nyílt architektúra miatt nagyon sokan készítettek hozzá hardver egységeket és programokat egyaránt. Rövid idő alatt rengeteg PC-t adtak el. Persze nem véletlenül féltek mások közzétenni a gépeik belső titkait. Ez a másolásra lehetőséget nyújtott. Nem is váratott sokáig magára a klón PC megjelenése. Tajwani gyártók

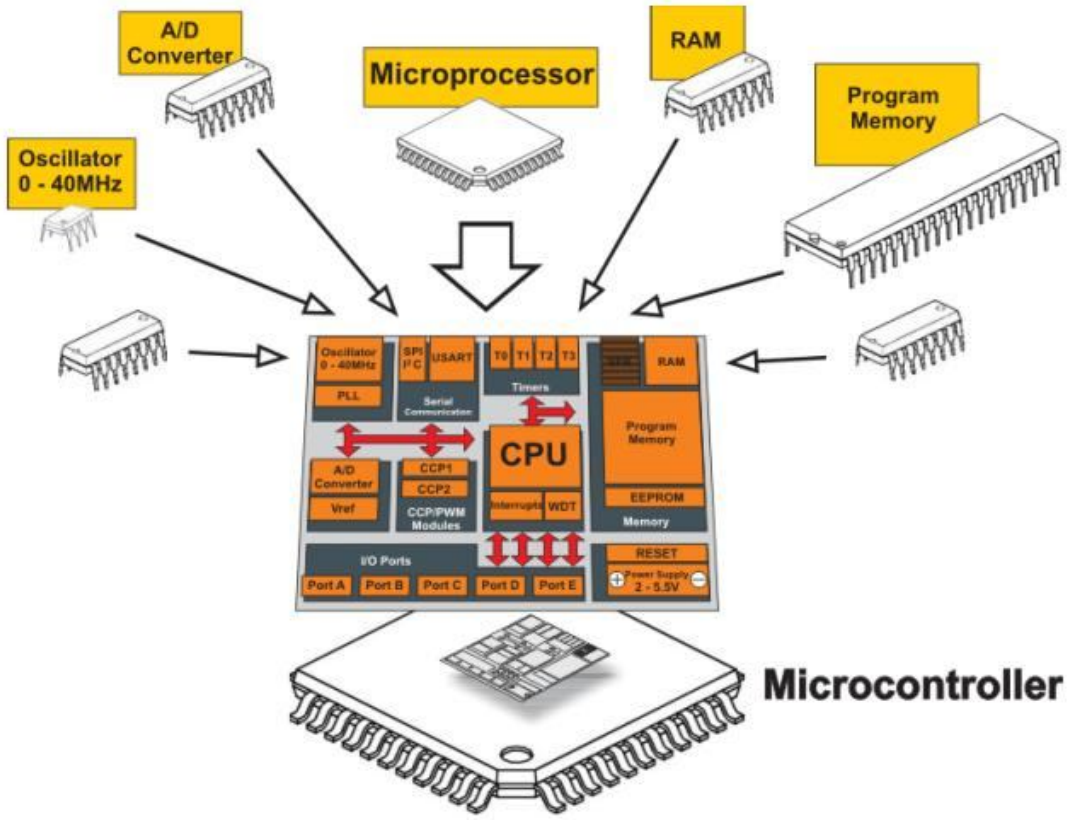
elkezdték az eredetivel teljesen megegyező alaplapok gyártását olcsó áron. Itt jött újból a piac fintora. Az olcsó másolat nem csökkentette az eredeti iránti keresletet, hanem egy ideig növelte: már azok is meg tudták venni, akiknek eredetire nem volt pénzük. Így egyre több ember ismerte meg, egyre több hardver és szoftver készült hozzá, tovább növelve a népszerűséget. Láttam olyan PC alaplapot, amit témavezetőm rakott össze annak idején: tajwani, profi színvonalon elkészített nyomtatott áramkör, amibe NDK-s és cirill betűs orosz IC-ket forrasztott. Érdekes látvány volt. Lelkes amatőrök Magyarországon is készítettek ezekről a nyomtatott áramkörökről némi javítgatás után használható másolatot. Hasonló jelenséget láthatunk napjainkban egyes szoftverek terén: az teszi ismertté és elterjedté, hogy létezik belőle törött verzió. Ezt feltelepítik, megismerik, megszeretik és a jobb támogatás érdekében idővel megveszik az eredetit. Jelenleg is van olyan szoftver, melynek többé- kevésbé jól működő törött verzióját híresztelések szerint maga a cég marketing igazgatója bocsátja útjára, az eredeti verzió megjelenése után bizonyos idővel. Ez persze valószínűleg nem így van, de mindenképpen mutatja, hogy az ismertség nagyon nagy piaci előnyt jelent. Ezt használta ki az IBM is. A PC további életútja mindenki előtt ismert.

Miért is tettünk ilyen kitérőt a mikroprocesszorok tárgyalása során? Azért mert az IBM ezen lépése meghatározó volt a mikroprocesszorok jövőjét illetően: az IBM nem a már akkor is jobbnak számító processzor típusok közül válogatott, hanem az Intel 8088-at választotta. Valószínűleg csak ők tudják miért. A PC hatalmas sikerének oldalvizén az Intel is megerősödött. A mai processzorai is kompatibilisek az eredeti 8080/8088 szériával. Ennek hátrányai is vannak, de az Intel a kompatibilitást piaci okokból megtartó stratégiája sikeresnek bizonyult. Más – számos esetben jobb, logikusabb – processzorok és a vevők kárára. A mikroprocesszorok mellet viszont fejlődésnek indultak a mikrovezérlők. Tekintsük át a mikroprocesszorok és a mikrovezérlők jellemzőit!

Mikroprocesszor kontra mikrovezérlő

Mikroprocesszor mikroprocesszortól több mindenben különbözik. Az első és legfontosabb különbség a funkcionalitás. Annak érdekében, hogy a mikroprocesszor használható legyen, más részegységeket is kell hozzáadni. Ilyen pl. a memória vagy az adatátvitelt segítő lehetőségek. Habár a mikroprocesszorok mini számítógépnek számítanak, a gyenge pontjuk az, hogy a perifériakezelő részük nincs eléggé kidolgozva.

Arról van szó, hogy a kommunikálás érdekében a mikroprocesszoroknak külső chipen lévő speciális áramkört kell használniuk. (1. ábra)



1. ábra[6]

Viszont a mikrovezérlőket éppen úgy tervezték, hogy minden egyben legyen benne. Nincs más külső komponens, ami más alkalmazásokhoz kellene. Minden szükséges áramkör, amik egyébként a perifériákhoz tartoznak, bele van integrálva. Így egy alkalmazás tervezésénél időráfordítást nyerünk és helyet takaríthatunk meg.

Beágyazott rendszerek

A beágyazott rendszerek egyetlen chip-en elhelyezett mikroprocesszorból, perifériákból és egyéb kiegészítő áramkörökből állnak. Ezek egy vezérlőmodult alkotnak, pl. Microchip PICmicro MCU, dsPIC Digital Signal Controller (DSC).

Irodalmak:

Nagyon jó oktatási anyag készült az ELTE Alkalmazott elektronika tárgyához, Vella Péter honlapján találhatóak[7].

A PIC architektúrájáról és programozásáról jó oktatási prezentációt készített Mersich András[8].

Táblázatos összefoglalót készített Myke Predko[9].

Számos jól használható részletes leírás jelent meg könyv formájában. A címéhez híven szinte minden megtalálható Di Jasio és szerzőtársai könyvében[10].

Kifejezetten önálló tanulásra szánta a könyvét John Morton[11].

Szintén önálló tanulásra alkalmas Sid Katzen könyve[12].

Több könyv is számos konkrét projectet ír le PIC mikrovezérlők felhasználásával[13], [14], [15], [16].

Kifejezetten kezdők számára írta meg a könyvét Nebojsa Matic[17].

A robotok építése terén is kedvelt megoldás a PIC mikrovezérlők alkalmazása. PIC18-as szériával mutatja be saját megoldását Sylvain[18], [19].

PDA alkalmazásával mutat be robot építést Doug Williams, de a módszerei mikrovezérlős alkalmazásokra is átültethetők[20].

A PIC assembler programozásához jó alapozó cikket írt Regényi Dávid[21].

A mikrovezérlő és a PC közötti különbségek

Amíg egy PC-t sokrétű programok futtatására és különböző eszközök csatlakoztatására használjuk, addig a mikrovezérlők egyetlen feladatra vagy feladatcsoportra fókuszálnak, nem is tartalmaznak többet egyetlen programnál. Tehát az előállításuk nem kerül sokba, mert csak ehhez az előbb említett egyetlen feladat végrehajtásához vannak méretezve, nem úgy, mint egy PC esetén. Mindig drágább és bonyolultabb valami univerzális dolgot előállítani, ami sok mindenre jó. A mikrovezérlő gyakran egy másik készüléknek csak egy kis apró funkcióját ellátó része, mint például füstjelző, televízió, fényképezőgép, videokamera, háztartási gépek, távirányító. A füstjelző feladata az állandó figyelés, a beérkező jelek kiértékelése és ennek függvényében annak eldöntése, hogy megszólaltassa- e a szirénát, amit akkor kell megtennie, ha füst jelenlétéről érkezik be jelzés a szenzoron. A füstjelzőben lévő program végtelen ciklusban fut és fut és fut. A lehetséges két eset a következő, vagy állandó mintavételezés, állandó pásztázás történik, vagy alvó módban várakozik a felébresztő jelre, hogy megszólaltassa a riasztót. Ezt a feladatot egy PC is el tudná látni, de nyilván látjuk, hogy fölösleges lenne erre az egyetlen kis feladatra egy komplett számítógépet üzemeltetni, nem is lenne sem költség-, sem energia-hatékony. Ha a tennivalót hasonló egyszerű feladatokra tudjuk bontani, akkor közvetlenül a mérés helyére helyezhetjük a mikrovezérlővel felépített áramkört. Ezek után vizsgáljuk meg, milyen fő egységeket tartalmaz egy mikrovezérlő.

A mikrovezérlő komponensei

A PIC egyes részei:

- Programmemória
- Fájlregiszterek
- I/ O portok
- A/ D konverterek
- Soros port
- Időzítők
- Egyéb perifériák

Programmemória

Az utasításokhoz, program futtatásához szükséges.

Fájlregiszterek

Olyan memória, mely a program számítási műveleteihez szükséges, ebben tárolódnak a változók.

I/ O portok

Állhatnak magas vagy alacsony logikai szinten. Inputként használva ezek bemenetként szolgálnak. Külső jelet érzékelve, azt feldolgozva, kielemezve tud a rendszer reagálni, például megszólaltatni egy hangszórót. Ha outputként, kimenetként használom, akkor ezeken a portokon tudok jeleket küldeni kifelé, amivel meg tudok hajtani például LED-eket vagy az előbb említett hangszórót tudom megszólaltatni.

A/ D konverterek

Ezek segítségével analóg kimenetű szenzorok jeleit is tudom fogadni. A külső analóg jeleket digitálissá alakítja és így digitális formában a processzor már fel tudja dolgozni.

Soros port

A másik mikrovezérlővel vagy egy PC-vel való kommunikációra szolgál.

Időzítők

Használhatók időzítésre, impulzusszámlálásra, lefutó és felfutó élek közti időtartamokat mérnek, jelet generálnak és fogadnak a kommunikációhoz valamint újraindítják a mikrovezérlőt, ha „lefagy”.

Egyéb perifériák

Egy példa: kritikusan alacsony tápfeszültségnél inkább biztonságosan elmentjük az adatokat és kikapcsoljuk az eszközt mielőtt teljesen megszűnik a tápfeszültség és akkor már nem tudnánk elmenteni az információkat.

A Timer modul

Időzítő modulok minden középkategóriás eszközben vannak. A TIMER0 modul jelen van minden PIC-ben.

Jellemzői:

- 8 bites időzítő/számláló
- olvasható és írható
- 8 bites szoftver programozható előosztóval
- belső vagy külső órajel választó
- megszakítás interrupt FFh- ról 00h-ra való túlcsorduláskor
- a lefutó vagy felfutó él közötti választás

Capture és Compare modul (Kiolvasó és összehasonlító modul)

Számos középkategóriás mikrovezérlő tartalmaz egy vagy több capture és compare modult, Capture/ Compare/ PWM modul neveken. Fő funkciója a modulnak az időzítő teljesítőképességének fokozása. Minden modul a következőket tartalmazza:

- 16 bites regiszter, ami működhet 16 bites kiolvasó (capture) regiszterként vagy 16 bites összehasonlító (compare) regiszterként
- PWM Master/ Slave terhelhető regiszter

Ha egy mikrovezérlőben egynél több capture/compare modul van, akkor azonos működéssel bírnak. A 16F877-es mikrovezérlőben a két modult CCP1 és CCP2 jelöli. Minden modulban egy Capture/ Compare/ PWM Register1 (CCPR1) két 8 bites regiszterből áll: CCPR1L (low byte, logikai nulla (0)) és CCPR1H (high byte, logikai 1 (1)). A CCP1CON regiszter ellenőrzi a CCP1 működését.

A CCP modul többek között időtartam mérésnél, számlálásnál, impulzusok és periodikus hullámok generálásánál és feszültség átlagolásnál használatos.

A kezdő PIC használók számára az elérhető irodalomból nehézséget szokott okozni a soros kommunikáció megértése. Ezért a soros kommunikációt részletesebben leírom.

Soros interfész

A soros interfészek kezelését a pl. PICBASIC utasítások elvégzik, de közvetlenül a regiszterek írásával, olvasásával is kezelhetők. Röviden összefoglalom a legfontosabbakat, hogy az adott PIC kézikönyvének használatához segítséget nyújtsak. Nem részletezem az egyes regiszterek és bitek jelentését, ami a kézikönyvből kiolvasható. A kézikönyv használatát ez a rövid összefoglaló nem helyettesíti, csak (remélhetőleg) könnyíti.

A PIC áramkörök általában kétféle hardver soros (serial) áramkört tartalmaznak.

- EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter), amit néha SCI (Serial Communications Interface) néven is emlegetnek.

- MSSP (Master Synchronous Serial Port)

Az elnevezésüket általában nem szokták magyarra lefordítani. Az aszinkron (asynchronous) elnevezés arra utal, hogy a közös nulla vezetéken kívül csupán egyetlen vezetéken áramlanak az adatok a PIC- ből kifelé (Transmitter, Tx) és egy másikon befelé (Receiver, Rx). Nincs szükség külön úgynevezett órajel (clock) vezetékre, ellentétben a szinkronnal (synchronous), ahol órajel vezetéket is használunk az egységek közötti adatátvitelre.
A soros átvitel során az elküldendő adatbájt egyes bitjeit időben egymásután küldjük el ugyanazon az egy vezetéken. Először a legalacsonyabb helyiértékű bitet, utolsóként a legmagasabb helyiértékűt. Hogy mikor érkezett meg a bit, a szinkron átvitelnél az órajel jelzi. Aszinkron átvitelnél egy kezdő (start) bit elküldése jelzi a bájt küldésének a kezdetét és legalább egy „stop” bit, azaz szünet a végét. A közbülső bitek érkezését az eltelt idő alapján állapítja meg a vevő egység., ezért a vevőnek is tudnia kell előre, hogy milyen időközönként érkezik egy bit. Ezt jelzi az adó és vevő esetén megegyező értékre beállított úgynevezett baud érték, ami a másodpercenként átvitt bitek számát jelenti. Szokásos értékeit – pl. 9600 – táblázatból nézhetjük ki. Az átvitt adat többnyire 8 vagy 9 bit, amit szintén előre rögzíteni kell.

Az EUSART használata

Ez az áramkör – mint a neve is jelzi – mind szinkron, mind aszinkron átvitelre használható.

Aszinkron mód

Aszinkron módú adat küldés a PIC Tx lábán

A küldés előtt a következő biteket kell beállítani: TXEN = 1, SYNC = 0, SPEN = 1. A legfontosabb használandó regiszter a „TXREG”. Ebbe írjuk azt a bájtot, amit el akarunk küldeni a soros vonalon. Az átvitelhez idő kell, ezért a TXREG írása előtt ki kell olvasni a TXIF regisztert. Ha a TXIF értéke 0, akkor az előző adat átvitele nem fejeződött be, várni kell, amíg 1 lesz az értéke. Ez megszakítást is okozhat, ha engedélyezve van. Engedélyezése a PIE1 regiszter TXIE bitjének 1-re állításával lehetséges. Az EUSART 9 bites adat átvitelére is használható.

Aszinkron módú adat fogadás a PIC Rx lábán

A fogadás előtt a következő biteket kell beállítani: CREN = 1, SYNC = 0, SPEN = 1. A start bitnek nevezett első bit mindig 0, az utolsó stop bit mindig 1. Ha ez nem teljesül, hibás az átvitel. Az adat beérkezését a PIE1 regiszter RCIF bitjének 1 értéke jelzi, ami megszakítást is okozhat. A beérkezett adat a RCREG nevű regiszterből olvasható ki.

Az EUSART 9 bites adat átvitelére is használható. Ennek tipikus esete, amikor egy közös vezetékre több egység csatlakozik és mindnek saját címe van. Pl. az RS-485 szabvány szerinti kommunikáció is így működik. A kilencedik bit 1 értéke jelzi, hogy cím érkezett. Az érkezett címet meg kell vizsgálni, hogy az adott egység címe-e. Ha igen, akkor a következő adatbiteket neki kell venni.

Szinkron mód

A szinkron mód használata során a PIC TX/CK lábát az órajel, az RX/DT lábat az adat küldésére és fogadására egyaránt használjuk. Ebből adódik, hogy – ellentétben az aszinkron kommunikációval ellentétben – egyszerre csak egyirányú kommunikáció lehetséges, hiszen mindkét irányra ugyanazokat a vezetékeket használjuk. Ezt „half

duplex” néven említi az irodalom. A két irányban külön vezetéken egyszerre adunk és veszünk neve full duplex. Nem szokás magyarra fordítani. A szinkron mód használata során az egyik egység úgynevezett „master” (vezérlő), míg a másik úgynevezett „slave” (szolga) üzemmódban dolgozik. A master adja az órajelet, vezérli az átvitelt. Ugyanazon a vezetéken több slave is lehetséges, de akkor gondoskodni kell arról, hogy a master ki tudja választani a megfelelőt. A PIC EUSART áramköre mind master, mind slave üzemmódban képes működni. Master mód beállításhoz a SYNC = 1, CSRC = 1, SREN = 0 (adás), SREN = 1 (vétel), CREN = 0 (adás); CREN = 1 (vétel) és a SPEN = 1 beállításokat használjuk. Címeik megtalálhatók a kézikönyvben. Slave mód esetén a CSRC bitet 0-ra kell állítani. Az adatok küldésére és fogadására ugyanazokat a regisztereket használjuk, amiket az aszinkron mód esetén.

Az MSSP (Master Synchronous Serial Port) használata

Ez az áramkör szinkron módú adatátvitelre használható. Alapvetően kétféle üzemmódban működhet: az úgynevezett SPI (Serial Peripheral Interface) és az úgynevezett I²C (Inter-Integrated Circuit) módban. Ezeket az elnevezéseket nem szokás lefordítani.

SPI üzemmód

Szinkron soros adatátviteli busz, ami a korábban említett full duplex üzemmódban működik, tehát külön vezeték van az elküldendő és külön a fogadandó adatok számára. A használt vezetékek elnevezései:

- Serial Clock (SCK) soros órajel
- Serial Data Out (SDO) soros kimenő adat
- Serial Data In (SDI) soros bejövő adat
- Slave Select (SS) szolga mód választás, néha a chip select megjelölés is használatos

Vezérlő/szolga (master/slave) módon működik és az adatvezetéken először a legmagasabb helyiértékű bit megy ki/jön be és utoljára a legalacsonyabb helyiértékű. A vezérlő (master) SDO vezetékét a szolga (slave) SDI, az SDI vezetékét a szolga SDO

vezetékéhez kapcsoljuk. Az SCK vezetékek össze vannak kötve, de az órajelet mindig a vezérlő adja. A vezérlőből általában minden egyes szolgához külön SS vezeték megy, de használatos láncba kötés is. Ebben az esetben az első szolga kimenő jele megy a következő szolga bemenetére és így tovább. Az utolsó szolga kimenete a vezérlő bemenetére kerül. Az egész tulajdonképpen egy hosszú eltolási (shift) regiszter. Ezt a módszert az irodalom daisy-chain néven említi. Az SPI üzemmódhoz használatos regisztereket nem részletezem, a kézikönyvben megtalálhatóak.

I²C üzemmód

Fontos üzemmód. Eredetileg a panelon az IC-k egymás közötti adatcseréjére dolgozták ki. Így kommunikál pl. a Microchip cég TC74 típusú hőmérő IC-je is. Az I²C üzemmód szintén vezérlő/szolga mód. Az IC-ket mindössze két vezetékből álló busz köti össze:

- SCL (Serial Clock) soros órajel
- SDA (Serial Data) soros adat

A vezérlő és a szolga között az SCL-t az SCL-hez, az SDA-t az SDA-hoz kell kötni. Mindkét vezetéket egy úgynevezett felhúzó ellenálláson keresztül a pozitív tápfeszültséghez (VDD) kötjük. A buszhoz csatlakozó minden egységnek saját címe van. Az adatátvitel azzal kezdődik, hogy a vezérlő egy start bit után elküldi annak a szolgának a címét tartalmazó bájtot, amelyikkel kommunikálni akar. Ezt követi egy bit, ami azt határozza meg, hogy a megcímzett egységre írni vagy róla olvasni akarunk. Ezek után ha a megadott című egység létezik a buszon, akkor küld egy nyugtázó (Acknowledge, ACK) bitet. Ezt követi a tényleges adatátvitel, melynek pontos leírása az egyes IC-k leírásában található. Gyakran a szolga több különböző regiszterét kell megcímezni, és adatot írni bele vagy olvasni ki belőle.

Architektúra

A PIC mikrovezérlők jellemzői:

- Harvard architektúra
- Az adatmemóriától eltérő szélességű utasításkód
- Általában egyszavas utasítások
- Az utasítások a legtöbb típusnál egyetlen belső órajel ciklus alatt végrehajtódnak
- Átfedéses utasítás végrehajtással nagyobb sebesség (az előző utasítás végrehajtása még nem fejeződött be, de a következő már elkezdődik)
- Csökkentett utasításkészlet
- Speciális, memóriába ágyazott regiszterek mint perifériális áramkörök

Harvard architektúra

A harvard architektúrájú számítógépek az adatokat és a programutasításokat elkülönítve tárolják. Ezáltal növekedett a teljesítményük és biztonságosabbak lettek. Ezzel ellentétes a Neumann architektúra, ahol ugyanaz a memória akár utasítást, akár adatot tartalmazhat.

Az adatmemóriától eltérő szélességű utasításkód

A memóriabuszok lehetnek eltérő szélességűek, például az adatbusz 8 bites, a programbusz 16 bites, azaz 16 bites az utasításhossza.

Egyszavas utasítások

A PIC sorozat szabványos utasításkészletében a legtöbb utasítás egyszavas.

Az utasítások gyakran egyetlen belső ciklus alatt végrehajtódnak

A PIC mikrovezérlőkben az utasítások egyetlen belső ciklus alatt végrehajtódnak. Először az utasítás beolvasása történik, majd a végrehajtás.

Átfedéses utasítás végrehajtás

Mivel az előző pontban vázolt két ciklus szét van választva, ezért átfedéses utasítás is végrehajtható (Pipelining). Ennek eredménye, hogy az egyszavas utasítás egy gépi ciklus alatt hajtódik végre.

Két gépi ciklusra van szükség a feltételes utasítások igaz ágánál. A kétszavas utasítás két gépi ciklus alatt hajtódik végre.

Minden utasítás négy oszcillátor jel periódusból áll. 4 MHz-s oszcillátor frekvencia esetén egy normál utasítás végrehajtási ideje így 1µs. A feltételes utasítások igaz ágánál a végrehajtási idő 2 µs. Kétszavas feltételes utasításoknál igaz ág esetén pedig 3 µs a végrehajtási idő.

Csökkentett utasításkészlet

A PIC mikrovezérlőkben viszonylag kevés utasítással megoldható minden elvárható programfeladat. Ehhez egységes regiszterkialakítás és jól tervezett utasításkészlet szükséges. A PIC 16-os sorozatánál 35 utasítás is elegendő, a későbbi típusoknál az utasítások száma növekszik. Ezek a mikrovezérlők csökkentett utasításkészletű CPU-val, úgynevezett RISC-el (Reduced Instruction Set Computer) vannak felépítve. Minden regiszter közvetlen és közvetett címzéssel is elérhető. Minden regiszter bit címezhető, bitenként módosítható.

Előnyök

- Könnyen megtanulható
- Gyors működés
- A program futási ideje könnyen kiszámítható az egy belső ciklus alatt végrehajtandó egyszavas utasítások miatt

Hátrányok

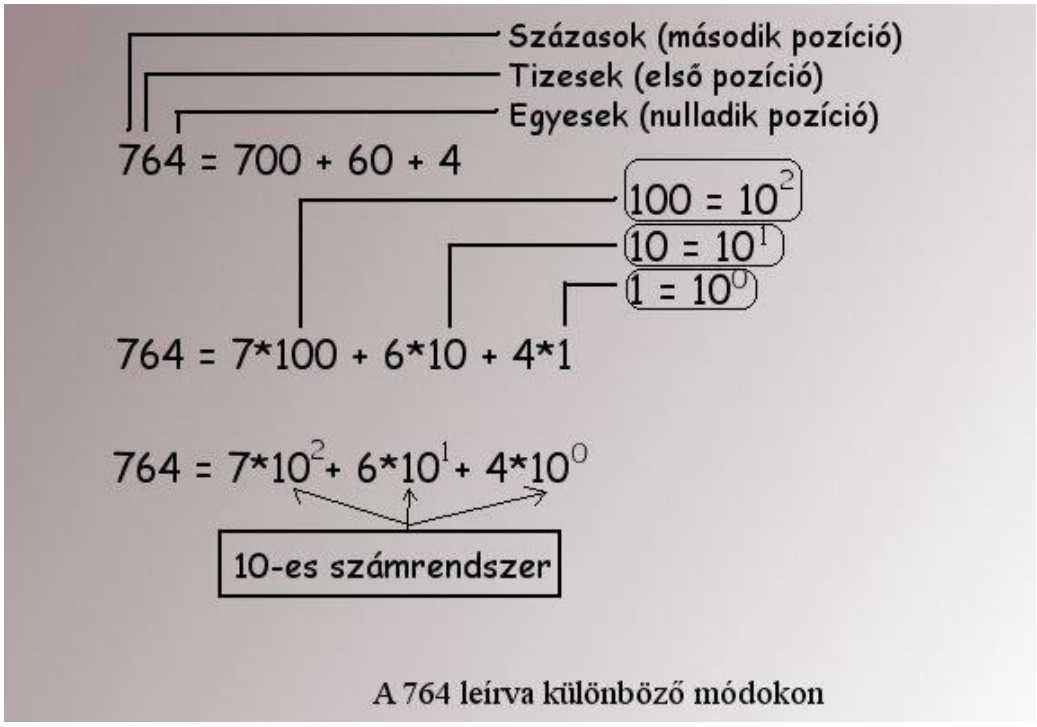
- Magas szintű nyelven való programozásához, komplex fordítóprogram szükséges és hosszú a fordítási idő
- Az utasításkészlet vezérlésorientált, ezért más jellegű feladatok megoldásánál nehézségekbe botlunk

Számrendszerek

Az informatikus körökben ismert vicc szerint kétféle ember létezik, aki ismeri a bináris számokat, és aki. Nem érted ezt a viccet? Akkor te a második csoportba tartozol, de a helyzet nem reménytelen. Ahhoz, hogy a mikrovezérlők működését megértsük, meg kell ismerkednünk a számrendszerekkel, így a következő részben ezeket mutatom be.

A számok világa

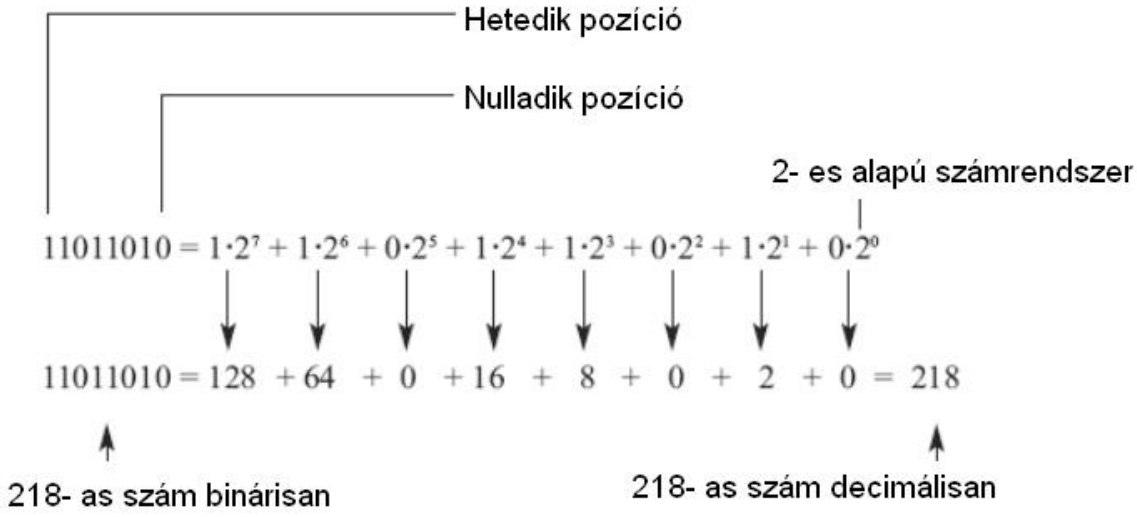
A matematika egy jó tudomány. Minden logikus és egyszerű, mint ez. Az egész univerzum leírható tíz számjeggyel. Vajon pontosan tíz számjegyre van szükségünk? Természetesen nem. Emlékezzünk vissza iskolás korunkra. Például, mit jelentett 764? Négy egyes, hat tízes és hét százas. Egyszerű! Írjuk ezt le kicsit bonyolultabban: 4 + 60 + 700. Még bonyolultabban: 4*1 + 6*10 + 7*100. Kicsit tudományosabban: 4*10^0 + 6*10^1 + 7*10^2. Mit is jelent ez valójában? Miért használjuk pontosan ezeket a számokat: 10^0, 10^1 és 10^2? Miért mindig a 10-es számrendszer? Ez azért van, mert 10 különböző számot használunk (0, 1, 2, ... 8, 9). Más szóval, mert 10-es alapú számrendszert használunk, vagyis decimális számrendszert. 2. ábra



2. ábra

Bináris számrendszer

Mi lenne, ha csak két számjegyet használnánk, a 0-t és az 1-et? Például: 11011010. Egy könyv hány oldalát foglalja magába a 11011010 szám? Annak érdekében, hogy megtudjuk, kövessük az előző példa logikáját, csak fordított sorrendben. Szóval a 2-es számrendszer (bináris számrendszer) csak két számjegyet használ, a 0-t és a 1-et. (pl. 3. ábra)



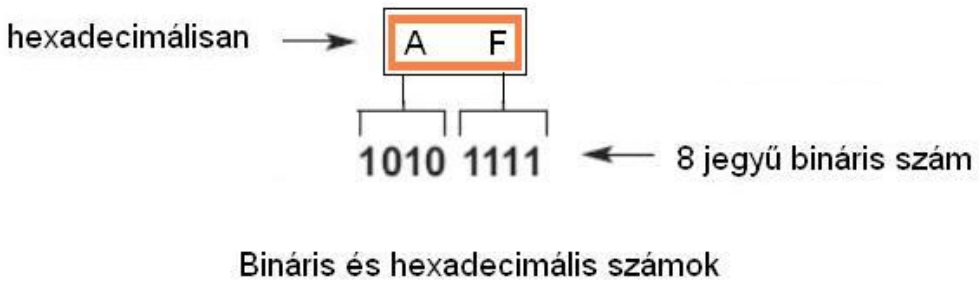
3. ábra

Nyilvánvaló, hogy ez a szám ugyanaz, csak két különböző módon van leírva. Az egyetlen különbség a szám leírásához szükséges számjegyek számában van. Egy számjegy (2) arra van, hogy leírjuk a 2-es számot decimális rendszerben, mivel két szám (1 és 0) arra való, hogy leírjuk a számot bináris számrendszerben. Légy üdvözölve a bináris aritmetikában. Van ötleted hol használják ezt? Különösen szigorúan ellenőrzött laboratóriumi körülmények között, a legbonyolultabb elektronikus áramkörök nem tudják teljes pontossággal meghatározni a különbséget két érték között (például két feszültség érték), ha azok túl kicsik (kevesebb mint néhány milivolt). Ennek az okai az elektromos zajok és valami meghatározatlan dolog, „realisztikus munkahelyi környezet" (tápfeszültség váratlan változásai, hőmérsékletváltozás stb.). Képzeljünk el egy olyan számítógépet, ami 10-es számrendszer szerint működne a következő módon: 0= 0 V, 1= 5 V, 2= 10 V, 3= 15 V, 4= 20 V... 9= 45 V!? Ennyiféle feszültségértéket túl bonyolult volna használni. Sokkal egyszerűbb megoldásnak tűnik a bináris logika, ahol a 0 jelenti,

hogy nincs feszültség, az 1 pedig hogy van. Egyszerűen könnyebb 0- t vagy 1- t írni, ahelyett, hogy kiírnánk „nincs feszültség” vagy „van feszültség”. Ez az úgynevezett logikai nulla (0) és a logikai egy (1), amivel az elektronikai eszközök tökéletesen megbirkóznak és könnyen végrehajtják a végtelenül bonyolult matematikai műveleteket is. Az eszközöknek csak azt kell tudniuk, hogy van-e feszültség vagy nincs. Természetesen ez a digitális elektronika.

Hexadecimális számrendszer

A számítástechnikai fejlesztések legeslegelején rájöttek, hogy az emberek nehezen kezelik a bináris számokat. Ezért egy új számrendszer jött létre, hogy megkönnyítse a munkájukat. Ez 16 különböző számjegyet használ. Az első tíz számjegy megegyezik a megszokott tíz számjegyünkkel (0, 1, 2, 3,... 9), de van még hat további számjegy. Hogy ne kelljen újabb szimbólumokat kitalálni, a ábécé betűit használjuk, A, B, C, D, E és F-et. Összefoglalva a hexadecimális számrendszer számjegyei a következők: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Mi ennek a látszólag bizarr kombinációnak az értelme? Csak nézd, milyen tökéletesen illeszkedik a minden a bináris számokhoz (4. ábra).



4. ábra

A legnagyobb négy számjegyből álló bináris szám az 1111. Ez tízes számrendszerben 15-nek felel meg. Hexadecimális számrendszerben ez egyetlen digittel leírható, ami az F. Ez a legnagyobb egyszámjegyű szám a hexadecimális számrendszerben.

BCD kód

A BCD kód tulajdonképpen egy bináris kód csak decimális számokra. Arra használják, hogy lehetővé tegye elektronikus áramkörök kommunikációját a perifériákkal tízes számrendszerben. 4-jegyű bináris számból áll, ami az első tíz számjegyet képviseli (0, 1, 2, 3 ... 8, 9). Bár négy számjegy összesen 16 lehetséges kombinációt adhat, csak az első tizet használják.

Számrendszerek közti konverzió

A bináris számrendszer a leggyakrabban használt, a decimális rendszer a legérthetőbb, míg a hexadecimális rendszer valahol közöttük helyezkedik el. Ezért nagyon fontos, hogy megtanuljuk, hogyan kell átalakítani egy számot egyik rendszerből a másikba, azaz, hogyan kell nullák és egyesek sorozatát számunkra felfogható értékké alakítani.

Bináris szám decimálisra alakítása

A bináris szám számjegyei különböző értéket jelentenek attól függően, hogy mi a számbeli pozíciójuk. Minden pozícióban lehet 0 vagy 1, értelmezni a jobb oldalról kezdve kell. Ahhoz hogy átalakítsunk egy bináris számot decimálissá, az értéket meg kell szorozni a megfelelő számjeggyel (0 vagy 1) és hozzáadni az eredményhez.

Például:

$$110 = 1*2^2 + 1*2^1 + 0*2^0 = 6$$

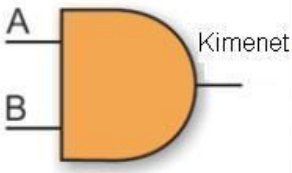
A bináris számrendszerrel a mikrovezérlő könnyen el tudja végezni a számára megadott számításokat és egyéb utasításokat. Ezek végrehajtását egyszerű logikai áramkörökből felépített komplex áramkörök végzik. Ismerkedjünk meg a logikai áramkörökkel.

Logikai áramkörök

Gondolkodjunk el azon, hogy egy digitális integrált áramkör, mikrovezérlő vagy processzor hogy néz ki belülről? Hogy néz ki a bonyolult matematikai műveleteket elvégző áramkör? A látszólag bonyolult kapcsolási rajzok csak néhány különböző elemet, úgynevezett „logikai áramköröket”, „logikai kapukat” tartalmaznak. Elvük a brit matematikus, George Boole által kitalált logika alapján működik. Az elv a 19. század közepén, az első izzó feltalálása előtt fogalmazódott meg. A fő gondolat az volt, hogy a logikai formulák kifejezhetőek legyenek algebrai függvényekkel. Az elméletből hamarosan megvalósulás lett és létrejöttek a ma ismert ÉS, VAGY és NEM logikai áramkörök. Az ismert Boole-algebra alapelvén működnek.

ÉS kapu

Egy logikai ÉS kapu két vagy több bemenettel rendelkezik és egy kimenettel. Nézzük meg a működést két bemenet esetén. A kimeneten csak abban az esetben jelenik meg logikai egy (1), ha mindkét bemeneten (A ÉS B) logikai egy (1) volt. Minden más esetben a kimenet logikai nulla (0) (5. ábra).



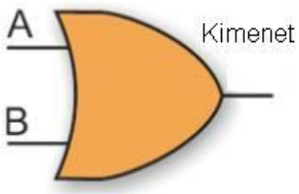
A	B	Kimenet (A ÉS B)
0	0	0
0	1	0
1	0	0
1	1	1

5. ábra

Ha a kapu több mint két bemenettel rendelkezik, akkor is ugyanez a működési elv, a kimeneten csak akkor jelenik meg logikai egy (1), ha az összes bemenet (A ÉS B) logikai egyben (1) van. Minden más bemeneti feszültség logikai nullát (0) eredményez.

VAGY kapu

Hasonlóan az előző esethez, a VAGY kapu is két vagy több bemenettel rendelkezik és egy kimenettel. Nézzük ismét azt az esetet, ha két bement van. Logikai egy (1) jelenik meg a kimeneten, ha egy vagy több bemeneten logikai egy (1) van. Abban az esetben, ha minden bemenet logikai nullában (0) van, a kimenet is logikai nulla (0) lesz (6. ábra).



A	B	Kimenet (A VAGY B)
0	0	0
0	1	1
1	0	1
1	1	1

6. ábra

NEM kapu

Ennek a logikai kapunak csak egy bemenete és egy kimenete van. Működése rendkívül egyszerű. Ha logikai nulla (0) jelenik meg a bemenetén, akkor logikai egy (1) jelenik meg a kimenetén, ha logikai egy (1) jelenik meg a bemenetén, akkor a kimeneten logikai nulla (0) jelenik meg. Tehát ez a kapu megfordítja a jelet, ezért inverternek hívják (7. ábra).

A	**Kimenet (NEM A)**
0	1
1	0

7. ábra

KIZÁRÓ VAGY kapu

Ez a kapu egy kicsit bonyolultabb, mint az előzőek. Ez az össze korábban ismert kapu kombinációja. A bemenet és a kimenet között nem egyszerű meghatározni a kölcsönös függőséget. Logikai egy (1) csak abban az esetben jelenik meg a kimeneten, ha a bemenetek különböző logikai értékkel bírnak (8. ábra).

A	B	Kimenet
0	0	0
0	1	1
1	0	1
1	1	0

8. ábra

Ezek után kezdjük el az ismerkedést a dolgozat fő témáját jelentő PIC mikrovezérlőkkel.

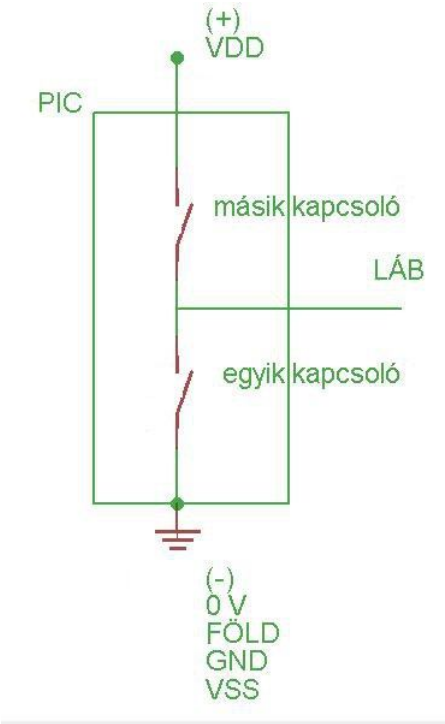
Ismerkedés a PIC mikrovezérlőkkel

Az elkövetkezőkben néhány alapvető fogalmat tisztázunk. Számos cég gyárt mikrovezérlőt, ezeknek vannak hasonló és különböző tulajdonságai. Népszerű az Atmel mikrovezérlő családja. Nagyon sok készülékben található úgynevezett „ARM” mikrovezérlő. Számos telefon, PDA belsejében ezt találjuk. Az ARM mikrovezérlők, microprocesszorok piaci előretörésének lehetünk szemtanúi. Licenszelt architektúra alapján számos cég gyártja. Dolgozatom írásakor vált elérhetővé Magyarországon is a Nuvoton cég ARM M0 szériájú mikrokontrollere, a 32 bites kontrollerek között feltűnően alacsony – néhányszáz forintos – áron. A legegyszerűbb típus 200 (kettőszáz!) Ft körüli áron kapható. Ezért az árért a 32 bites kontroller tartalmaz 8 Kb ROM-ot, 4 Kb RAM-ot, 4 darab 32 bites időzítőt, 2 db UART, 2 db SPI, 1 db I^2C áramkört, 8 csatornás PWM és 8 csatornás, 12 bites A/D konverter áramkört. Sajnos ezek az áramkörök az amatőr gyakorlatban is használható DIP tokozásban nem kaphatók, így barkácsolás szintjén nem alkalmazhatók. Az SMD panel amatőr körülmények között nehezen kezelhető. A panel ára többszöröse, mint a vezérlő maga.

Nagyon nagy mennyiséget gyártanak a Microchip nevű cég PIC nevű mikrovezérlőiből. Széles választékú a DIP tokozású verziója is. Amatőr felhasználásra igen népszerű. Én dolgozatomban ezzel foglalkozom.

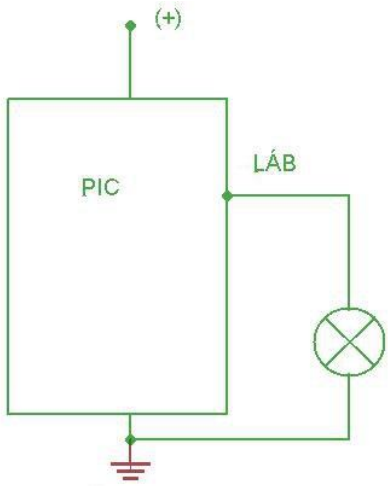
A mikrovezérlők megfelelően programozva nagyon sok feladatot el tudnak látni. Ha el akarjuk kezdeni a használatukat, célszerű először csak a legegyszerűbb feladatokkal megismerkedni és utána folyamatosan bővíteni ismereteinket. A szakirodalom általában túl bonyolultan kezdi, ezért tanár nélkül sokan el se merik kezdeni. Megpróbálok egy olyan – leegyszerűsített – leírást elkészíteni, hogy akár segítség nélkül is el lehessen kezdeni a használatot.

Kezdésként a mikrovezérlőt tekintsük egy olyan eszköznek, amelynek vannak csatlakozói, „lábai”. Ezek a lábak belül kapcsolóáramkörökhöz vannak kötve. A kapcsoló áramkörök a korábban már részletezett kapuáramkörökből vannak felépítve. Képzeljük el, hogy minden lábhoz hozzá van kötve két kapcsoló. Az egyiket ha bekapcsoljuk, akkor az adott lábat a tápfeszültség negatív oldalához csatolja, a másik pedig a pozitív oldalához. Áramkörileg biztosított, hogy mindkét kapcsoló egyszerre soha nincs bekapcsolva, nem leshet rövidzár (9. ábra).



9. ábra

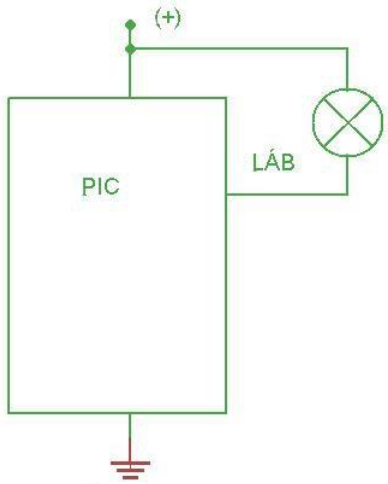
Elképzelhető olyan eset is, hogy egyik kapcsoló sincs bekapcsolva. Akkor azt mondjuk, hogy a láb nagy impedanciás állapotban van, de ezzel az esettel a dolgozatomban nem foglalkozom, bár ez az eset is jelentős: pl. a PIC24-es sorozat 3.3 V-tal működő tagjainál így lehet az 5 V-os szintű környezettel kapcsolatot teremteni. A PIC mikrovezérlők közül elsősorban azokkal foglakozunk, amelyek 5 V, vagy annál valamivel kevesebb tápfeszültséggel működnek. Próbákra tökéletesen megfelel a 4.5 voltos zsebtelep, de egyszerűség végett a dolgozatban mindenütt 5 V-ot említek. Már most megemlítem, hogy a jelenlegi PIC24-es sorozat 3.3 V-tal működik, az 5 V-os széria most van megjelenés alatt. Csatlakoztassuk egy „lámpa" egyik oldalát a PIC lábához, a másik oldalát a PIC- kel közös negatív (föld, GND, VSS) ponthoz. Ebben az esetben a „lámpa" világít, ha a PIC belsejében a láb a pozitív ponthoz van kötve, nem világít, ha a negatívhoz (10. ábra).



10. ábra

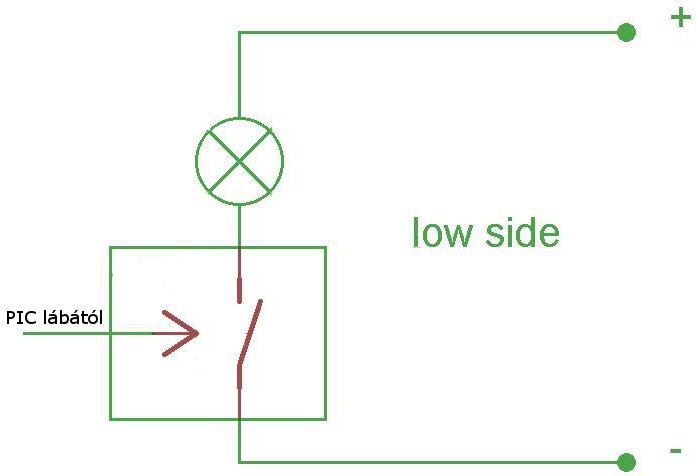
A „lámpa" bármilyen fogyasztó lehet, aminek nem nagyobb a teljesítményigénye, mint amit a PIC elbír. Ez mintegy 50 mA. A gyakorlatban ez azt jelenti, hogy a "lámpa" egy LED (világító dióda) lehet. A továbbiakban a lámpát szimbolikusan egy fogyasztónak tekintjük és nem jelzem idézőjellel. Vizsgáljuk meg a másik esetet is, ha a lámpánkat a láb és a pozitív (+) oldal közé kötjük. Ebben az esetben a lámpa akkor világít, ha a PIC belsejében a láb a negatív oldalhoz van kötve (11. ábra).

A mikrovezérlő programozásával tulajdonképpen ezeket a belső kapcsolókat kacsolgatjuk önműködően.



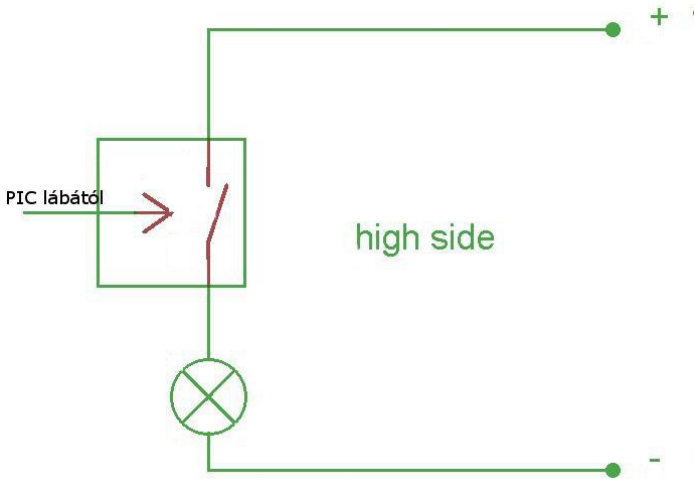
11. ábra

Ha mindössze arra tudnánk használni a mikrovezérlőnket, hogy ledeket villogtassunk, akkor ez egy szűk felhasználási kör lenne. Nagyobb áramot igénylő fogyasztók kapcsolásához egy külön „kapcsolót" kell használni, amit a PIC lába kapcsol. Két alapesetet különböztetünk meg. A kapcsoló a negatív pont (föld) és a lámpa között helyezkedik el. Ezt az irodalom „low side" néven említi. Magyarul néha negatív oldali kapcsolásnak nevezik, bár többnyire az eredeti angol nevét használják. A kapcsoló vezérlését jelképező nyíl a PIC lábához van kötve (12. ábra)



12. ábra

A másik eset, ha a kapcsoló a pozitív (+) pont és a lámpa között helyezkedik el. Ezt az irodalom „high side” néven említi. Magyarul nevezhetjük pozitív oldali kapcsolásnak. (13. ábra)

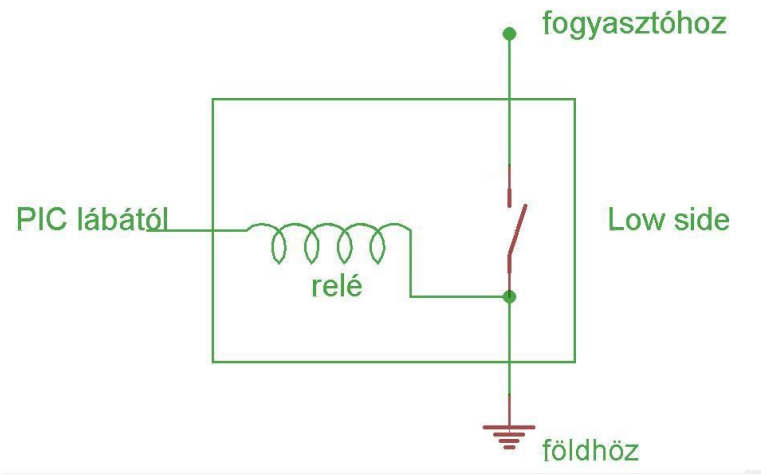


13. ábra

A kapcsoló többféle eszköz lehet, pl. relé (jelfogó), bipoláris tranzisztor vagy FET. Az irodalom általában külön kapcsolásokat közöl az egyes elemekkel való megvalósításhoz. Én ettől eltérek: megadom mind a negatív, mind a pozitív oldali kapcsoló megvalósítását az egyes áramköri elemekkel és a továbbiakban csak mint szimbolikus kapcsolót tekintem. Feltételezek némi előismeretet a relé, tranzisztor és FET működéséről ezért itt csak emlékeztetőként sorolok fel néhány jellemzőt: A relében egy tekercsre adott áram mágneses tere mozgat egy mágnesezhető kapcsolót. A low side és a high side kapcsolás megoldása relével a (14. ábra) (15. ábra) látható.

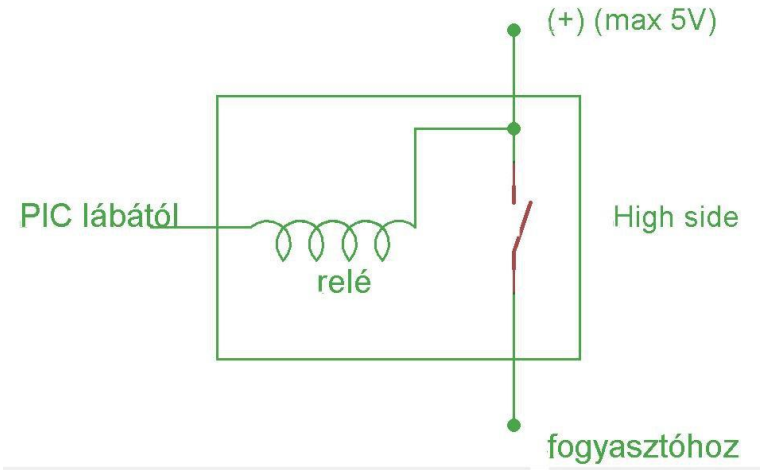
Kapcsolás megoldása relével

Low side



14. ábra

High side



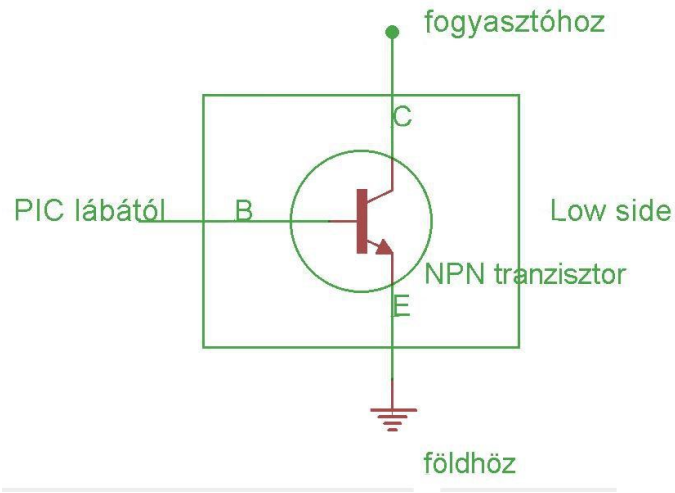
15. ábra

Az általam megadottól különböző kapcsolással megoldható a vezérlő egység és a fogyasztó teljes elektromos elválasztása is. Ezt a relés megoldást csak elméleti okokból adtam meg, a gyakorlatban nagyon ritkán használjuk. Ha mégis, akkor a relé tekercsének induktivitása miatt egy diódát párhuzamosan kell kötni a tekercsel. Én ezt nem jelöltem, hogy csak a lényeg szerepeljen. Inkább tranzisztorral vagy FET-el dolgozunk. A tranzisztort és a FET-et mi kizárólag kapcsoló eszközként használjuk

(digitális technika). Az analóg alkalmazásával nem foglalkozunk. Ha a tranzisztor bázisa (B) és emittere (E) között folyik egy „kis” vezérlő áram, akkor az emitter és a kollektor (C) között vezet (mint egy bekapcsolt kapcsoló), ellenkező esetben nem vezet (kikapcsolt kapcsoló). A „kis” vezérlő áram akkora, hogy a PIC által adott áram megfelelő. Tranzisztorból létezik úgynevezett NPN és PNP. Az áram iránya az NPN és a PNP tranzisztor esetén ellentétes. Az emitter és a kollektor közül az emitter NPN típus esetén a negatívabb, PNP esetén a pozitívabb. A bázis az emitterhez képest az NPN tranzisztornál pozitív, a PNP tranzisztornál negatív ugyanúgy, mint a kollektor. Az utóbbi években nagymértékben csökkent a nagy áramerősséget is elviselő FET ára A FET kapcsolását a source (S) és a gate (G) közötti feszültség végzi, nem áram, mint a bipoláris tranzisztornál. Ha van megfelelő feszültség a source és a gate között, akkor a source és a drain között vezet (bekapcsolt kapcsoló) ellenkező esetben nem (kikapcsolt kapcsoló). N típusú FET esetén a source és a drain közül a source a negatívabb, P típusúnál fordítva. A source-höz képest a gate feszültsége ugyanolyan előjelű, mint a drain feszültsége. A FET is tranzisztor, a nevében a T betű azt jelenti. Az irodalom gyakran alkalmazza azt az egyszerűsítést, hogy a bipoláris tranzisztort egyszerűen csak tranzisztorként emlegeti. A két alapkapcsolást a (16. ábra, 17. ábra, 18. ábra, 19. ábra) mutatja.

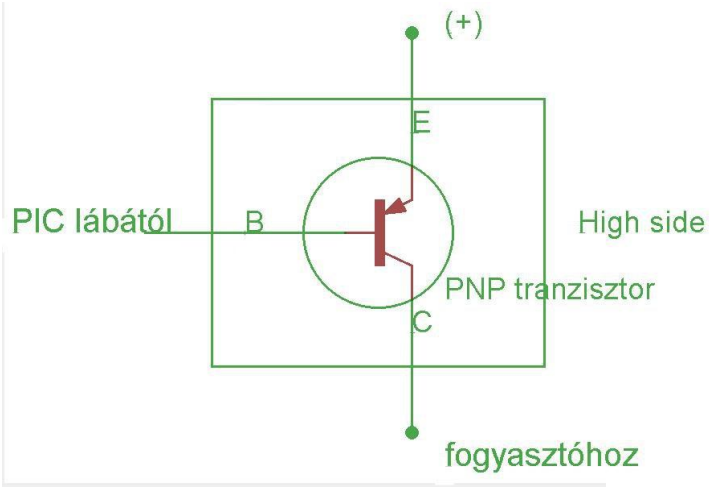
Kapcsolás megoldása bipoláris tranzisztorral

Low side



16. ábra

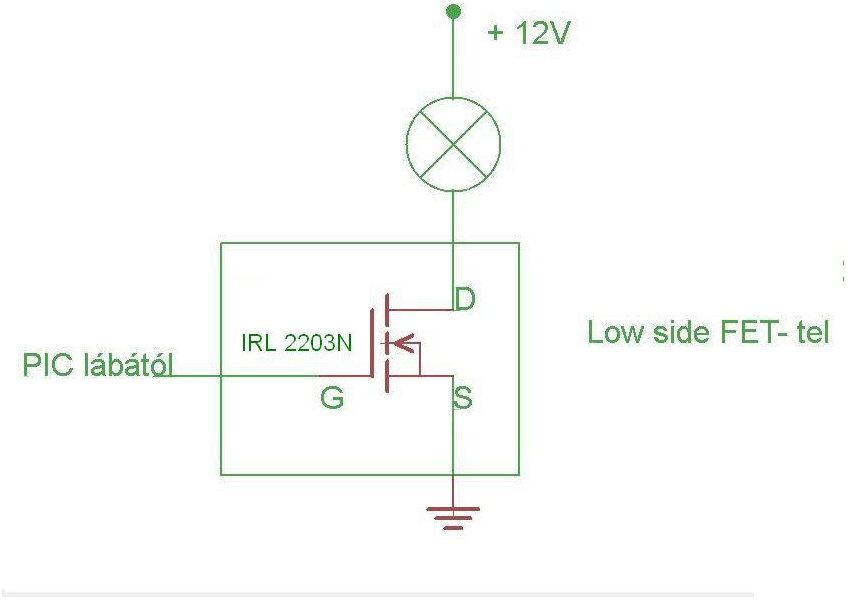
High side



17. ábra

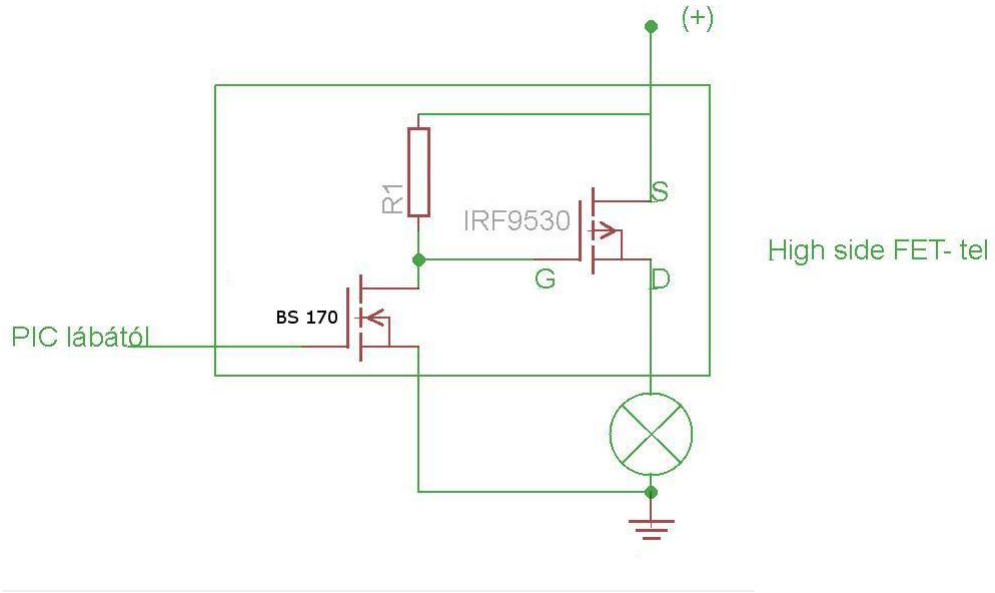
Kapcsolás megoldása FET-tel

Low side



18. ábra

High side



19. ábra

Amikor órán eddig a pontig jutunk, meg szokták kérdezni, hogy milyen szempontok alapján válasszunk a lehetséges alternatívák közül. Néhány szempont a választáshoz:

- Ha a feladat jellege lehetővé teszi, használjunk low side (negatív oldali kapcsolású) megoldást. Vannak esetek, amikor a fogyasztó egyik sarka a PIC föld lábával közös potenciálon – azaz összekötve – kell legyen. Ebben az esetben csak a high side (pozitív oldali kapcsolású) megoldás jöhet szóba. Hasonlóan csak a high side megoldást használhatjuk pl. számos gépkocsi elektronikai megoldásnál. Példának vegyük a hűtő ventilátor motorját. Mivel ennek a kapcsoló áramköre időnként elromlik és felforr a hűtővíz, beépíthetünk egy mikrovezérlős tartalék megoldást. A hűtővíz hőmérsékletét érzékelő szenzor jele alapján a mikrovezérlő elindítja a hűtő ventilátorát, ha azt érzékeli, hogy a hőmérséklet magasabb annál, ami hibátlan működés közben előfordulhat. Azaz meghibásodott a kapcsoló áramkör. Ha a low side megoldást választanánk, akkor a ventilátor motorjának (fogyasztó) egyik sarka állandó összeköttetésben lenne az akkumulátor (+) sarkával. A gépkocsiknál a teljes karosszéria folyamatosan összeköttetésben van az akkumulátor (-) sarkával. Nem lenne szerencsés megoldás, ha egy alkatrész – adott esetben a ventilátor motor – állandóan (+) feszültség alatt lenne akkor is, amikor az autó nem üzemel. Egy

esetleges rövidzárlat komolyabb problémát okozhatna annak ellenére, hogy a túláram kivédésére természetesen beépítünk egy olvadó biztosítékot. Itt a high side megoldást kell alkalmaznunk.

- Relés megoldást ne használjunk. Vannak ugyan olyan relék, amelyek behúzó tekercsének a működtetéséhez a PIC elegendő áramot ad, de hagyjuk a relés megoldást akkorra, amikor már nagy gyakorlatunk lesz. Akkor látni fogjuk, hogy nagyon ritkán kell és egy kicsit másként bekötve használjuk. Ha mégis a relés high side megoldást választanánk, ügyeljünk arra, hogy a relé által kapcsolt feszültség az általam megadott kapcsolással nem lehet nagyobb, mint a PIC tápfeszültsége. (Más kapcsolással lehet, de arra itt nem térek ki, mert az egyes kapcsoló elemek hasonlóságát akartam kihangsúlyozni.)

- A szakirodalom általában a bipoláris tranzisztoros megoldásokat javasolja. Az a véleményem, hogy a technika mai állása szerint a FET árának csökkenése és paramétereinek (megengedett áramerősség) javulása miatt célszerű FET-et használni. Low side megoldáshoz NPN bipoláris tranzisztor vagy N típusú FET kell, high side megoldáshoz PNP bipoláris tranzisztor vagy P típusú FET. N típusú FET esetén ügyeljünk arra, hogy a vezérlő (kapcsoló) feszültsége TTL szintű legyen. Megfelelő pl. az IRL sorozat valamelyik eleme. Ha nem ilyet választunk, a FET-et nem tudja tökéletesen kapcsolni a PIC, mert a PIC által alkalmazottnál nagyobb feszültségre lenne szükség a source és a gate között. Majd ha már komolyabb gyakorlatunk lesz, alkalmazhatunk FET meghajtó áramkört is.

- P típusú FET-ből jelenleg még nehéz beszerezni TTL kapcsoló feszültséggel működőt. Egy másik FET-tel kibővített kapcsolást adtam meg. Az okok és az áramkör részletezése túlmutat a dolgozat keretein, szakkönyvekben megtalálhatók.

- A konkrét kapcsolásokban szükség lehet egyes helyekre ellenállások beépítésére. Erre a konkrét esetekben térek ki.

A PIC mikrovezérlő különböző szériái

PIC 10F széria

A Microchip legkisebb teljesítményű és egyben legolcsóbb mikrovezérlője. 6 lábú tokozásban kerül forgalomba. A 6 láb közül 4 használható ki/bemeneti célokra. Utasításai 12 bitesek. Nagyon kevés a program memóriája (0.25-0.5 Kszó) és az adatmemóriája (16-24 byte). 4 Mhz-es belső oszcillátort tartalmaz, maximális órajel frekvenciája 4 vagy 8 Mhz. 1 db. 8 bites számlálón (timer) kívül más speciális áramkört nem tartalmaz. Megjelenés előtt áll a PIC10F320 és PIC10F322 típus, ami már 16 Mhz órajelet használ és 2 db. 8 bites számlálója van. EEPROM-ot nem tartalmaz.

Tudásuk nagyon kevés, ezért használatukat nem javasolom annak ellenére, hogy áruk 100-150 Ft körül van.

PIC 12F széria

A 10F szériához hasonlóan kevés lábú verzió. 8 lábú tokozásban gyártják, amiből 6 használható ki/bemeneti célra. Utasításai 14 bitesek. A sorozat különböző típusainak az árai gyakorlatilag megegyeznek, ezért célszerű a legnagyobb tudásút választani. Jelenleg a 12F683 egy gyakran használt típus. 2 K szó program memória, 128 byte adatmemória, 256 byte EEPROM, 20 Mhz frekvencia, 8 Mhz belső órajel generátor, 4 csatornás A/D konverter, 2 db. 8 bites és 1 db. 16 bites számláló, 2-5.5 V tápfeszültség a jellemzői. Kisebb feladatokra alkalmas. Nemcsak gépi kódú, hanem kisebb PICBASIC vagy C nyelvű program is belefér. Legújabb – már kapható – típus a 12F1822, ami a 12F683 típushoz képest dupla program memóriát tartalmaz és a működési frekvenciája 32 Mhz. Ugyanígy 32 Mhz a belső órajel generátor frekvenciája is. 1 db. USART és 1 db. MSSP modult tartalmaz, ami megkönnyíti a soros kommunikációt. Kibocsátás alatt van a PIC12F1840, amely 4 K szó program memóriát és 256 byte adatmemóriát tartalmaz. Áruk 200- 250 Ft körül van.

PIC14 széria

Elavult, gyártás, támogatás megszűnt.

PIC16F széria

Rendkívül népszerű, nagy sikerű típus. Utasításai 14 bitesek. Már száznál is többféle típus tartozik a családba. 14, 18, 20, 28 és 40 lábú DIP és néhány egyéb tokozásban gyártják. Program memória mérete 0.5K-16 K, RAM mérete 25-1536 byte, EEPROM mérete 0-256 byte. Működési frekvenciája 16-32 Mhz. Többnyire tartalmaz 1-2 USART, 1-2 MSSP(SPI/I2C) áramkört és 1-6 db. 8 bites, valamint 0-3 db. 16 bites számlálót. Többnyire tartalmaznak A/D konvertert és belső órajel generátort. Az újabbak az érintőpanel kezelését könnyítő áramkört is tartalmaznak.

PIC18 széria

A PIC18 széria jó leírása található Dogan Ibrahim könyvében.[22] A PIC16-os széria egy kiváló általános célú vezérlő, de a technika fejlődésével a további fejlesztések korlátokba ütköztek. Kicsi mind a program- mind az adatmemóriája, a verem tárolója (stack). A megszakítás (interrupt) lehetőségek korlátozottak, az összes megszakítás ugyanazt az egyetlen megszakítási vektort használja. Utasításai között nem szerepel szorzás és osztás. A feltételes ugrás megoldása a SKIP és GOTO együttes használatával bonyolult, ezért a Microchip továbblépésként kifejlesztette a PIC18 szériát az alábbi lehetőségekkel: 77 utasítás, PIC16 forráskód kompatibilitás, nagyobb címtartomány, nagyobb sebesség, 8*8-as hardverszorzó egység, többszintű megszakítás, továbbfejlesztett perifériális lehetőségek, stb.

PIC24 széria

A további fejlesztéseket korlátozta az eddig bemutatott vezérlők 8 bites adatszélessége. A továbblépést a PIC 24-es széria jelentette, melynek legfontosabb újdonsága a 16 bites adatok kezelése. Megtartották a Harvard architektúrát, 24 bites utasítás hosszal. Az utasítás számláló (program counter) is 24 bites, ami 4 megaszó utasításmemória címzését teszi lehetővé. Megjelent 16 darab munkaregiszter (working register), melyek

mindegyike tartalmazhat adatot, címet vagy különbségi címet (offset). Megjelent többféle címzési mód. Rendelkezik 17*17 bites szorzó és egészosztást támogató egységgel. Néhányan úgy ítélik meg, hogy az egyre népszerűbb ARM processzorok architektúrájához kezdett hasonlítani. Használata valóban kényelmes, ezért mindenkinek javasolom.

PIC-ek összehasonlító táblázata

Az alábbi táblázatban felsoroltam néhány – általam kiválasztott PIC adatait, jelenlegi áraival együtt (1. táblázat).

A PIC használatát bemutató irodalom zöme – sajnos – a PIC 16F84-et, PIC16F628-at és a PIC16F877-et veszi mintapéldának. Ez elég nagy baj! Ezek ugyanis régi típusok, jelenlegi áruk a teljesítményükhöz viszonyítva irreálisan magas. Dolgozatom egyik fontos célja, hogy lebeszéljem a PIC áramkörrel ismerkedő diákokat ezek használatáról. Válasszunk modernebb áramkört. Ha nagyon egyszerű áramkört akarunk, akkor se a PIC16F84-et válasszuk. Nagyon olcsó megoldás helyette a PIC16F54/57/59. Ha nagyon kis méretű, egyszerű megoldásra vágyunk, alkalmazzuk a PIC12F sorozatból a PIC12F683-at! Ha a 16F szériából akarunk keresni, akkor jó választás a PIC16F1938/39. Ha nincs különösebb okunk a 16F széria használatára, akkor válasszunk a 18F vagy inkább a 24-es sorozatból. A 18-as sorozat eléggé hasonló a 16-os sorozathoz, csak kényelmesebb, logikusabb, nagyobb tudású. A 24-es sorozat tagjai már egy magasabb kategóriát képviselnek, így javasolt azok használata.

Ha eldöntöttük, hogy melyik PIC áramkörrel akarunk dolgozni, szükség van egy fejlesztő környezetre.

Fejlesztőkörnyezet

Ahhoz hogy a mikrovezérlő tudjon egyáltalán valamit csinálni, ami megkülönbözteti például egy darab fától, kell bele egy program, amit nekünk kell megírni vagy letölteni valahonnan, ha megépítünk egy kapcsolást. A program megírásához kell egy fejlesztőkörnyezet, egy programozói felület, egy szövegszerkesztő felület. A beágyazott rendszerekben a programok általában fixek, nem lehet őket könnyen módosítani, nem úgy, mint egy PC-nél, hogy tudok választani: egy web böngészőt indítok el, egy szövegszerkesztőt szeretnék épp futtatni vagy bármi mást. A következőkben egy ingyenesen elérhető fejlesztőkörnyezetet mutatok be.

MPLAB

A Microchip által biztosított PIC assembly fejlesztői környezet neve MPLAB. A Microchip honlapjáról ingyenesen letölthető program. A legfrissebb MPLAB verzió a 8.66-os. A mikrovezérlő programozásához van szükségünk erre a fejlesztőkörnyezetre. Integrált fejlesztőkörnyezet, azaz több részből áll és többféle programozót és debuggert (hibakeresőt) kezel. 32 bites Windows operációs rendszereken fut, Mac, UNIX vagy Linux rendszereken nem fut. A fejlesztői környezet egységes, grafikus felhasználói felülettel rendelkezik.

Részei

- Szerkesztő
- PASM nevű assembler
- MPLINK linker
- hibakereső

A program megírását a szerkesztőben végezzük. Úgy viselkedik, mint bármelyik Windows-os szerkesztő és a szokásos szerkesztő funkciókkal rendelkezik, például kivágás- beillesztés, keresés- csere, visszavonás- újra funkció.

PASM assembler

A programok lefordítása a feladata ennek a résznek. Beolvassa a szerkesztőben megírt assembler nyelvű forrásfájlt és kódot generál belőle. A generált kód lehet abszolút vagy áthelyezhető. Az abszolút kód közvetlenül a PIC-ben hajtódik végre. Az áthelyezhető kód összekapcsolható más, külön- külön összeállított modulokkal vagy könyvtárakkal.

MPLINK linker

Ez az összetevő egyesíti az assembler által generált modulokat könyvtárakkal vagy más fájlokkal egyetlen futtatható „.hex” formátumú fájllá.

Hibakereső

Több hibakereső kompatibilis az MPLAB fejlesztői környezettel. A hibakeresők átnézik a kódot, a program kritikus helyein töréspontokat helyeznek el a könnyebb vizsgálhatóság érdekében, és figyelik a változókat, regisztereket a program futása közben. Ez egy hatékony eszköz, hogy javítsuk a programhibákat.

Telepítés

A Microchip honlapjáról http://www.microchip.com töltsük le az MPLAB IDE v8.66 szoftvert. Telepítés után klikkeljünk az MPLAB IDE ikonra. Állítsuk be a megfelelő elérési utakat, mappákat.

Mintapéldák MPLAB-bal

Ha felinstalláltuk az MPLAB rendszert és felraktuk a HI-TECH C fordítót, elkezdhetjük első C nyelvű programunk megírását. Ha úgy döntünk, hogy kipróbáljuk a CCS cég C fordítóját is, akkor azt is installáljuk fel.[23] Ennek a demó verziója 45 napig használható, de csak élő internet kapcsolattal. Sokan vannak, akik nem szeretik, ha valamelyik program a gép tulajdonosának tudta és engedélye nélkül ismeretlen adatokkal „haza telefonálgat”, ezért ezzel a fordítóval nem is foglalkozom.

A HI-TECH C és az MPLAB C (Microchip) fordítók közül a PIC18-as szériához bármelyiket választhatjuk, de a PIC10/12/16 szériához csak HI-TECH van, a PIC24, dsPIC és a PIC32 szériákhoz csak MPLAB C-t (20. ábra).

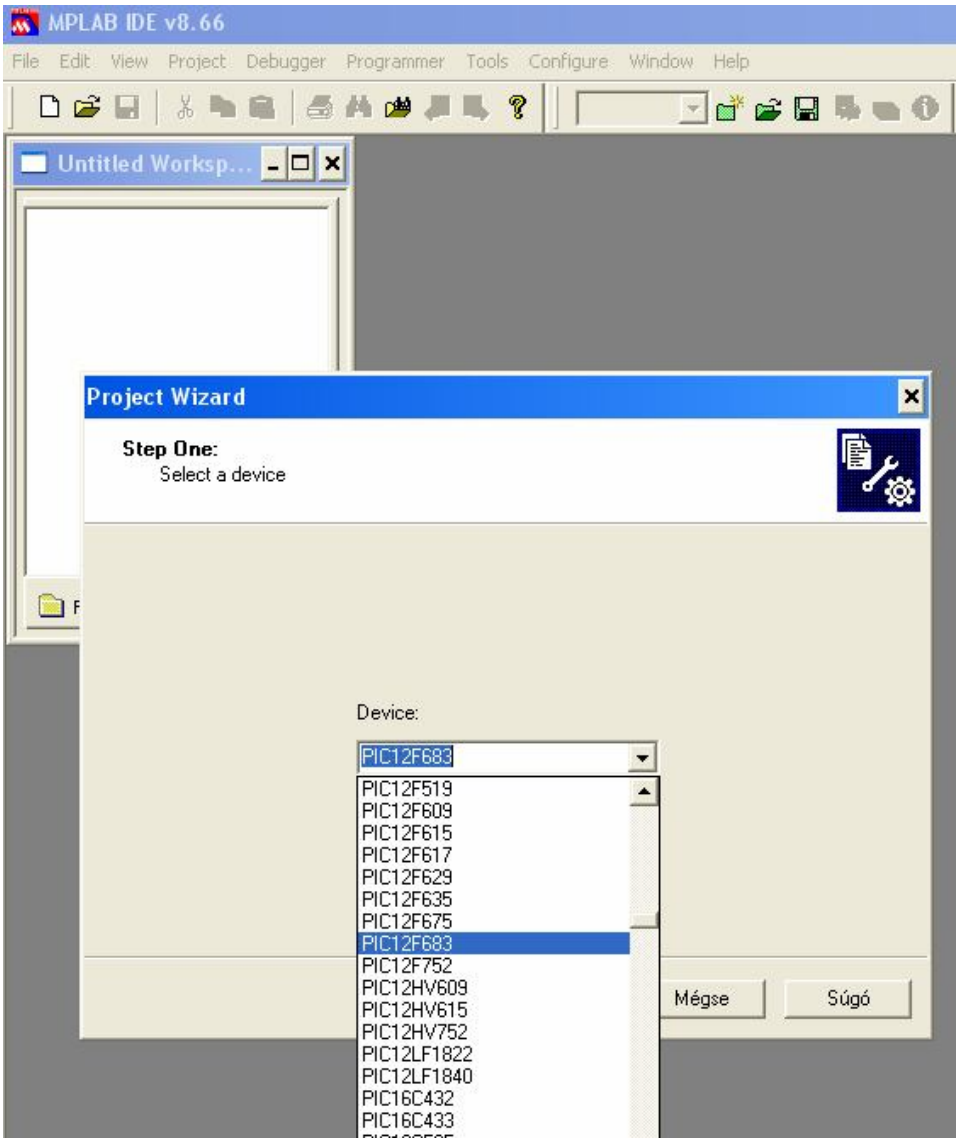
	PRO	Standard	Lite	Evaluation
PIC10/12/16 MCUs	HI-TECH C	HI-TECH C	HI-TECH C	HI-TECH C
PIC18 MCUs	HI-TECH C	MPLAB C HI-TECH C	MPLAB C HI-TECH C	MPLAB C HI-TECH C
PIC24 MCUs and dsPIC DSCs		MPLAB C	MPLAB C	MPLAB C
PIC32 MCUs		MPLAB C	MPLAB C	MPLAB C

20. ábra [24]

Ha az installálással megvagyunk, indítsuk el az MPLAB programot!

Mintapélda PIC12F683-mal

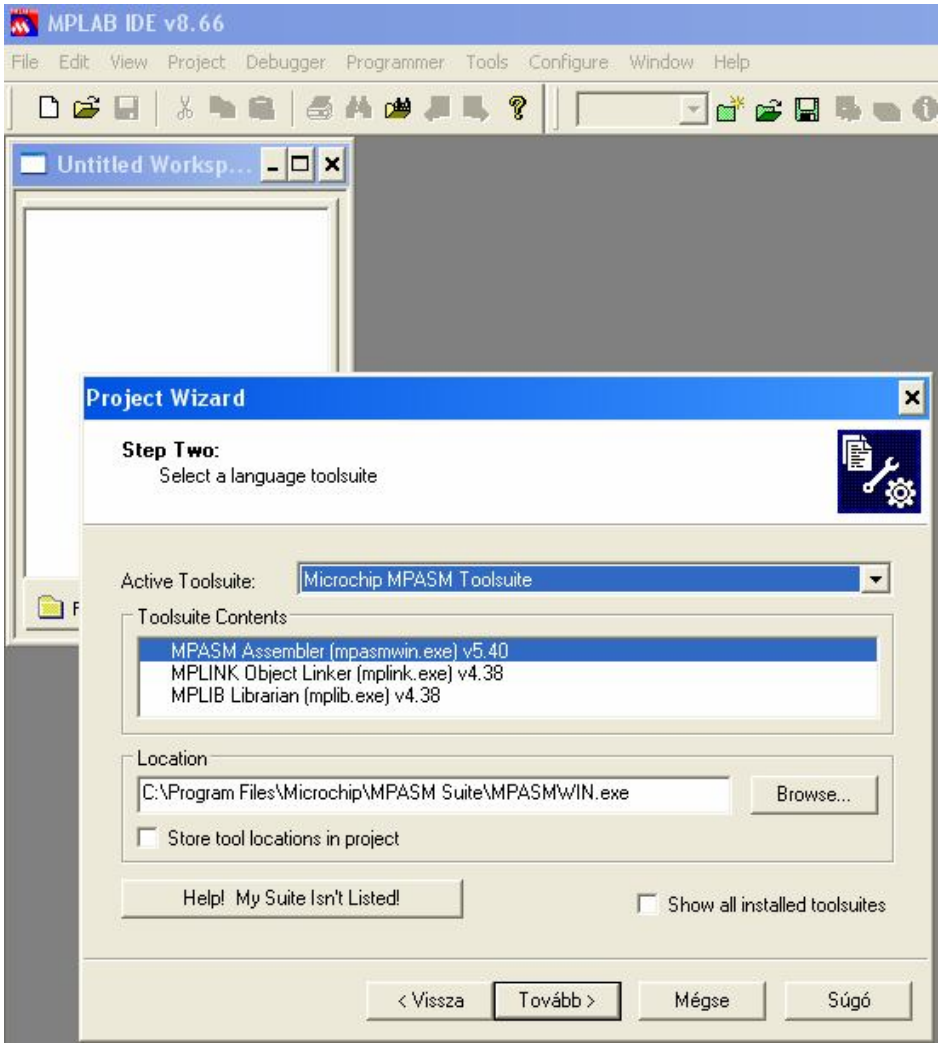
A Project/Project Wizard menüponttal kezdhetünk új projectet. Eső lépésként meg kell adnunk a használni kívánt PIC típusát. Mintának vegyük a PIC12F683-at (21. ábra).



21. ábra

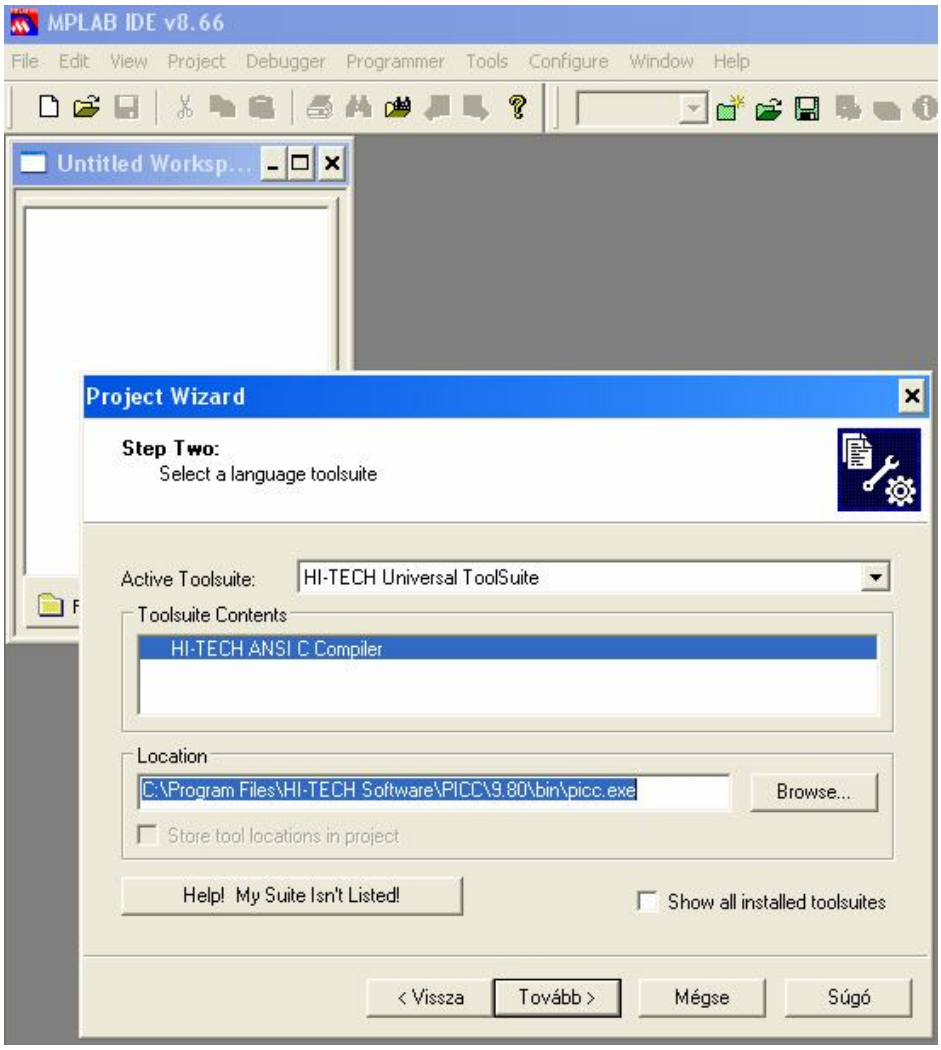
Következő lépésben ki kell választanunk, hogy az MPLAB-on belüli lehetőségek közül melyikkel akarunk dolgozni. Dolgozhatunk pl. assemblerrel, HI-TECH vagy Microchip C fordítóval.

Ha assemblerrel dolgozunk, akkor azt jelöljük ki (22. ábra).



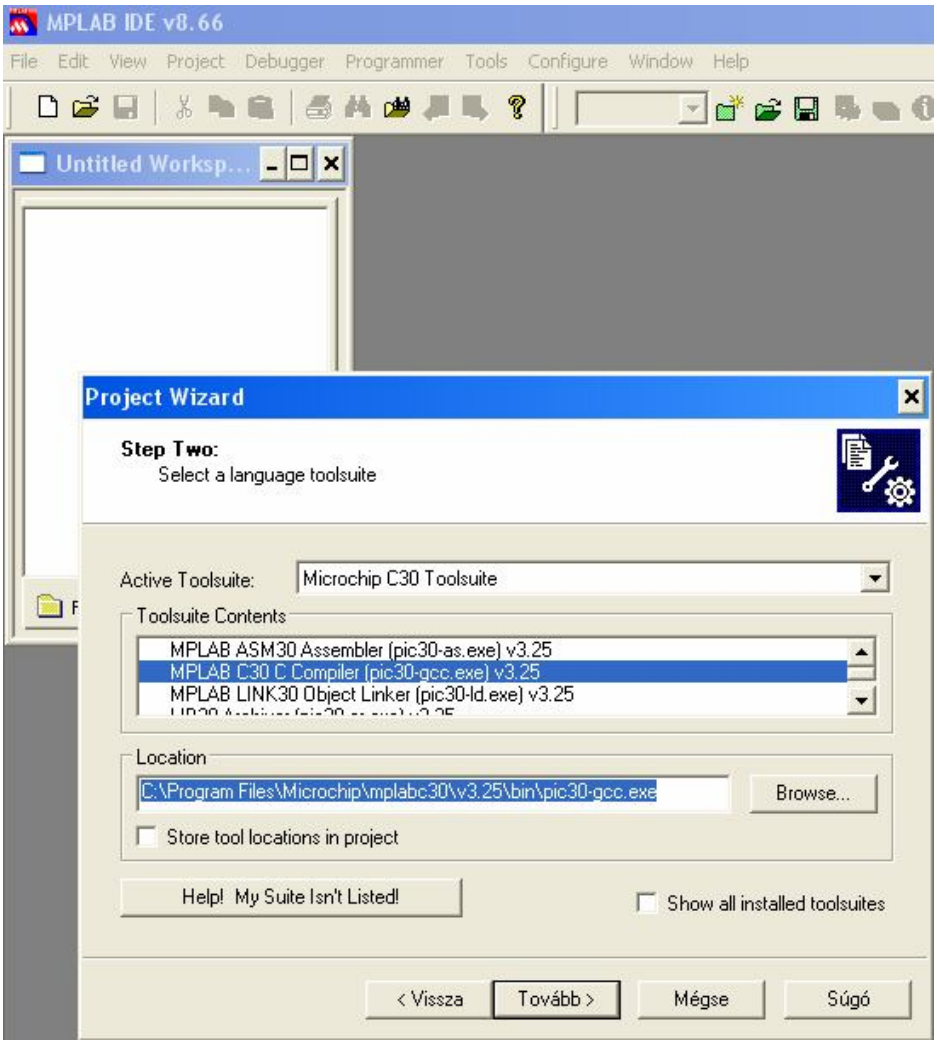
22. ábra

Ha olyan PIC-et választottunk, hogy létezik hozzá HI-TECH C fordító, akkor választhatjuk azt (23. ábra). A következő mintapéldában a 12F683 PIC-et és a HI-TECH C fordítót használjuk.



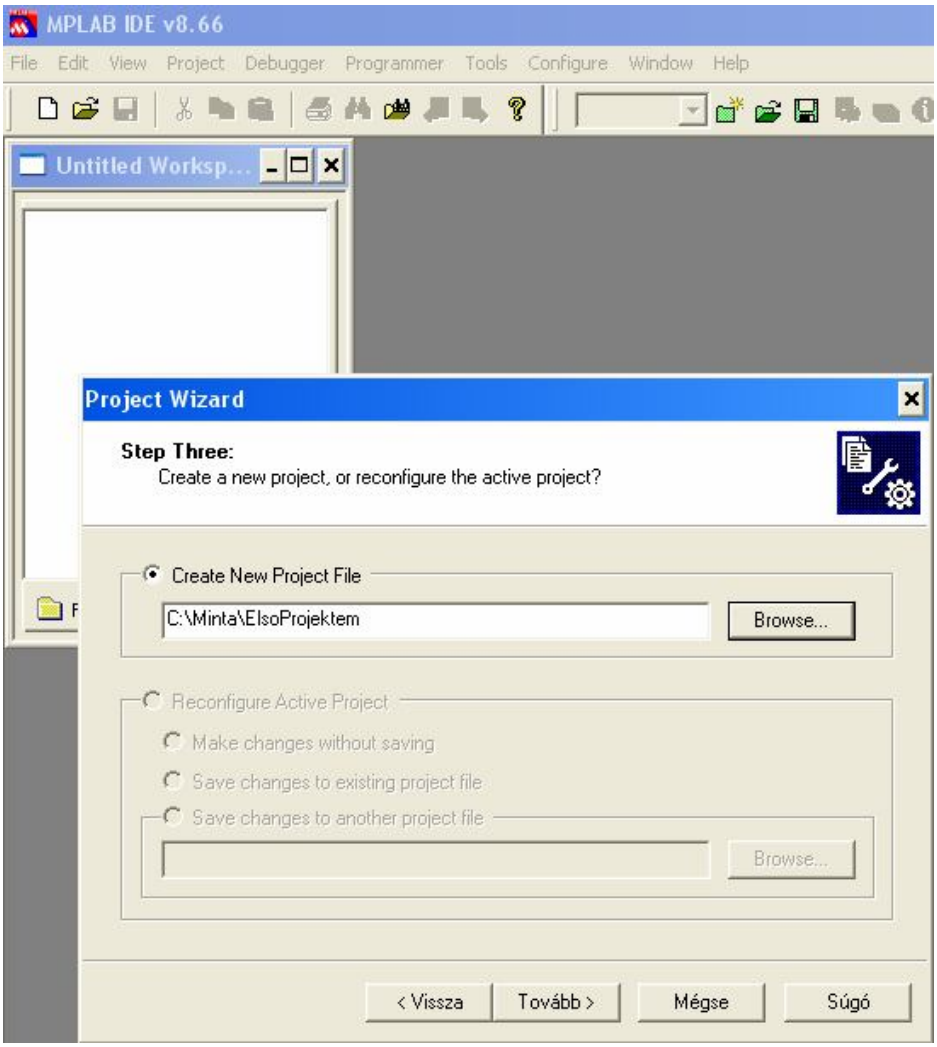
23. ábra

Ha megfelelő PIC-et választunk, dolgozhatunk a Microchip C fordítójával is (24. ábra).



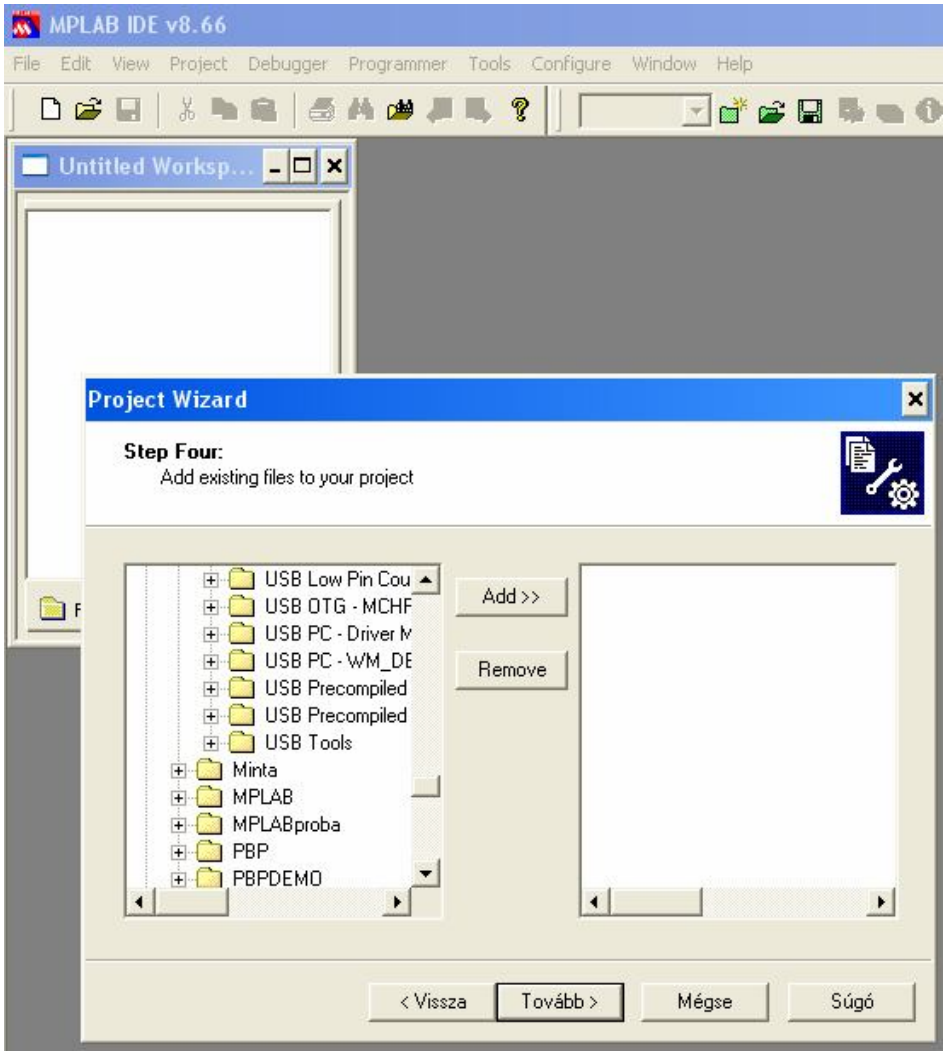
24. ábra

A következő lépés a project nevének a kiválasztása (25. ábra).



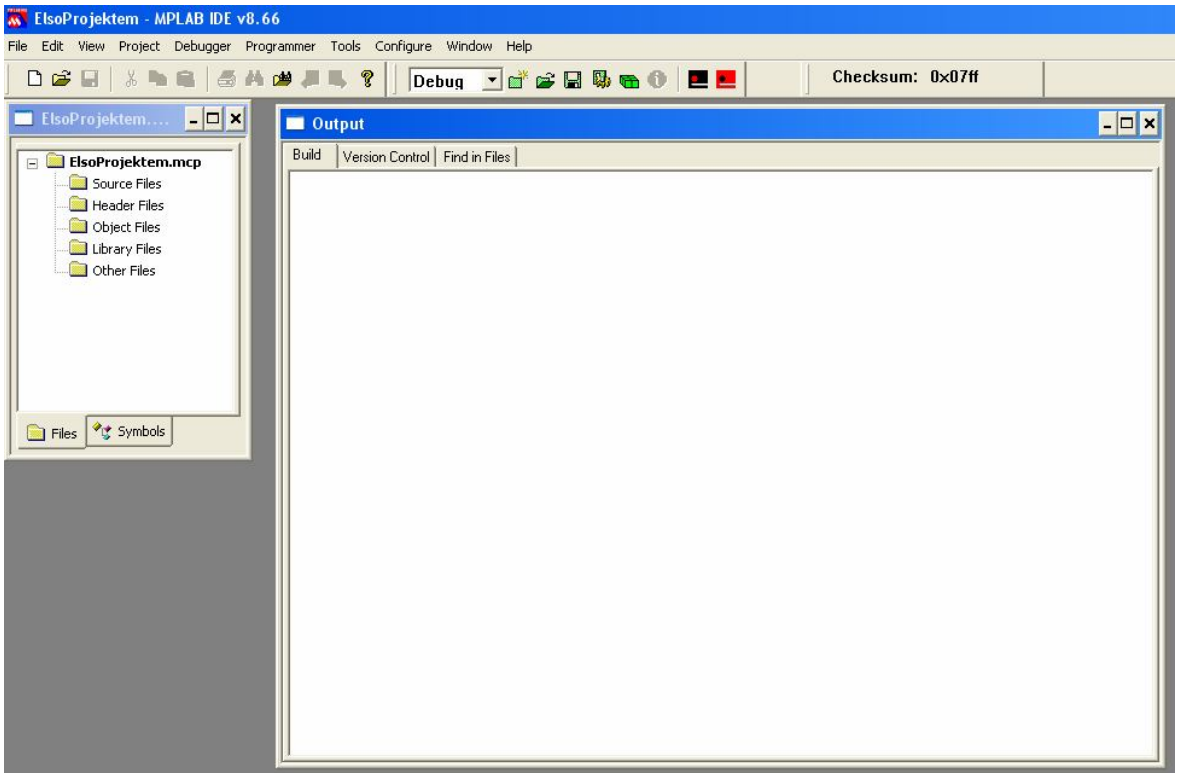
25. ábra

Adjuk hozzá a már megírt fájlt, ha már készítettünk. Ha nem, akkor most nem adunk hozzá semmit, majd később (26. ábra).



26. ábra

A leírt lépések elvégzése után megkapjuk az induló képernyőt (27. ábra).



27. ábra

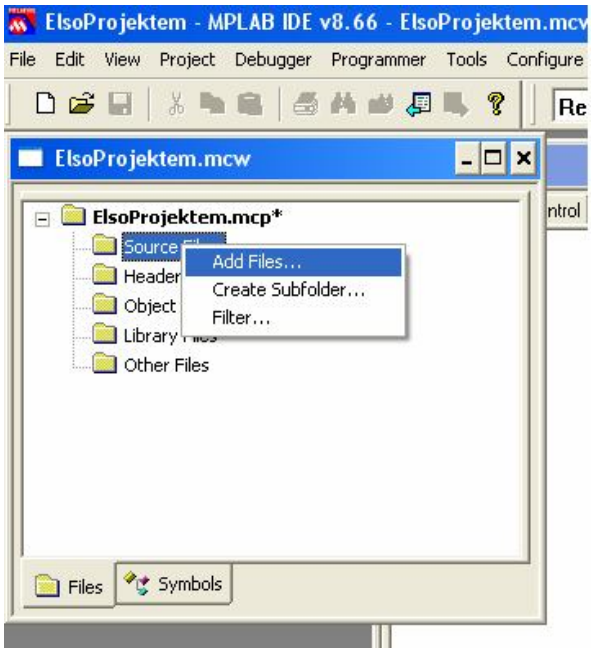
Mintaprogram C nyelven

Most elkezdhetjük a tényleges munkát. Notepad-ben (jegyzettömb) írjuk meg első C nyelvű programunkat. Legyen ez a program az alábbi:

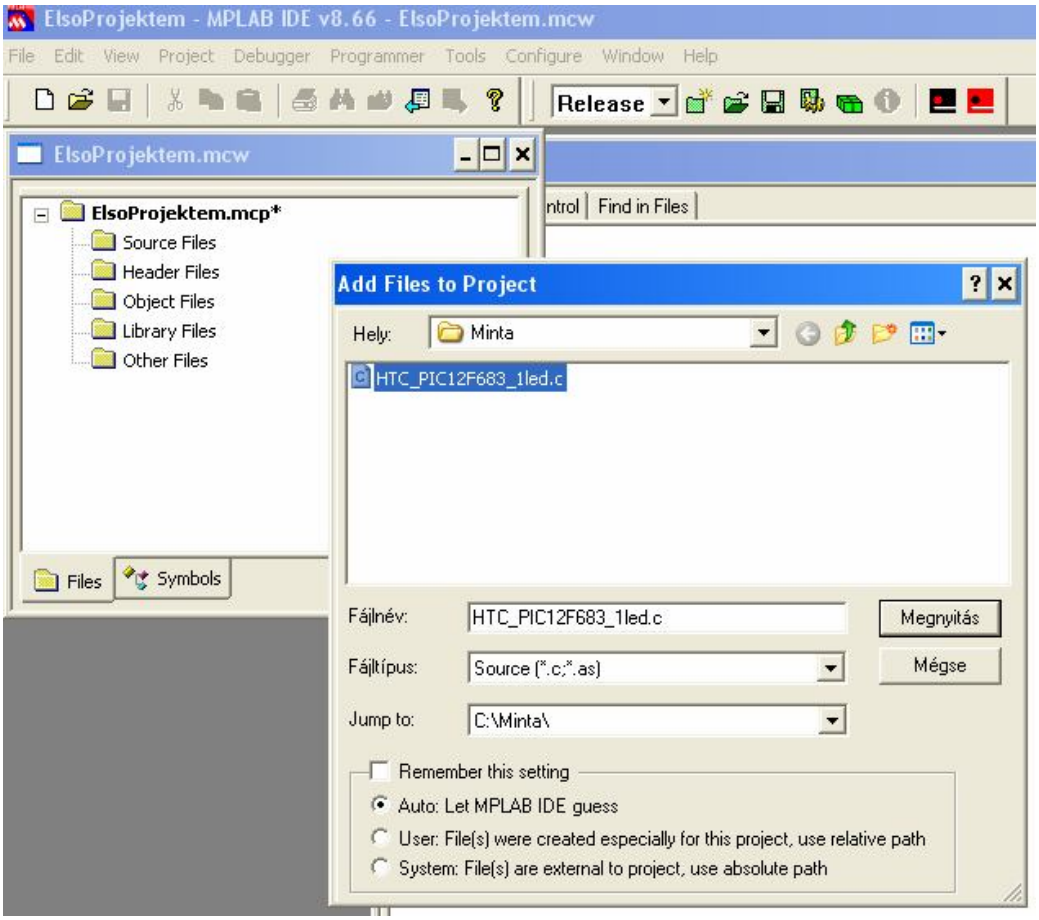
```
#include <htc.h>
void main()
{
TRISB = 0; //PORTB legyen output
PORTB = 0xFF; //B port minden lábát tegyük 1, azaz közel 5V-os állapotba.
}
```

Az első sor (#include) is szükséges. Jelzi a fordítónak, hogy a HI-TECH C előre megírt részeit fogjuk használni, hiszen ezzel a fordítóval akarunk dolgozni. A PIC esetében a portok, amik lábak csoportjai, PORTA, PORTB, stb. néven használhatók. A TRISB adja meg, hogy kimenetként akarjuk használni minden lábát a portnak, mert a TRISB mind a 8 bitjébe 0-t írtunk. Ha valamelyik lábat inputnak szánjuk, akkor az adott lábnak megfelelő bitet a TRISB regiszterben 0-ra kell állítani. Ez a program – szándékunk szerint – a B port minden lábát 1, azaz magas, vagyis közel 5 V-os szintre állítja.

Mentsük el valamilyen néven abba a könyvtárba, amit megadtunk a projekt wizardban. Az MPLAB project ablakában a Source Files alá rakjuk be a megírt fájlt (***28. ábra***, ***29. ábra***).

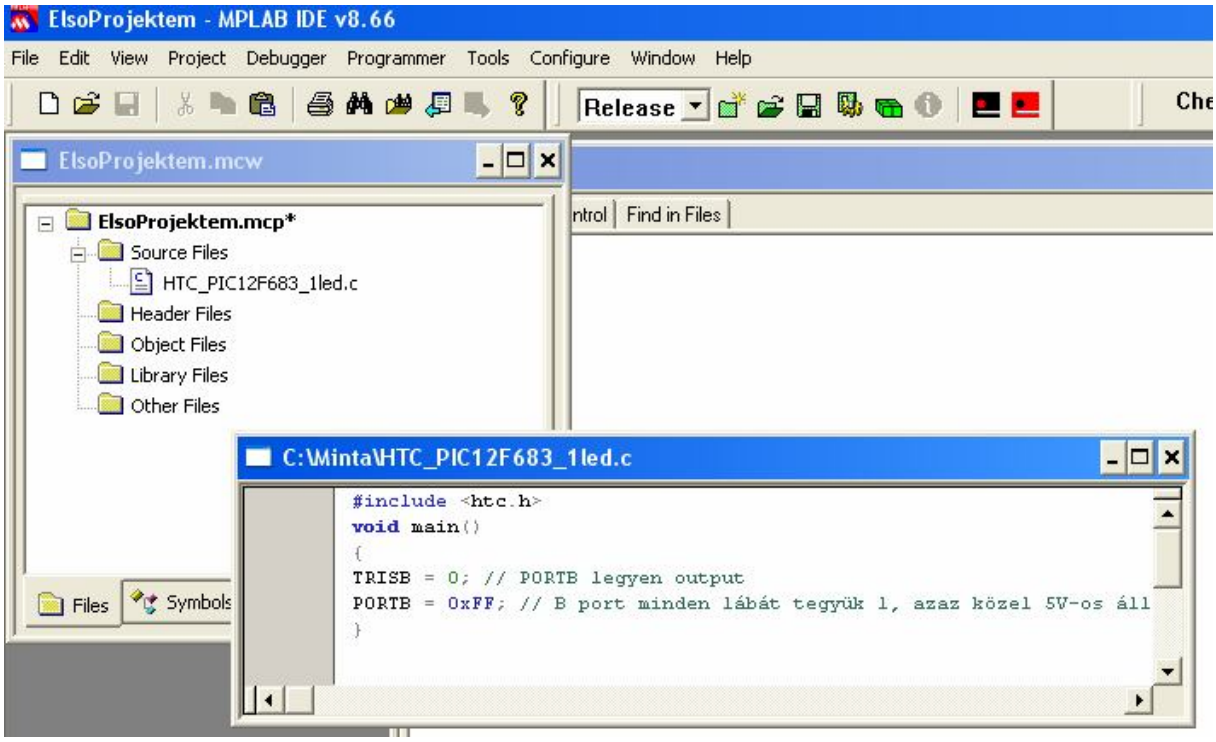


28. ábra



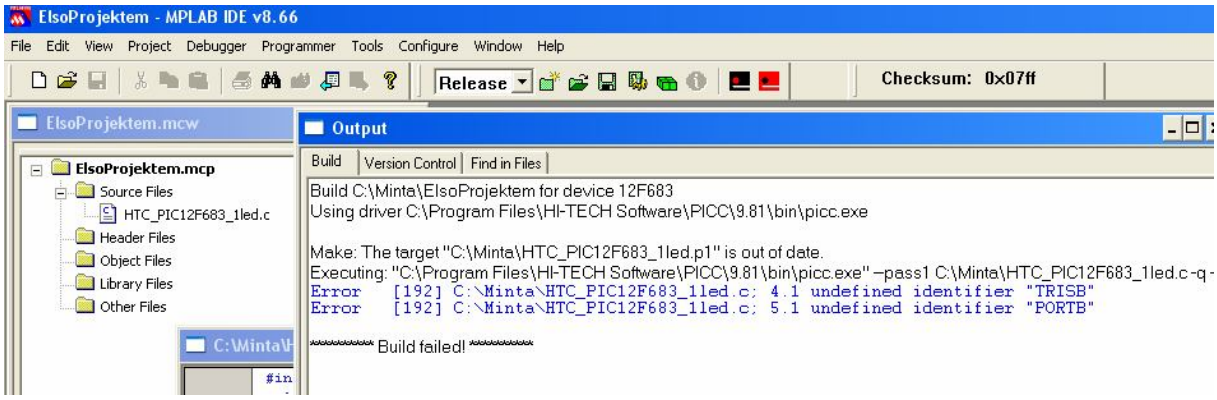
29. ábra

Ha mindent jól csináltunk, a Source Files alatt megjelenik a programunk. Ha duplán kattintunk a Project ablakban a forrásfájlunk nevére, egy újabb ablakban meg is nyithatjuk. Itt szerkeszteni is lehet (30. ábra)



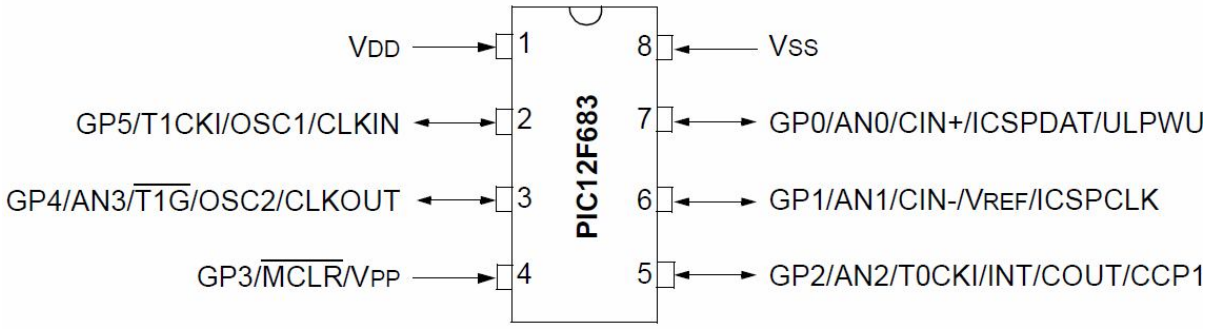
30. ábra

Elképzelésünk szerint már akár fordíthatjuk is a programunkat: Project menü/Build. Ezt az eredményt kapjuk (31. ábra).



31. ábra

Ez bizony hibás! Van két „undefined identifier”, azaz definiálatlan azonosító. A TRISB és a PORTB. Valóban, a többi PIC- kel ellentétben a PIC12-es sorozatnál nem ez a neve a portnak. Hogyan tudhatjuk, hogy mi? Bizony le kell tölteni a Microchip oldaláról a PIC12F683 leírását és meg kell benne nézni. Mielőtt továbbmennénk, vizsgáljuk meg ennek a PIC-nek az adatait. A lábkiosztását a (32. ábra) mutatja felülnézetben.



32. ábra

Az 1. lábra kell kötni a tápfeszültség pozitív sarkát, a 8-asra a negatívot. A többi láb adatforgalomra használható Nézzük meg az adatmemória térképét is (33. ábra).

	File Address		File Address
Indirect addr.(1)	00h	Indirect addr.(1)	80h
TMR0	01h	OPTION_REG	81h
PCL	02h	PCL	82h
STATUS	03h	STATUS	83h
FSR	04h	FSR	84h
GPIO	05h	TRISIO	85h
	06h		86h
	07h		87h
	08h		88h
	09h		89h
PCLATH	0Ah	PCLATH	8Ah
INTCON	0Bh	INTCON	8Bh
PIR1	0Ch	PIE1	8Ch
	0Dh		8Dh
TMR1L	0Eh	PCON	8Eh
TMR1H	0Fh	OSCCON	8Fh
T1CON	10h	OSCTUNE	90h
TMR2	11h		91h
T2CON	12h	PR2	92h
CCPR1L	13h		93h
CCPR1H	14h		94h
CCP1CON	15h	WPU	95h
	16h	IOC	96h
	17h		97h
WDTCON	18h		98h
CMCON0	19h	VRCON	99h
CMCON1	1Ah	EEDAT	9Ah
	1Bh	EEADR	9Bh
	1Ch	EECON1	9Ch
	1Dh	EECON2(1)	9Dh
ADRESH	1Eh	ADRESL	9Eh
ADCON0	1Fh	ANSEL	9Fh
General Purpose Registers 96 Bytes	20h	General Purpose Registers 32 Bytes	A0h BFh
			C0h
			EFh
		Accesses 70h-7Fh	F0h
	7Fh		FFh
BANK 0		BANK 1	

33. ábra

Mint tudjuk, a PIC mikrovezérlők a külvilággal kapcsolatot biztosító valamint a belső működést szabályozó regisztereket „memóriába ágyazott” módszerrel kezelik. Pl. a TMR0 regiszter címe 01h (így jelöli a leírás a hexadecimális számot). Látjuk, hogy két memória bankba vannak csoportosítva a regiszterek. A PIC csak egy banknyi tartományt tud közvetlenül megcímezni, ezért az utasítások előtt be kell állítani aktívnak azt a bankot, amelyikben a használni kívánt regiszter van. Ugyanez a helyzet az írható/olvasható memóriával (RAM) is. Az első bankban (bank0) a 20h és 7Fh között van RAM, a másodikban (bank1) A0H-tól BFh-ig.

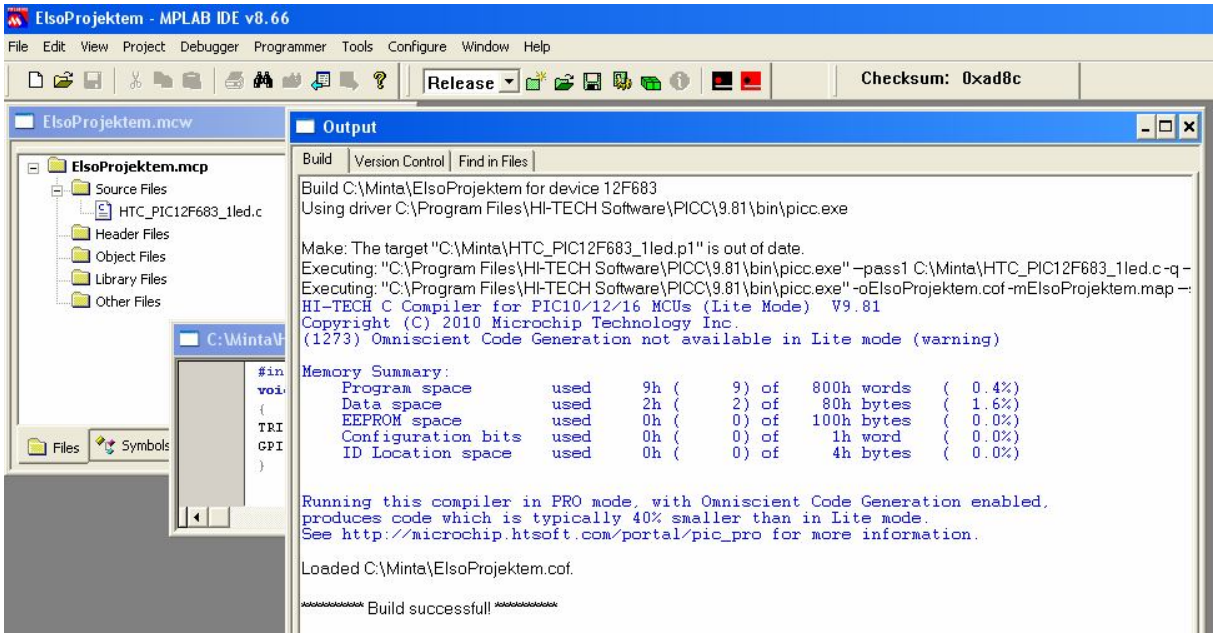
Nézzünk bele a Program Files/Microchip/MPASM Suite/P12F683.INC fájlba is, mert ténylegesen ezt használja a fordító! Egy részletében ezt láthatjuk:

```
; --------- Register Files ----------
; ------------- Bank0 ------------------
INDF ..... EQU  H'0000'
TMR0 ..... EQU  H'0001'
PCL.....EQU  H'0002'
STATUS ..... EQU  H'0003'
FSR ..... EQU  H'0004'
GPIO ..... EQU  H'0005'
PCLATH..... EQU  H'000A'
INTCON .....EQU  H'000B'
PIR1 .....EQU  H'000C'
TMR1L .....EQU  H'000E'
TMR1H..... EQU  H'000F'
T1CON ..... EQU  H'0010'
TMR2 ..... EQU  H'0011'
T2CON ..... EQU  H'0012'
CCPR1 ..... EQU  H'0013'
CCPR1L..... EQU  H'0013'
CCPR1H ..... EQU  H'0014'
CCP1CON ..... EQU  H'0015'
WDTCON..... EQU  H'0018'
CMCON0..... EQU  H'0019'
CMCON1..... EQU  H'001A'
ADRESH .....EQU  H'001E'
ADCON0 ..... EQU  H'001F'

; ------------- Bank1 ------------------
OPTION_REG..... EQU  H'0081'
TRISIO ..... EQU  H'0085'
PIE1 .....EQU  H'008C'
PCON .....EQU  H'008E'
OSCCON ..... EQU  H'008F'
OSCTUNE..... EQU  H'0090'
PR2 ..... EQU  H'0092'
WPU ..... EQU  H'0095'
WPUA ..... EQU  H'0095'
IOC ..... EQU  H'0096'
IOCA ..... EQU  H'0096'
VRCON ..... EQU  H'0099'
EEDAT ..... EQU  H'009A'
EEDATA ..... EQU  H'009A'
EEADR.....EQU  H'009B'
EECON1 .....EQU  H'009C'
EECON2..... EQU  H'009D'
ADRESL.....EQU  H'009E'
ANSEL .....EQU  H'009F'
```

Itt vannak a regiszterek nevei és címei felsorolva. A **GPIO EQU H'0005'** jelenti, hogy a kimeneti port címe 0005 hexadecimális számrendszerben és a neve GPIO. A Microchip leírásában a PIC lábkiosztásának ábráján is így szerepel. Írjuk át gyorsan a forrásfájlban! Mit írjunk a TRISB helyére? A TRIS regiszterek a másik lap ugyanazon helyén szoktak lenni, ahol az adott kapu. Tehát a 0005+0080=0085 összeadás alapján a hexadecimális 85 címen kell keresnünk, mert egy lap címtartománya hexadecimális 80. Valóban, meg is találjuk: TRISIO EQU H'0085'

Most javíthatunk!

```
#include <htc.h>
void main()
{
TRISIO = 0;          // PORT legyen output
GPIO = 0xFF;         //port minden lábát tegyük 1, azaz közel 5 V-os állapotba.
}
```

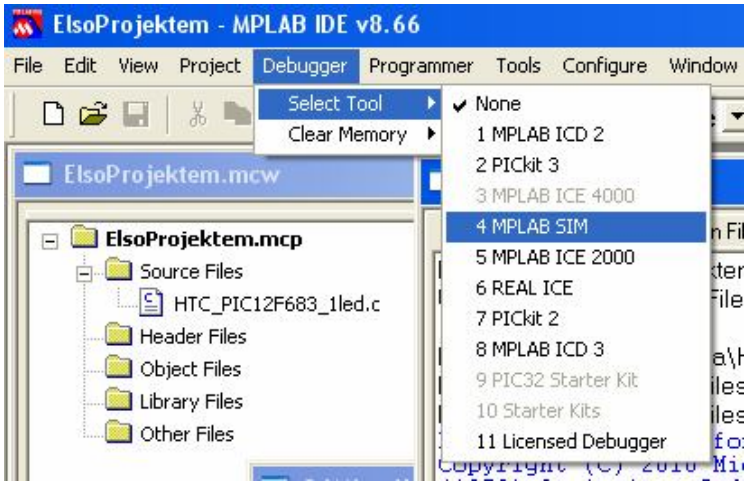


34. ábra

Örüljünk!!! Lefordította! (34. ábra) Persze figyelmeztetésként (warning) tájékoztat, hogy a Lite (ingyenes) verziónál van jobb is, de azt majd úgyis csak akkor vesszük meg, ha ebből akarunk megélni. Tanulásra tökéletes a Lite verzió. Nincs más hátra, mint be kell programozni („égetni”) a lefordított programot a PIC-be. Azért annyira ne siessünk, mert ha hibás, akkor nehezebben lesz kideríthető, mintha most óvatosabbak vagyunk. Először ellenőrizzük le az MPLAB SIM lehetőséggel (35. ábra).

Az MPLAB SIM

Ez egy hibakereső szimulátor. Úgy viselkedik, mintha valóban a PIC-ben futna a programunk és mutatja az eredményeket.

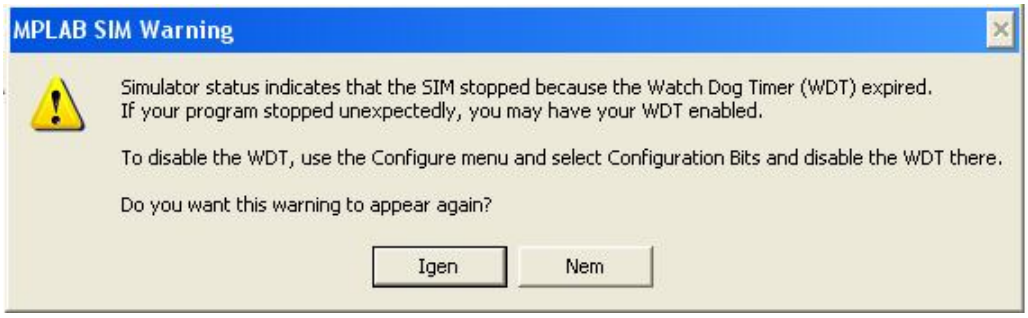


35. ábra

Kiválasztottuk a debuggert (hibakeresőt). Megjelenik egy üres ablak Output néven és a menü sorban valahol a hibakeresést segítő gombok (36. ábra)

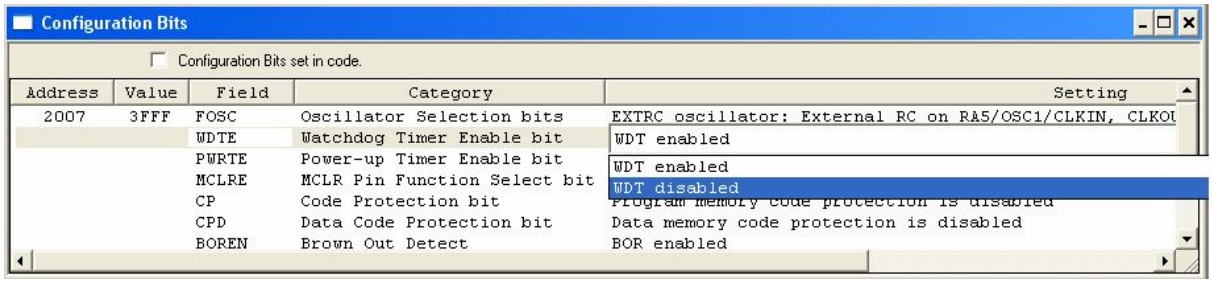
36. ábra

A nyíllal indíthatjuk a programot, a kapcsos zárójel pedig struktúrákba való belépést, átugrást, kilépést jelent, de ezzel részletesebben nem foglalkozunk. Indítsuk el a nyíllal!



37. ábra

Ezt az üzenetet kapjuk (37. ábra). Tegyük, amit javasol. Nyissuk meg a Configure/Configuration Bits menüt (38. ábra)

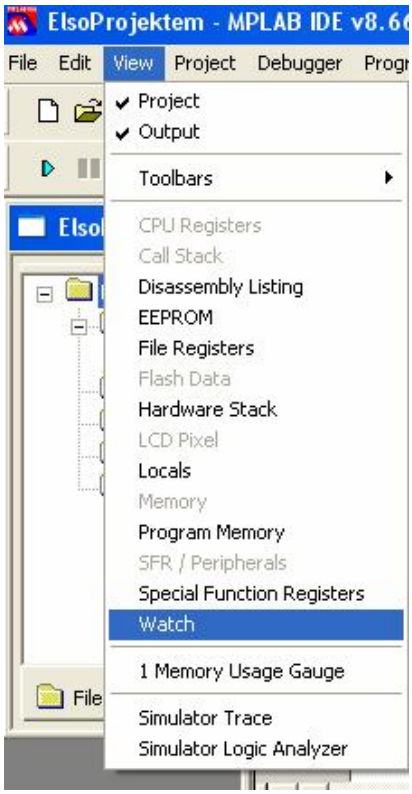


38. ábra

A Configuration Bits set in code lehetőségnél vegyük ki a pipát és már szerkeszthetjük is a konfigurációs biteket. A WDT-t tiltsuk le. Utána ne felejtsük el visszarakni a pipát! Be is csukhatjuk ezt az ablakot. A konfigurációs bitek használatának részletes megismerését hagyjuk akkorra, amikor már gyakorlottabbak vagyunk. A többit is így lehet beállítani

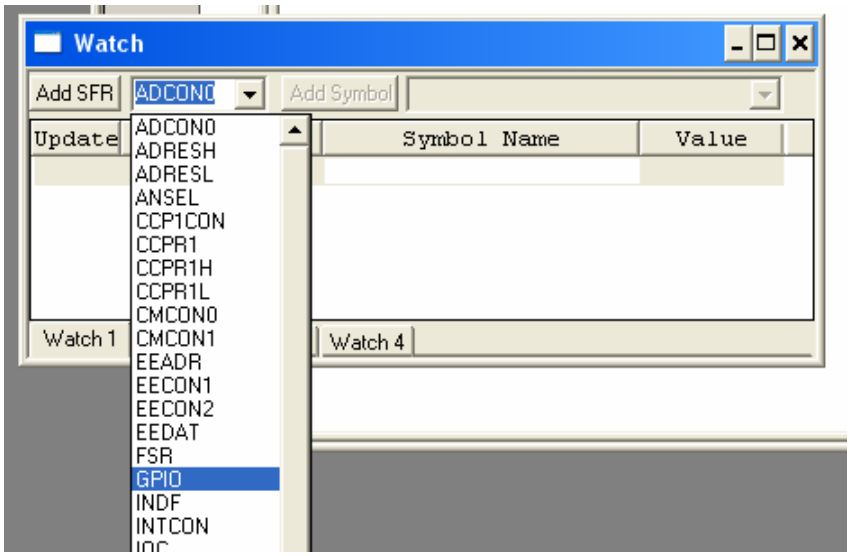
Ha ismét elindítjuk a programot, hibajelzést ugyan nem látunk, de mást se. Először is vegyünk észre egy problémát. A mikrovezérlőket olyan programmal nem szokás használni, ami egyszer fut le! Mindig végtelen ciklust használunk. Ennek tudatában most mégsem írjuk át.

A View/ Watch menüvel nyissunk meg egy hibakereső „watch” ablakot (39. ábra)!



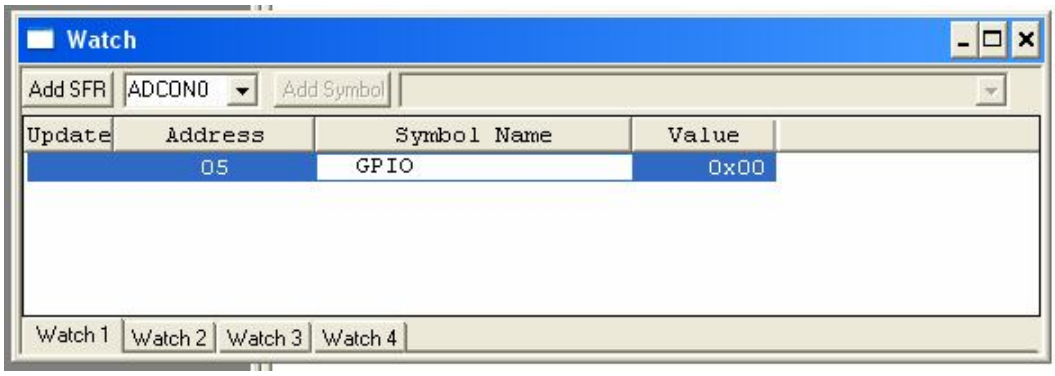
39. ábra

Jelöljük ki a GPIO regisztert, majd az Add SFR gombbal aktiváljuk (40. ábra)



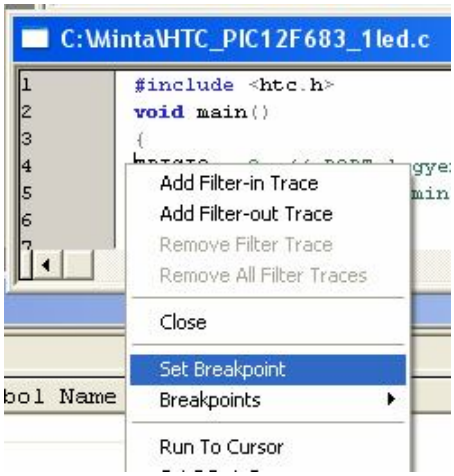
40. ábra

A következő ablakot kell látnunk (41. ábra):



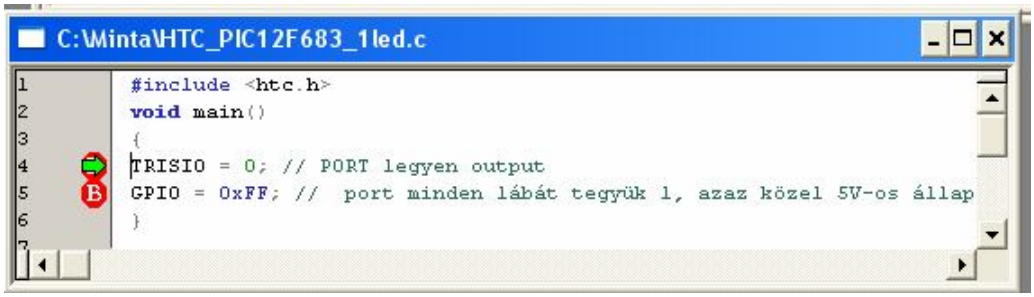
41. ábra

A könnyebb kezelhetőség érdekében az Edit/ Properties menüben a C File Types fülön bejelölhetjük a Line Numbers-t. A debuggert nagyon sokféleképp használhatjuk. Én itt csak egy egyszerű példát mutatok be. A forrásfájlban menjünk a kurzorral a 4. sorra és a jobb egér gombbal helyezzünk el egy töréspontot („break point”) (42. ábra)



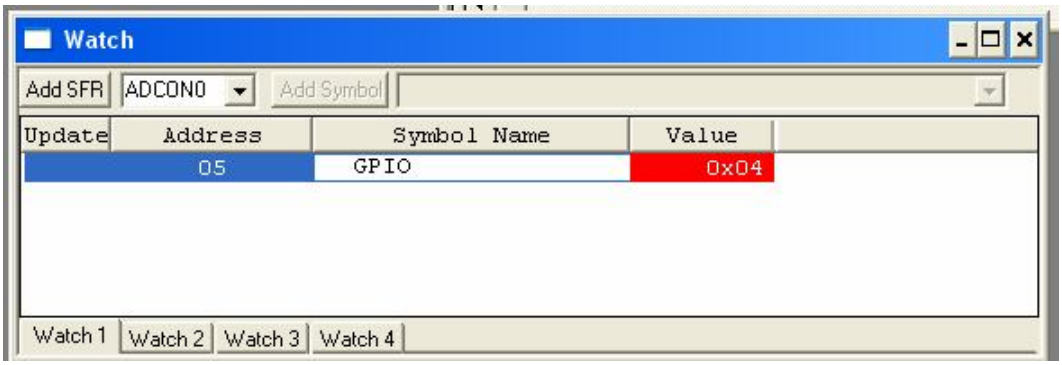
42. ábra

Ugyanezt elérhetjük a sorszámokat tartalmazó mezőn dupla kattintással is. Az 5. sorra ugyanígy tegyünk töréspontot. Látjuk, hogy a töréspontot tartalmazó sorok előtt piros B betű jelent meg. Nyomjuk meg a menü soron az indítást jelentő zöld nyilat! Ezt látjuk a (43. ábra).



43. ábra

Tehát a program elindult, majd az első törési pontnál, azaz a 4. soron lévő utasítás végrehajtása után megállt. A zöld nyíl jelzi, hol áll a program. Közben a Watch ablakban láthatjuk, hogy a GPIO értéke 00. Nyomjuk meg újra a menü soron a zöld nyilat! Azt várnánk, hogy a GPIO értéke FF lesz, hiszen azt raktuk bele. Helyette ezt látjuk (44. ábra):



44. ábra

Elkövettük a kezdő PIC programozók szokásos hibáját! A PIC analóg célokra is használható lábai induláskor analóg használatra vannak beállítva. Ha digitális célra akarjuk használni, először át kell állítani! A PIC12F628 leírásában azt találjuk a GPIO port leírásánál szürkével kihangsúlyozva, hogy ha I/O lábként akarjuk használni, akkor a

```
CMCON0=0x07;
ANSEL=0;
```

utasításokkal az analóg egységet le kell tiltani. Módosítsuk a programot!

```
#include <htc.h>
void main()
{
CMCON0=0x07;
ANSEL=0;
TRISIO = 0;          // PORT legyen output
GPIO = 0xFF;         //port minden lábát tegyük 1, azaz közel 5 V-os állapotba.
}
```

Ha ezt futtatjuk az előbbi módon, akkor azt találjuk, hogy a GPIO értéke 0x17 lesz. Ez még mindig nem az, amit vártunk, de ha ezt a problémát is megoldjuk, több már nem is lesz.

Írjuk fel bináris alakban a 0x17-et:

0001 0111

A legutolsó a GP0, majd balra sorban a GP1, stb. Ezt rakja ki a PIC lábaira. Most, első alkalommal részletezzük egy kicsit. Az utolsó 1 azt jelenti, hogy a PIC GP0, azaz a 7-es sorszámú lábára 1 kerül kiírásra, ami azt jelenti, hogy „magas feszültség szintre”, azaz 5 V közeli állapotba kerül. A GP1, GP2 úgyszintén. A GP3-nak megfelelő helyen 0 áll, tehát a PIC 4-es sorszámú lábán 0 van, ami alacsony (0-hoz közeli) feszültséget jelent. Vizsgáljuk meg az ominózus nullákat. A legegyszerűbb a helyzet a két balszélső bittel: ilyen lábak ezen a PIC-en nem is léteznek, tehát nincs vele dolgunk. Mi a helyzet a GP3-mal? Szintén megtalálhatjuk a leírásban, hogy ez mindig input, nem tudjuk outputra állítani. Még mindig ott van problémás esetnek a GP5. Vegyük elő a leírást és nézzük meg a lábak bekötését! A 2-es sorszámú lábnál, azaz a GP5-nél a következőket találjuk:

GP5/T1CKI/OSC1/CLKIN

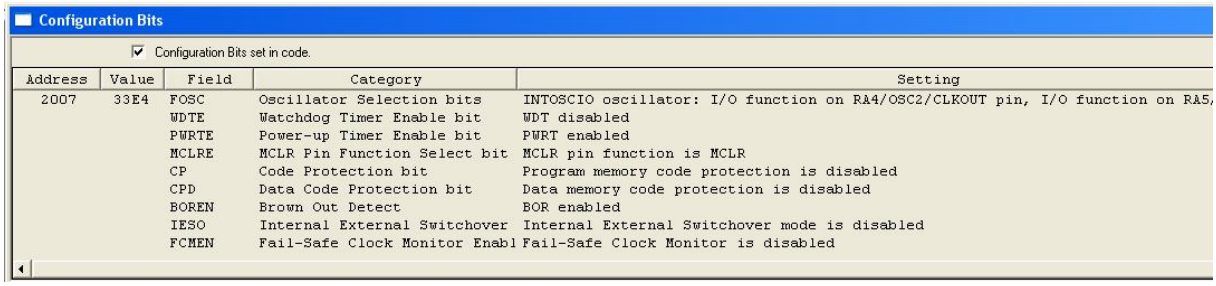
A PIC esetében minden láb több funkcióval bír és a regiszterekkel állíthatjuk be, hogy éppen melyik funkciója szerint használjuk. Ez a láb egyben oszcillátor bekötési lábként (OSC1) is használható. Nem arra állítottuk be véletlenül? Még nem tudjuk, mert nem

foglakoztunk részletesen a konfigurációs bitekkel. Menjünk ismét a Configure/ Configuration Bits menüpontra. Az első sort nézzük meg:

EXTRC oscillator: External RC on RA5/OSC1/CLKIN/CLKOUT

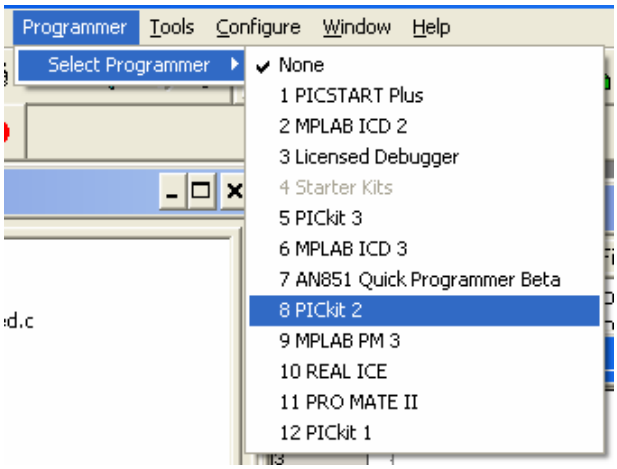
Tehát az RA5 bizony külső RC oszcillátorra van állítva. Hogy miért nem GPIO5 van írva? Mert ez egy sajtóhiba, értsünk az RA5 helyére GPIO5-öt.

Állítsuk be a konfigurációs biteket az alábbiak szerint (45. ábra):



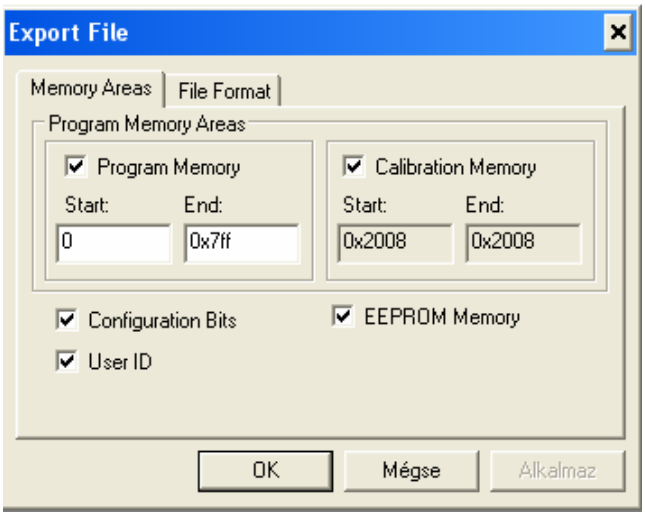
45. ábra

Ha most futtatjuk a programot, az eredmény 0x037 lesz, ami már megfelel az előbbiek figyelembe vételével a várakozásainknak. Most jöhet a programnak a PIC-be égetése. A Programmer/ Select Programmer/ PICkit2 menüvel kijelöljük a PICkit2 programozót (46. ábra). Ha nekünk más programozónk van, akkor azt. Ne feledjük előtte a Debuggert „None”-ra állítani, mert a szimuláció és az égetés egyszerre nem megy!



46. ábra

Ha az égető csatlakoztatva van a géphez és a megfelelő PIC bele van helyezve az égetőbe, akkor ki tudjuk vinni a PIC-be a programot. Már most felhívom a figyelmet, hogy ha a munka könyvtárunkban kapott .hex fájlt akarjuk bevinni a PIC- be más programmal, mint az MPLAB, akkor előtte a .hex fájlt a File/Export File menüvel vigyük ki úgy, hogy a konfigurációs bitek kiírását is állítsuk be. Ha ezt nem tesszük meg, akkor az égetés során az égető programmal be kell állítanunk ezeket a biteket (47. ábra).



47. ábra

A disassembly lista

Közben néhány egyéb dolgot is megnézhetünk az MPASM segítségével. A View/Disassembly Listing bekapcsolásával megjelenik egy úgynevezett disassembly lista. A programunkat a C fordító lefordította gépi kódra, majd ebben a listában láthatjuk a kész program assembly nyelvű változatát. Közben tisztázzunk két fogalmat: a gépi kódokat közvetlenül használó nyelv neve assembly, az ezt tényleges gépi kódra (bináris számok sorozata) fordító program neve assembler.

Ezek a korrekt elnevezések, de gyakran tapasztalhatjuk azt a nyelvi egyszerűsítést, hogy a nyelvet is assemblerként emlegetik. Nézzük meg a lista egy részletét (48. ábra):

```
---   C:\Minta\HTC_PIC12F683_1led.c   ------------------
1:                  #include <htc.h>
2:                  void main()
3:                  {
4:                  CMCON0=0x07;
   7F6     3007     MOVLW 0x7
   7F7     1283     BCF 0x3, 0x5
   7F8     0099     MOVWF 0x19
5:                  ANSEL=0;
   7F9     1683     BSF 0x3, 0x5
   7FA     019F     CLRF 0x1f
6:                  TRISIO = 0; // PORT legyen output
   7FB     0185     CLRF 0x5
7:                  GPIO = 0xFF; //  port minden lábát 1
   7FC     30FF     MOVLW 0xff
   7FD     1283     BCF 0x3, 0x5
   7FE     0085     MOVWF 0x5
8:                  }
   7FF     2800     GOTO 0
```

48. ábra

A 4. sorban megjelenik az eredeti C nyelvű program: CMCON0=0x07. Ezt követi 3 számozatlan sorban ennek az assembly nyelvű megfelelője:

```
MOVLW        0x7
BCF        0x3,0x5
MOVWF       0x19
```

Az assembly utasításokat megtaláljuk számos irodalomban, én itt csak az indító lökést szeretném megadni ezek értelmezéséhez. A MOV-val kezdődő utasítások az adatmozgató (move) utasítások. Az utasítások következő betűi az adatmozgatás irányát adják meg: honnan, hová. A mikrovezérlőt is úgy kell elképzelni, mint egy bővített zsebszámológépet, aminek van egy, a kiírást tartalmazó regisztere (ezt akár munkaregiszternek is nevezhetnénk) és van néhány memóriája. Adatokat a memóriák és a „munkaregiszter” között bármikor átvihetünk, de az egyik memóriából a másikba csak úgy, hogy az egyikből behozzuk a „munkaregiszterbe”, majd onnan kivisszük a másikba. Ha műveletet végzünk, akkor a „munkaregiszter” és a begépelt szám között végződik el a művelet és az eredmény (többnyire) a “munkaregiszterben” képződik. Valamennyire hasonlóan működik a PIC is. A „munkaregiszter” neve ténylegesen munkaregiszter és W (Work) a jele. A PIC esetében természetesen nem tudjuk az

adatokat begépelni, hanem a programba írjuk bele. Az így beírt adatok jele L (literal). Ezek alapján a MOVLW 0x7 utasítás azt jelenti, hogy vigyük (MOV) az utasításba írt 0x7 számot, mint literalt (L) a munkaregiszterbe (W). Vegyük a következő utasítást: BCF 0x3, 0x5. Ha az irodalomban megnézzük a PIC assembly utasításait, a BCF a bit orientált műveletek között található és azt jelenti, hogy adott esetben a 0x3 címen lévő memóriának az 5. bitjét törli (Bit Clear File register). A PIC esetében az egyes memória cellákat a – gyakran félreértésre okot adó – fájl regiszter (file register) névvel illetjük és a teljes memóriában elfoglalt helyük (sorszámuk) alapján azonosítjuk. Tehát ismételve: töröljük a 3. memória 5. bitjét. Ne feledjük, hogy a bitek sorszámozása 0-val kezdődik, tehát az első, vagyis a legkisebb helyiértékű bit neve „0" bit. A fájl regiszter elnevezésnél fontos kihangsúlyozni, hogy semmi köze a PC-n fájlként megismert adathalmazokhoz. Itt a memória egyes elemeit jelenti. De mi is található a PIC 3-as sorszámú memóriájában? Nézzük meg a korábban már használt listánkban:

STATUS ..... EQU H'0003'

Tehát a status nevű adat, mint 1 bájt. Ennek az egyes bitjei tartalmazzák például a W munkaregiszter tartalmának a tulajdonságait (pl. hogy nulla-e, ez a Z mint Zero bit) Ennek a szerepét és használatát majd meg kell tanulnunk. Mint már láttuk, a memória két „bank" alatt van felsorolva: Bank0 és Bank1. Ahhoz, hogy a kívánt memóriát el tudjuk érni, azt a bankot kell kiválasztanunk, amelyikben a kívánt memória (fájl regiszter) található. Ha a status bájt 5. bitjét nullázzuk, akkor a bank0 lesz kiválasztva, ha 1-re állítjuk, akkor a bank1. Tehát ez a bit egy bank kiválasztó bit. Más típusú PIC-ek esetében is szükséges a memória bank kiválasztása, de nem biztos, hogy pont így. Jelen esetben a bank0 lesz kiválasztva, a következő utasítás előkészítéseként. Nézzük is a következő utasítást: MOVWF 0x19. A korábbiak alapján ez azt jelenti, hogy vigyük (MOV) a W regiszter tartalmát az utasításban található című fájl regiszterbe (F). Címként a 0x19 szerepel. Ezután keressük ki, hogy a 19-es címen mi található: CMCON0..... EQU H'0019'

azaz a W tartalmát a CMCON0-ba írjuk. Ismételjük át: bevittünk 7-et a munkaregiszterbe, kiválasztottuk azt a bankot, ahol a cél memória (CMCON0) van, majd a W tartalmát kivittük a CMCON0-ba. Ez valóban a CMCON0=0x07 utasításnak felel meg. Ugyanígy értelmezhetjük a többi utasítást is. A BSF 0x3, 0x5 a BCF ellentettje, tehát nem kinullázza, hanem beállítja (Set), vagyis 1-et helyez a 3. memória, mint fájlregiszter 5. bitjébe. Tehát a bank1-et választjuk ki, mert a következő utasítás

majd az ANSEL- re vonatkozik. Az ANSEL címeként 0x9F van megadva. A következő CLRF 0x1f utasításban a cím viszont 0x1F. Láthatjuk, hogy a bank1 esetén az utasításban lévő címhez 0x80-at hozzá kell adni, hogy a felsorolt címet kapjuk. 0x80+0x1F=0x9F. A CLRF (CLeaR File register) a megadott memóriát kinullázza, azaz ANSEL=0. És így tanulhatjuk tovább az assembly nyelvet, hogy a későbbiekben assembly nyelvű programot is tudjunk írni.

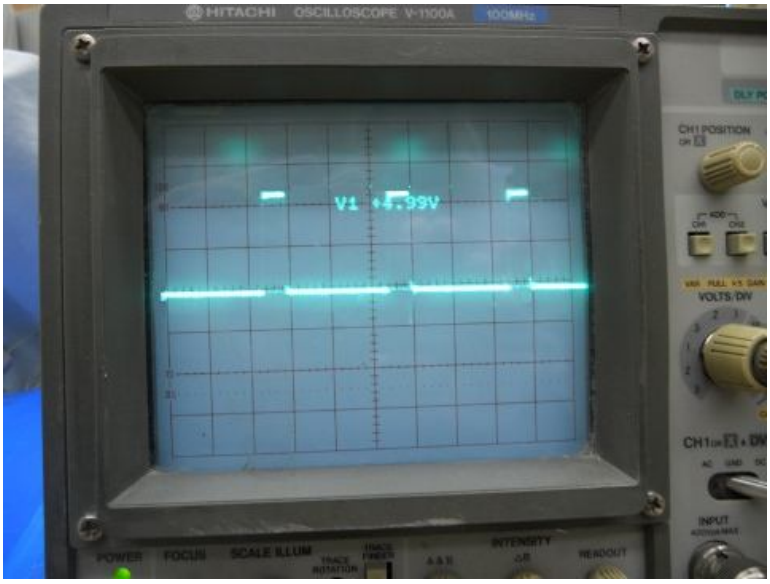
A program módosítása

Ha ezt a programot égetjük be a PIC-be, az eredmény nem lesz túl látványos. Valamelyik GPIO láb és a föld közé kötött LED világít. Ezért módosítsunk egy kicsit a programon!

```
#include <htc.h>
void main(){
        CMCON0=0x07;
        ANSEL=0;
        TRISIO = 0;   // PORT legyen output
        while (1){
            GPIO = 0xFF; //port minden lábát tegyük 1-be, azaz közel 5 V-os állapotba.
            GPIO = 0x00; //port minden lábát tegyük 0-ba, azaz közel 0 V-os állapotba.
        };
};
```

Itt az ideje néhány szót ejteni a C nyelv elemeiről. Természetesen a C nyelv leírására sem vállalkozhatok, hiszen sokszáz oldalas könyvekben van leírva. Csak a legelemibb utasításokról szólok. A program elején a #include utasítással adjuk meg a felhasználandó – előre megírt – programkönyvtárat. Ez fordítóprogramonként más és más is lehet. Ehhez a fordítóhoz a htc.h nevű fájl tartozik. A tényleges program utasításai a void main () {} kapcsos zárójelében helyezkednek el. Az utasítások tetszőleges sorváltást és szóközt tartalmazhatnak, azt a fordító nem veszi figyelembe. A // után a megjegyzést írtam. Szintén figyelmen kívül hagyja a fordító. Minden utasítást ; (pontosvessző) zár le. Léteznek ciklus utasítások, mint pl. a while. A while utáni {}-ben lévő utasítások addig ismétlődnek folyamatosan, amíg a while utáni () zárójelben lévő kifejezés értéke nulla nem lesz. Adott esetben a ()- be egyet írtunk, tehát soha nem lesz nulla. Ez így végtelen ciklus, azaz addig ismétlődik, amíg a PIC tápfeszültségét ki nem kapcsoljuk. Ez egy szokásos megoldás. Fordítsuk le a programot és égessük be a PIC-be. Helyezzük be egy áramkörbe, ahol a PIC valamelyik GPIO lába és a föld

(zsebtelep negatív sarka) közé egy LED-et kötöttünk. Azt várjuk, hogy a LED villogni fog, hiszen ismétlődően hol 0, hol 1 kerül a lábra kiírásra. Ezzel ellentétben azt tapasztaljuk, hogy a LED folyamatosan világít. Mi a hiba? Debugger-ben megfelelőnek tűnik, mégsem villog. Pedig villog, csak nem látjuk, mert nagyon gyorsan teszi. Ha oszcilloszkópon megnézzük, láthatjuk, hogy a PIC lába hol alacsony, hol magas feszültség szinten van (49. ábra).



49. ábra

Hogy a LED-del is lássunk valamit, tegyünk be a programunkba némi időhúzó, lassító utasítást. Ez is lehet ciklus, pl. „for” ciklus:

```
#include <htc.h>
void main(){
        CMCON0=0x07;        //analog kikapcsolás
        ANSEL=0;
        TRISIO = 0;         // PORT legyen output
        int i;
        while (1){
            GPIO = 0xFF; //port minden lábát tegyük 1-be, azaz közel 5 V-os állapotba.
            for (i=1; i<0xffff; i++)
                    ;
            GPIO = 0x00; //port minden lábát tegyük 0-ba, azaz közel 0 V-os állapotba.
            for (i=1; i<0xffff; i++)
                    ;
        };
};
```

Deklaráltunk egy egész (int) típusú „i” nevű változót, ez lesz a ciklusváltozó. A „for” ciklus az i-nek 1-et ad, majd végrehajtja a következő utasítást a pontosvesszőig. Itt most a „for” zárójele és a pontosvessző között nincs semmi, tehát a semmit hajtja végre ☺, de azt 0xffff-szer. Egyesével növeli az i értékét (i++) addig, amíg az i<0xffff feltétel teljesül. Utána továbbmegy a következő utasításra. Ha ezt a programot égetjük a PIC-be, már látható sebességgel fog villogni. A for (i=1; i<0xffff; i++) sor helyett írhatjuk a _delay(1000000) utasítást is. Ez utóbbi egymillió gépi ciklusidőt várakozik. Elvileg léteznek a __delay_ms(x) __delay_us(x) függvények is, melyeknek miliszekundumban illetve mikroszekundumban kell megadni a várakozási paramétert, de ezek a Lite verzióban nem működnek. A panelről – melybe belehelyezzük a PIC-et – egyelőre nem beszélünk, majd egy későbbi fejezetben.

Megvalósítás PICBASIC nyelven

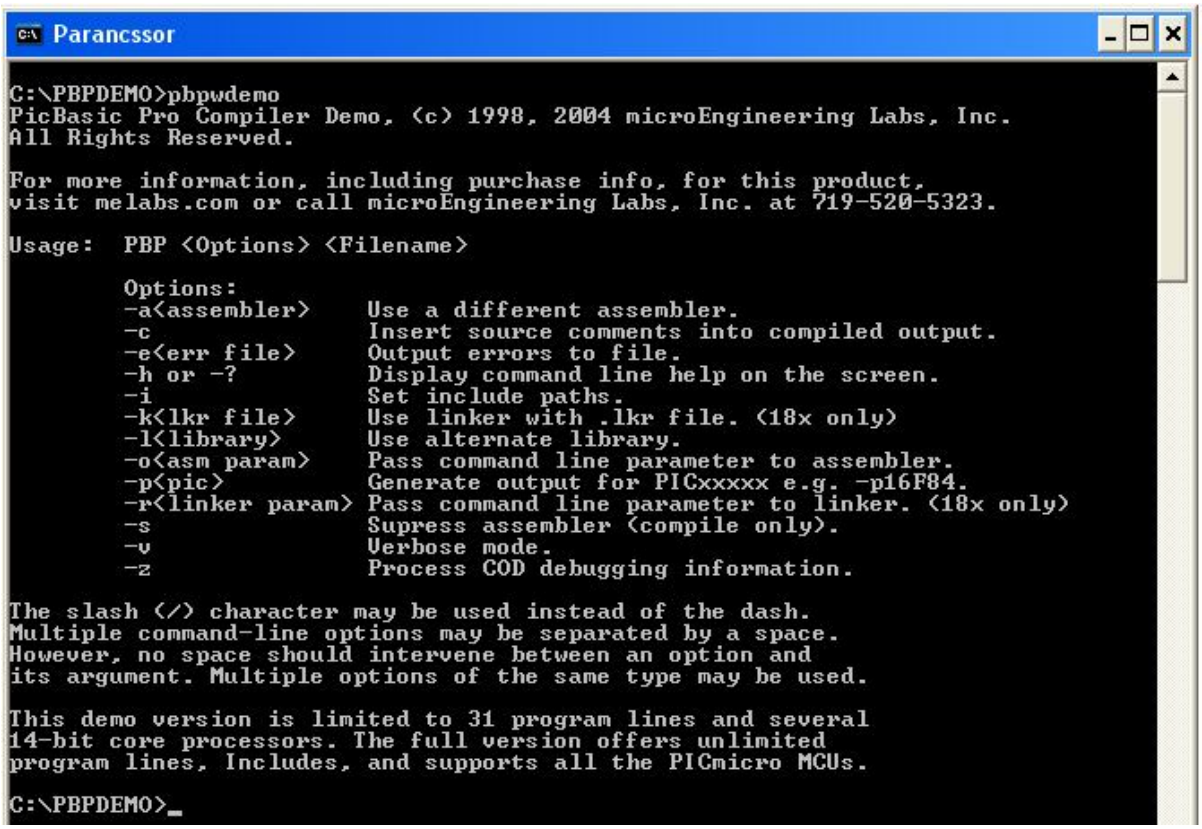
Ha idáig eljutottunk, nézzük meg, hogyan tudjuk megvalósítani ugyanezt más programozási nyelven. Sokan kedvelik egyszerűsége és könnyű használhatósága miatt a PICBASIC nyelvet, annak is a MELABS cég által gyártott verzióját. Csökkentett tudású, de ingyenes demó verzió is létezik. Töltsük le a cég oldaláról a demó verziót és a kézikönyvet! [25]

Installálás után kapunk egy PBPDEMO nevű könyvtárat. A gyártó honlapján a DEMO verzió által is támogatott processzorok között a PIC12F683 elsőként van felsorolva. Hibázunk, ha ezt el is hisszük, de erről majd később beszélünk. Tegyünk úgy, mintha elhittük volna ☺.

A következő BASIC programmal tervezzük kipróbálni.

```
CMCON0=7; ‘analog kikapcsolas (ne hasznaljunk ekezetet)
ANSEL=0;
TRISB=%00000000  'a binaris formaban megadott szam jelolese

CIKLUS:
GPIO.0=1
PAUSE        500
PORTA.0=0
PAUSE        500
GOTO CIKLUS

END
```

Ismét nem részletezem a teljes használatot, hiszen letöltöttük a kézikönyvet, abban minden érthetően és részletese le van írva. Most is használhatjuk a regiszterekeknek azokat a neveit, amit a Microchip adott nekik. A regiszterek egyes bitjei külön is „megcímezhetők”. Ami a 12F683-nál GPIO, az itt PORTA. A PORTA.0 a PORTA regiszter 0. bitjét jelenti. A PICBASIC-ben van egy PAUSE nevű utasítás, ami várakozást jelent. A PAUSE 500 utasítás fél másodperces várakozást (500 miliszekundum) eredményez. A programba címkéket rakhatunk, amit a kettőspont jelöl. A cimkét használhatjuk a GOTO utasításban. Itt így valósítjuk meg a végtelen ciklust. A PICBASIC fordítót célszerű az MPLAB-tól függetlenül, önállóan használni. A fenti programot jegyzettömbbe gépeljük be és a PBPDEMO könyvtárba mentsük el .BAS kiterjesztéssel. Ne adjunk hosszú nevet, mert DOS ablakot használunk. Legyen pl. LED1.BAS. Nyissunk egy DOS ablakot és lépjünk a PBPDEMO könyvtárba. Itt található a PBPWDEMO. Ezt használjuk. Ha csak a program nevét gépeljük be (utána persze enter), akkor egy help listát kapunk, amiből láthatjuk a használatát.



50. ábra

Ezek alapján a pbdwdemo –p12F683 LED1.BAS parancsot kell begépelnünk. Próbáljuk ki! A következő hibajelzést kapjuk: ERROR: Processor „12F683” not supported in

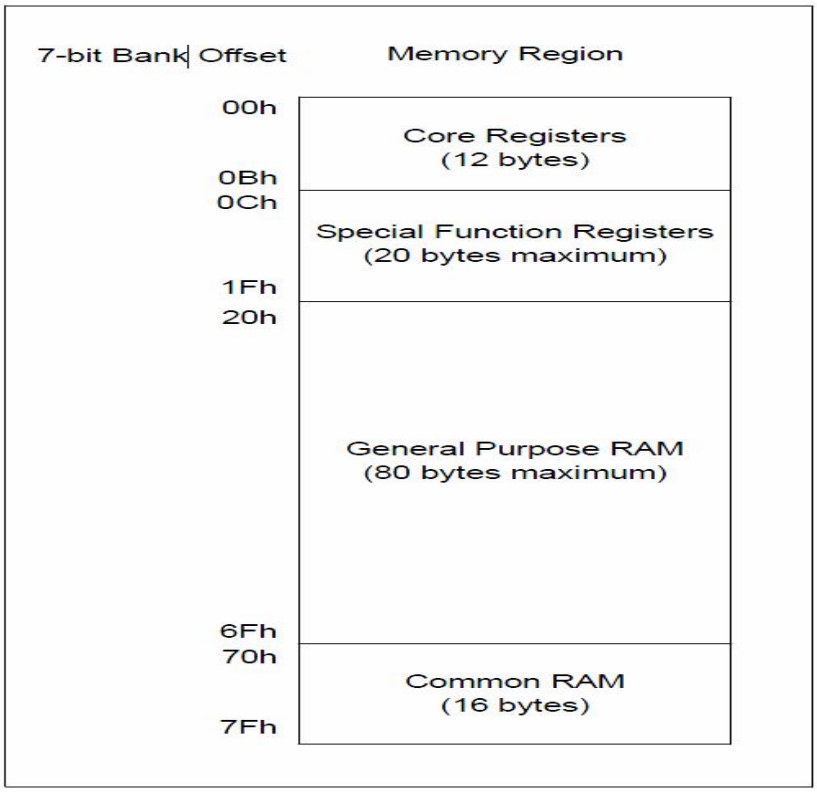
demo version. Tehát a demó verzió nem hajlandó a PIC12F683 PIC-re fordítani. Pedig a honlapon egyértelműen az olvasható, hogy használható erre a processzorra is. Ha egyelőre még nem akarjuk megvenni a teljes verziót, akkor keressünk egy hasonló processzort és fordítsuk arra. Hasonló a PIC16F628. Fordítsunk erre. Ezzel nem sértjük a licenszet, mert a honlapon egyértelműen ott van, hogy a demó verziót is használhatjuk a 12F683-ra. Az internetről ne töltsünk le kalóz verziót, mert az illegális! A PIC16F628 használatához kissé át kell írni a programot, mert PORTA az első kimeneti port neve és más az analóg kikapcsolás is. Az analóg kikapcsolást mindig nézzük meg az adott PIC leírásából, mert változik!

```
CMCON=0
TRISA=%00000000  'a binaris formaban megadott szam jelolese

CIKLUS:
PORTA.0=1
PAUSE        500
PORTA.0=0
PAUSE        500
GOTO CIKLUS

END
```

A kapott LED1.HEX fájlt beégethetjük a PIC12F683-ba is, fog működni. A konfigurációs biteket viszont az égetés során a PIC12F683-nak megfelelően kell beállítani. Természetesen a PIC16F628-cal is működik, ha van ilyen PIC a fiókunkban. Semmiképp ne vásároljunk ilyet, mert a mai viszonyok között irreálisan drága.

A PIC16F1939 használata

Az előzőekben bemutattam a buktatókat. Szerencsére a dolog nem ennyire bonyolult, ha magasabb kategóriájú PIC-et választunk. Az előbbi programot berakhatjuk pl. egy PIC16F1938-ba vagy egy PIC16F1939-be. Ilyet vásárolni is érdemes. Ez már egy újabb széria. A memória kiosztása is sokkal logikusabb (51. ábra). A tartomány elején van egy „Core Registers” tartomány, ami minden lapon közös. Ugyanígy közös a lap végén lévő 16 bájt is.



51. ábra

A PIC 18-a sorozat.

Eddig a C fordítók közül csak a HI-TECH C-t tudtuk használni. Ha továbblépünk a PIC18-as sorozatra, akkor választhatunk a HI-TECH és az MPLAB C fordítója közül. Ha még egy szinttel tovább lépünk és a PIC24-es sorozatot használjuk, akkor már a HI-TECH nem használható. Ha lehetőségünk van rá, célszerű ezt a sorozatot használni. 16 bites adatokkal dolgozik, nagyobb a memóriája, gyorsabb és logikusabb a szervezése. A következő szint a PIC32 lenne, de az már alapjaiban különbözik az eddigiektől, ezért azzal ebben a dolgozatban nem foglalkozom.

Következő példánk legyen a PIC18F43K22 típusú 40 lábú PIC. Töltsük le a Microchip honlapjáról a kézikönyvét és látni fogjuk, mennyi mindent tud. Itt erre is csak egy egyszerű mintapéldát mutatok be. A buktatókat már korábban bemutattam, így most csak a ténylegesen jó programot írom le. Találjunk ki egy feladatot. Például legyen egy zöld és egy fehér LED valamint egy kapcsoló. Ha a kapcsoló egyik állásban van, akkor felváltva villogjon a két LED, a másik állásban pedig együtt. Nézzük meg a PIC rajzát

és válasszunk lábakat. Sok lába van, bőven válogathatunk. A példában a LED-eket az RB0 és RB1 lábakra tettem. Tehát a B port 0. és 1. bitjei működtetik a LED-eket. A kapcsolót az RB2-re tettem. Azért a B portra célszerű kötni a kapcsolót, mert ennek a portnak a lábaihoz belső felhúzó ellenállások vannak kötve. Ezért nem szükséges a kapcsoló egyik végét egy ellenálláson keresztül a pozitív tápfeszültséghez kötni. A következő programot használjuk:

```
#include <p18cxxx.h>
void delay (void)
{
        // keszítsunk sajat keslelteto fuggvenyt
        unsigned long i;
        for (i = 0; i < 5000 ; i++);
}

void main (void){

        ANSELB=0;               // PORTB digitalis legyen
        TRISB = 0b00000100;     //0bxxxxx100 utolso 2 bit output
                                // RB2 input
                                // tobbi most lenyegtelen

        EnablePullups();        // PORTB felhuzo ellenallasok bekapcs.
                                //  INTCON2bits.RBPU=0;   is ugyanez
        while (1){
                //if ( (PORTB & 0b00000100) <> 0 ){
                if ( PORTBbits.RB2){
                        PORTB=0b00000001;       // vagy PORTBbits.RB0=1
                                                // PORTBbits.RB1=0
                        delay();
                        PORTB=0b00000010;       // vagy PORTBbits.RB0=0
                                                // PORTBbits.RB1=1
                        delay();
                }
                else
                {
                        PORTB=0b00000000;       // vagy PORTBbits.RB0=0
                                                // PORTBbits.RB1=0
                        delay();
                        PORTB=0b00000011;       // vagy PORTBbits.RB0=1
                                                // PORTBbits.RB1=1
                        delay();
                };
        };
}
```

Az önálló tanulás segítése céljából részletezzük egy kissé a programot! Az #include <p18cxxx.h> utasítás minden , PIC18 sorozatú PIC-et használó program elejére kell. Készítünk egy saját késleltető függvényt egy üres FOR ciklussal. A neve legyen delay. Előtte a void azt jelöli, hogy ennek a függvénynek nincs visszatérési értéke. Pascalban ennek procedure a neve. A zárójelben lévő void mutatja, hogy ez egy paraméter nélküli függvény. A C nyelvű programban mindig a „main” nevű függvény kezd el futni. Ebben az ANSELB=0 sor a B porton kikapcsolja az analóg funkciókat, hogy digitális célra használni lehessen. Erről a leírást a PIC18(L)F2X/4XK22 Data Sheet I/O ports fejezetében találjuk meg. Használhatunk egy EnablePullups függvényt az összes felhúzó ellenállás bekapcsolására. Honnan tudhatjuk, hogy milyen függvényeket használhatunk? Nyissuk meg az <ahová a programot telepítettük mappa>/mplabc18/verziószám/doc könyvtárban a „MPLAB-C18-Libraries_51297f.pdf” fájlt. Ebben végignézhetjük a használható függvényeket és rövid leírásukat. A felhúzó ellenállásra („pull up”) vonatkozó információkra vagyunk kíváncsiak. Keressünk rá a pullup szóra! Ezt (is) találjuk:

EnablePullups
Function: Enable the internal pull-up resistors on PORTB.
Include: portb.h
Prototype: void EnablePullups(void);
Remarks: This function enables the internal pull-up resistors on PORTB.
File Name: pullen.c

A függvény neve: EnablePullups. A file neve: pullen.c. Ha további információkra vagyunk kíváncsiak erről a függvényről, keressük meg a pullen.c fájlt a windows keresőjével! Az <ahová a programot telepítettük mappa>/mplabc18/verziószám/src/pmc_common/PORTB könyvtárban található. Nyissuk meg! Láthatjuk, hogy a törzsében az INTCON2bits.RBPU=0; utasítás van. Ez engedélyezi a felhúzó ellenállásokat. Persze az EnablePullups helyett az INTCON2bits.RBPU=0; utasítást is írhatjuk, az eredmény ugyanaz. Szintén felmerül önálló tanulás során a kérdés: hogyan tudhatjuk, hogy ezt így kell írni: INTCON2bits.RBPU?

Menjünk az <ahová a programot telepítettük mappa>/mplabc18/verziószám/h könyvtárba és keressük meg az általunk használt PIC információit tartalmazó fájlt. Ez jelen esetben a p18f43k22.h. Nyissuk meg ezt a fájlt! Keressünk rá az INTCON2bits szóra! Ezt találjuk:

```
extern volatile near unsigned char INTCON2;
extern volatile near union {
  struct {
    unsigned RBIP:1;
    unsigned :1;
    unsigned TMR0IP:1;
    unsigned :1;
    unsigned INTEDG2:1;
    unsigned INTEDG1:1;
    unsigned INTEDG0:1;
    unsigned NOT_RBPU:1;
  };
  struct {
    unsigned :7;
    unsigned RBPU:1;
  };
} INTCON2bits;
```

Ebben láthatjuk, hogy milyen struktúrát használ a fordító. A C nyelv struktúráinak leírását tankönyvekben megtaláljuk.

Az if utasítás feltétel részében a PORTBbits.RB2-t vizsgáljuk. Ha 0, akkor hamisnak számít, egyébként igaznak. A feltételt persze írhattuk volna a (PORTB & 0b00000100) <> 0 formában is. Ez utóbbi variációban a PORTB és egy „maszk” között „bitenkénti és” műveletet végzünk. Mint tudjuk, az eredmény a PORTB-nek egy olyan módosítása lesz, melyben nullázódnak azok a bitek, ahol a maszkban 0 van, és változatlanul maradnak azok, ahol a maszkban 1 van. A PORTBbits.RB2-re vonatkozó információkat az INTCON2bits-hez hasonlóan kereshetjük meg. Ha a programot lefordítjuk, PIC-be írjuk és egy megfelelő áramkörbe helyezzük, akkor elvárásaink szerint működik. A PIC24-es sorozatról a minta paneloknál írok.

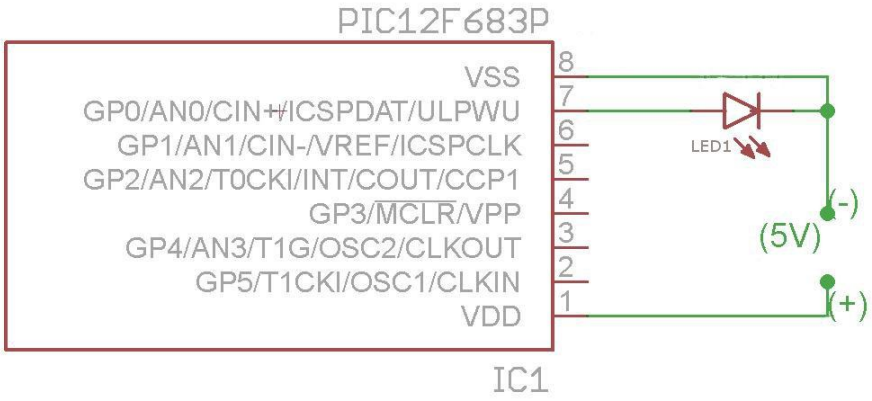
Minta Panelok

A következőkben bemutatok mintaként néhány általam megépített panelt, amit az oktatáshoz készítettem és témavezetőm is használja az óráján.

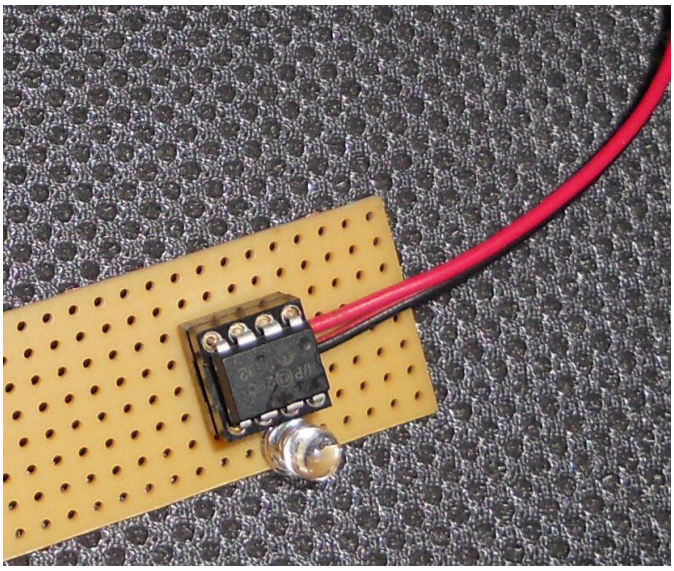
Elsőként egy PIC12F683-al megépített áramkört mutatok be annak szemléltetésére, hogy milyen kevés alkatrésszel is készíthető működőképes áramkör. Ez az áramkör semmi mást nem csinál, mint egy LED-et villogtat. Áramforrásként célszerű egy 4.5 V-os zsebtelepet használni. Az (52. ábra) látható a kapcsolási rajz, az (53. ábra) a megépített áramkör fényképe.

Korábban leírtam, hogy ha a PIC tápfeszültségénél nagyobb feszültséggel működő fogyasztót akarunk működtetni, akkor kapcsoló áramkört kell alkalmaznunk. Példaként egy 12 V-os izzó vezérlését mutatom be mind negatív oldali (low side) (54. ábra) és (56. ábra jobb oldala) mind pozitív oldali (high side) kapcsolással (55. ábra) és (56. ábra bal oldala). Ezt követően egy 40 lábú, PIC18-as szériájú vezérlővel mutatok be egy kapcsolót és két LED-et tartalmazó kapcsolást (57. ábra) és (58. ábra).

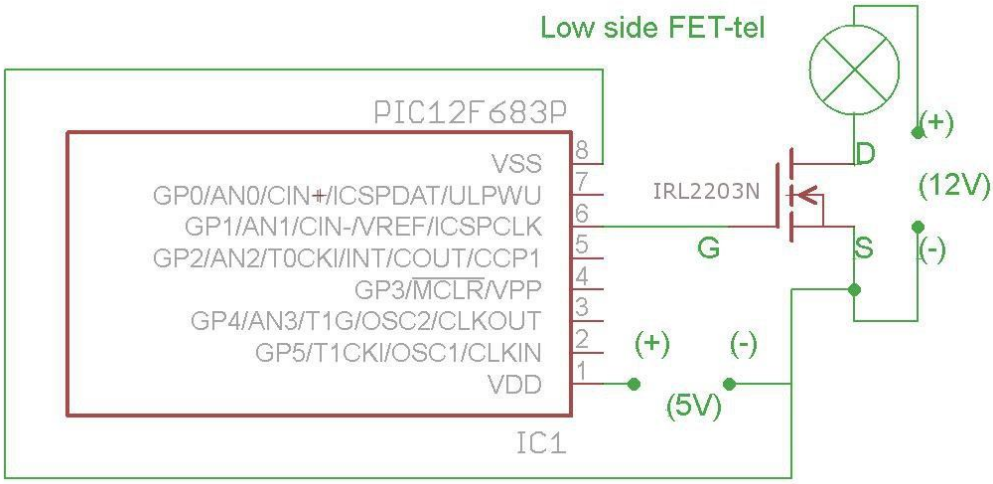
Bemutatok egy PIC24-es szériájú vezérlővel is egy mintapéldát (59. ábra) és (60. ábra). Itt hívom fel a figyelmet, hogy ezek a vezérlők 3.3 V-tal működnek, ezért egy feszültségszabályozó IC alkalmazása is szükséges. A dolgozatom írásakor a Microchip még csak tervezi az 5 V-tal működő 24-es szériájú vezérlőket, de még nem kaphatók. A nagyobb sebességű működés esetén javasolt a VDD (táp áramforrás pozitív sarka) és a VSS (táp áramforrás negatív sarka) közé kerámia kondenzátort helyezni. Bár nélküle is működik, itt ezekkel a 100 nF-os kondenzátorokkal mutatom be. Szintén kibővítettem az MCLR láb bekötését a Microchip által javasolt módon.



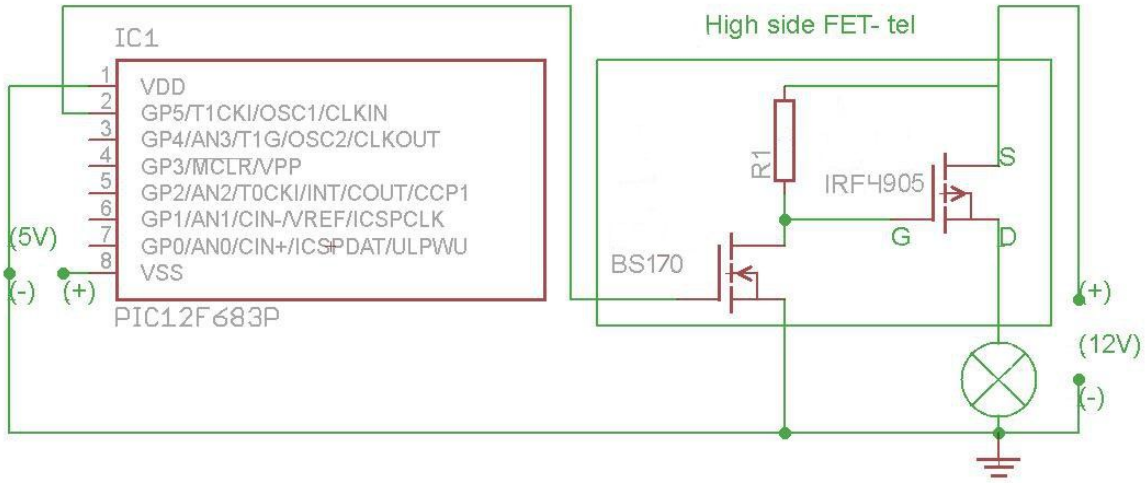
52. ábra



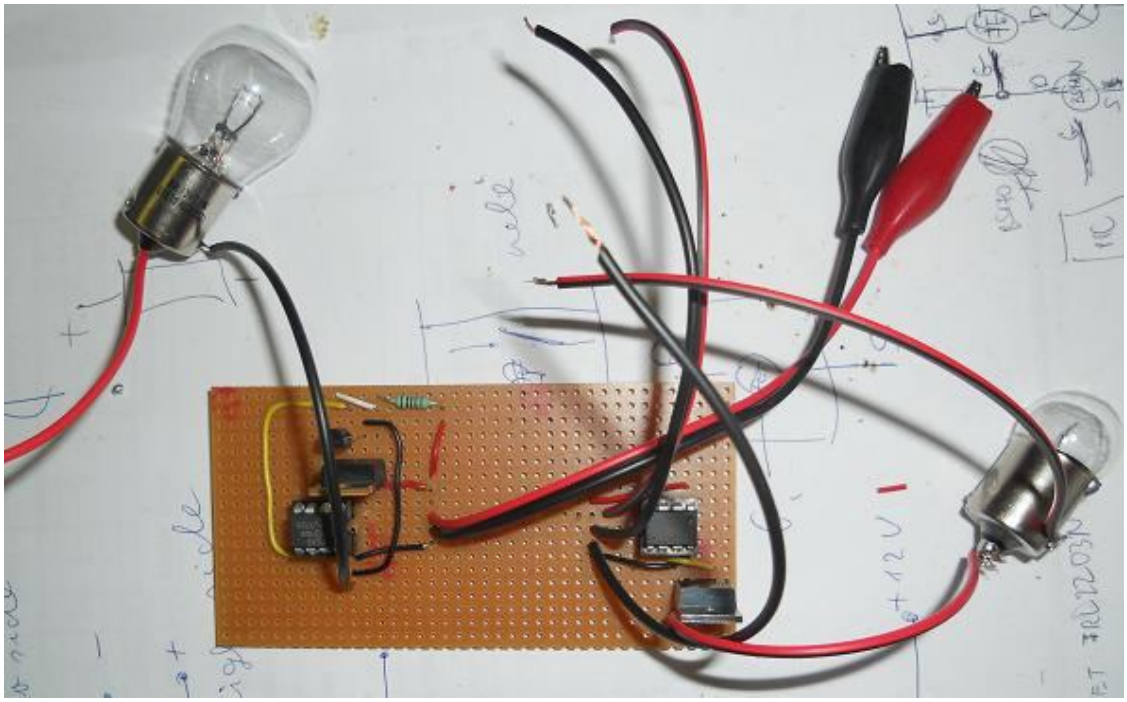
53. ábra



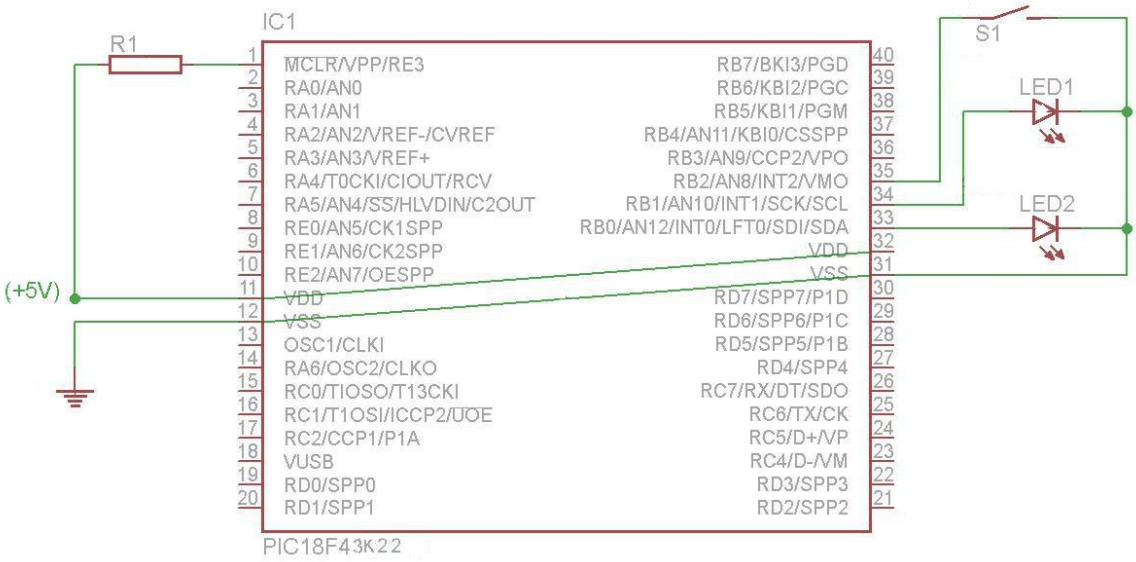
54. ábra



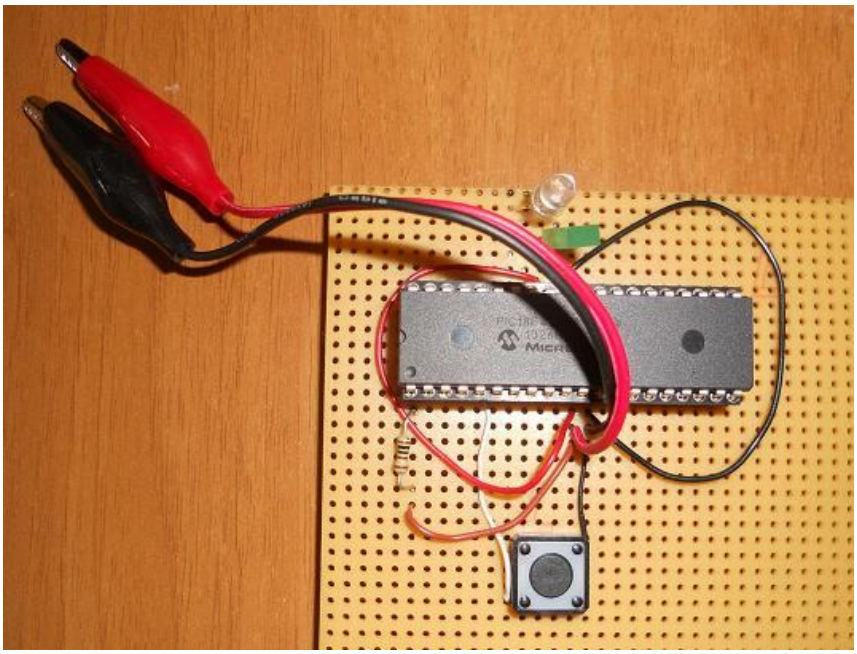
55. ábra



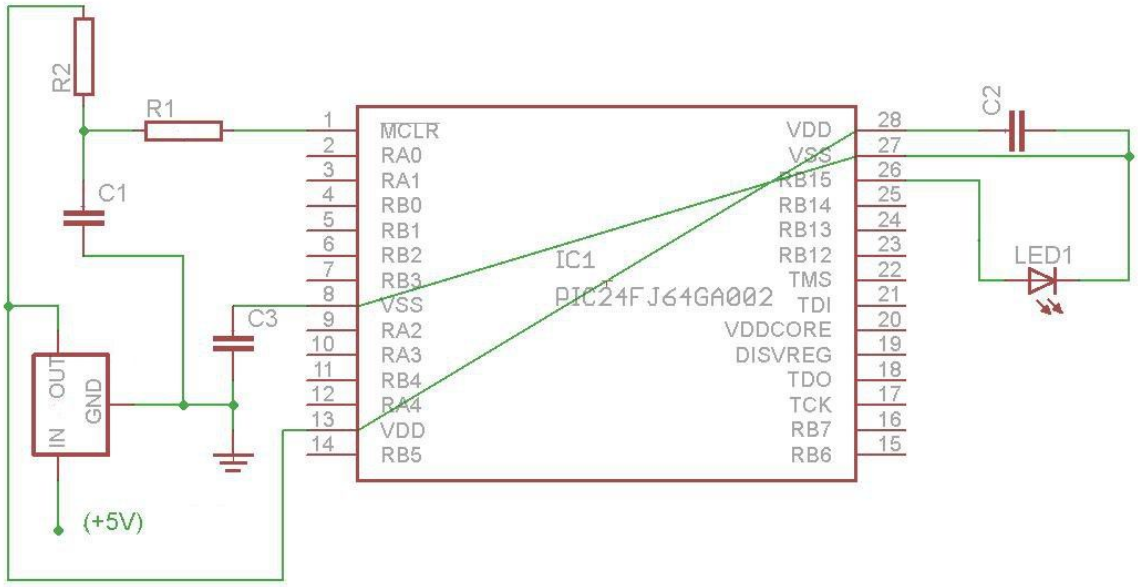
56. ábra



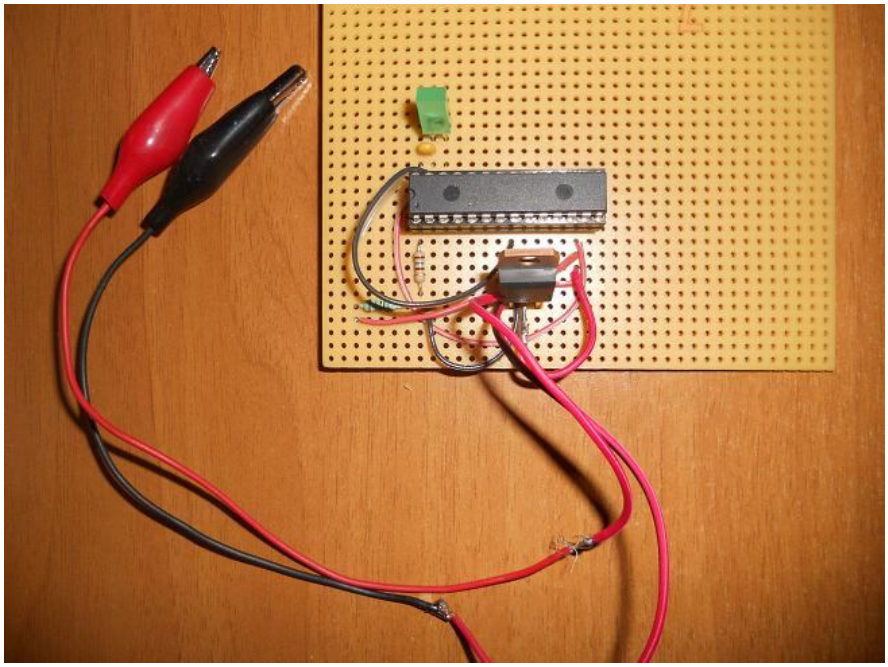
57. ábra



58. ábra



59. ábra



60. ábra

Léptetőmotor

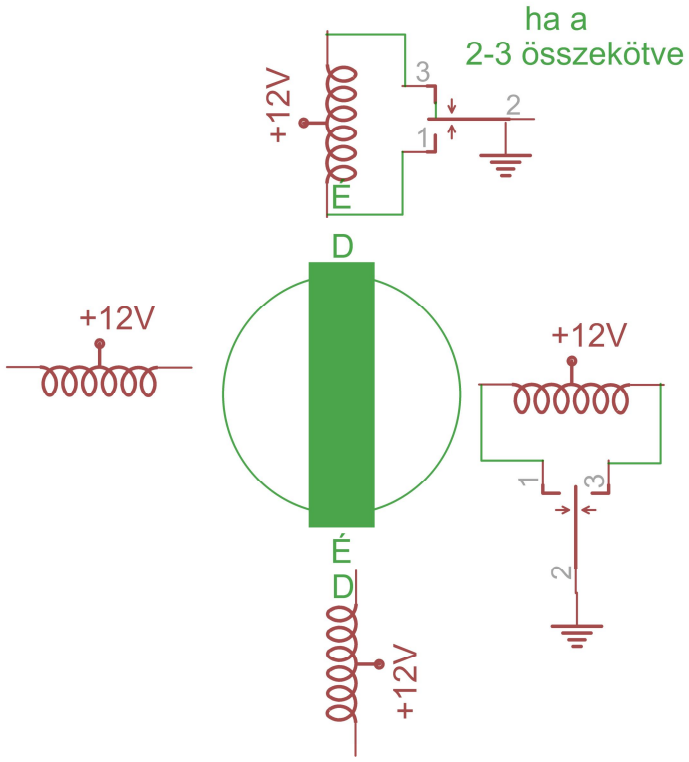
Azt tapasztaltam, hogy a léptetőmotor és a szervomotor programozása látványos és érdekes feladat szokott lenni, ezért röviden ezekről is írok.

A léptetőmotor működési elve vázlatosan

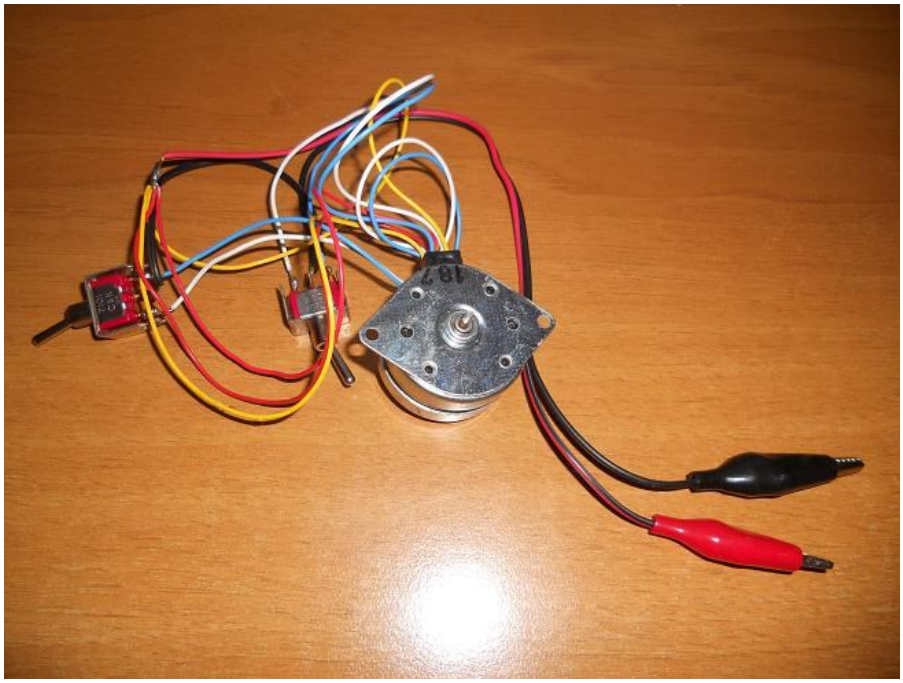
A léptetőmotorban egy állandó mágnes van a motor tengelyéhez rögzítve, valamint tartalmaz két tekercset. Elviekben tekinthetjük tekercs sorozatnak is, mint később látni fogjuk. Nem térek ki a különböző léptetőmotorok ismertetésére, csak az un. unipoláris léptetőmotorról írok, mivel annak a legegyszerűbb a programozása (61. ábra). Az egyszerűség végett a forgó állandó mágnest rúd-mágnesként ábrázoltam, a tekercsek közül pedig kettőt, a legfelsőt és a jobb oldalit ábrázoltam áramkörbe kötve. Mindkét tekercsnek van egy középső kivezetése is, amit a tápfeszültség pozitív sarkához csatlakoztatunk. A tekercsek két vége egy-egy háromállású kapcsoló két szélső kapcsához csatlakozik, míg a kapcsoló középső mozgó része a tápfeszültség negatív sarkához, amit a föld jellel jelöltem. A felső tekercshez csatlakoztatott kapcsolót helyezzük olyan állásba, hogy a középső pontja (2) a (3) ponttal legyen összekapcsolva. Akkor a tekercsben olyan irányú áram folyik, hogy a mágnes felőli vége északi mágneses pólusú lesz, tehát vonzza az állandó mágnes déli pólusát. Ez látható az ábrán. A továbbiakban végezzünk egy gondolatkísérletet! Ez már nincs jelölve az ábrán. Állítsuk a jobboldali tekercs kapcsolóját is olyan állásba, hogy annak is a mágneshez közelebbi oldala legyen északi mágneses pólusú. Ekkor mindkét tekercs egyformán vonzza a mágnes déli pólusát. A mágnes „másik felén” történteket nem részletezem, de ott meg egyformán taszítják, ami azt eredményezi, hogy a két tekercs közé középre fog beállni. Ezek után kapcsoljuk ki a felső tekercsre adott áramot a kapcsolóval. Akkor a mágnes déli pólusa a jobboldali tekercshez fordul. A következő lépés az lenne, hogy a felső tekercs kapcsolóját (1-2) állásba állítjuk, stb. Mivel a valóság kissé más, mint ez az elvi ábra, tovább nem is részletezem. Váltsunk taktikát! Az általam látogatott órán[26] azt láttam, hogy az elméleti fejtegetések helyett sokkal hatékonyabb, szemléletesebb egy megépített kapcsolás, így én is ezt követem. A léptetőmotort gyakran úgy szemléltetik, hogy sok tekercset rajzolnak, hogy sok pólus legyen. A gyakorlatban ezt egy ügyes trükkel mégis összesen két tekerccsel oldják meg, de ezt később részletezem. Gondolkodjunk két tekercsben, ahogy az ábrán is kettőt rajzoltam bekötve. A

megépítéshez a léptetőmotoron és az áramforráson kívül mindössze két háromállású kapcsolóra van szükségünk. Kössük be az ábra szerint (61. ábra). Egy másik ábra (63. ábra) a konkrét megvalósítás fényképét mutatja. A bekötéshez kell néhány szót szólni a tekercsek vezetékeinek azonosításáról. Egyszerű a dolgunk, ha öt vezetéket találunk a motorunkon. Ez a tekercsek két végének a két-két kivezetése, valamint a motorban belül összekapcsolt középső kivezetések Én ezt 12 V-tal jelöltem, mert az amatőr számára ezek a motorok érhetők el olcsón, illetve régi floppy vagy CD meghajtókból ezek szedhetők ki. Az egyes vezetékek általában különböző színűek, de a színek gyártmányonként eltérőek lehetnek. Az azonosításukat ohm mérővel végezzük.

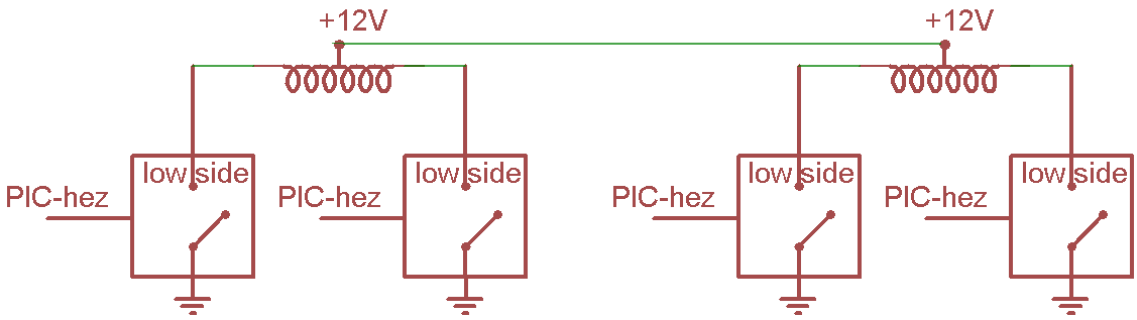
Kissé bonyolultabb a helyzet, ha a két tekercseknek nem az összekötött középső csatlakozási pontja van kivezetve, hanem a tekercs középen „el van vágva”. Ez esetben mindkét tekercs tulajdonképpen két tekercs, tehát összesen nyolc vezetékkel találkozunk. Némi méricskéléssel, leleményes kísérletezéssel ezeket is azonosíthatjuk. Ha minden kész, helyezzünk a léptetőmotor tengelyére egy „mutatót”, hogy lássuk is az eredményt. Váltsuk át valamelyik kapcsolót a másik állásba. A motor tengelye egy picit fordul, azaz a motor „lép” egy lépést. Utána váltsunk a másik kapcsolón, a motor tovább lép. Ha felváltva kapcsolgatjuk a kapcsolókat, a motor tengelye szépen lépeget egyik irányba. Próbáljuk ki azt az esetet, amikor nem felváltva kapcsolgatjuk a kapcsolókat, hanem ugyanazzal a kapcsolóval váltunk egymás után kétszer, a motor forgási irányt vált. Ebben a kísérletben mindent megtanultunk, ami a léptetőmotor programozásához kell. Ha az irodalomban mindenféle fázisdiagramokat láttunk, azt most tegyük félre! A saját kísérletünk tapasztalatai alapján haladjunk tovább. A programozáshoz ezeket a kapcsolókat kell megoldani. Mint látjuk, negatív oldali (low side) kapcsolásról van szó, tehát négy FET és természetesen a PIC elegendő e megoldáshoz (63. ábra). Számos játékos megoldást kitalálhatunk: például egyet balra, kettőt jobbra „tánclépés”, stb. Ha kapcsolót is csatlakoztatunk a PIC-hez, a motorunk „interaktív” lehet. A vázlatosság miatt nem jelöltem a kapcsolási rajzon, de a gyakorlati megvalósítás során a tekercsek induktivitása miatt a tekercsekkel párhuzamosan egy-egy „fordítva” bekötött diódát kell alkalmazni.



61. ábra



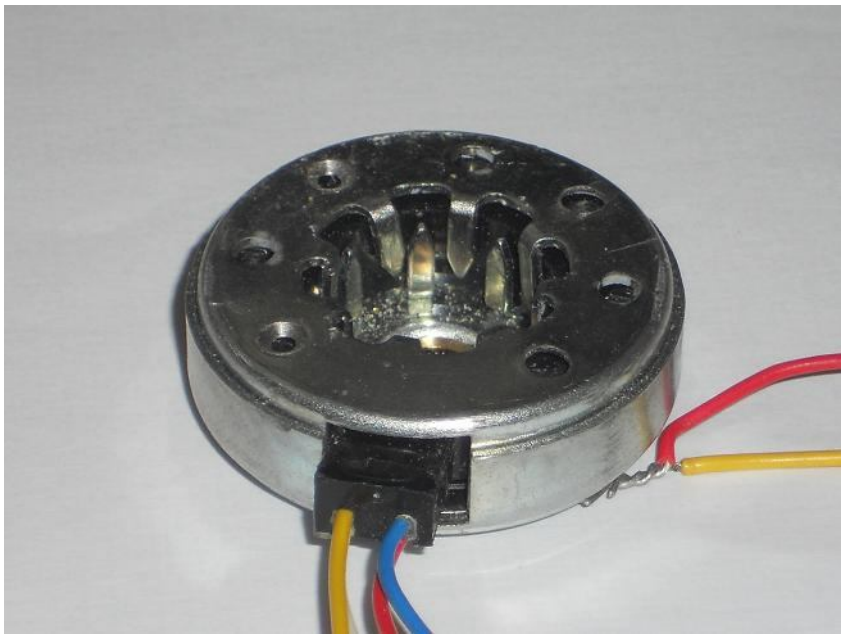
62. ábra



63. ábra

A léptetőmotor tekercseinek gyakorlati megvalósítása

Már említettem, hogy a gyakorlatban összesen két tekercset használnak, mégis sűrűn követik egymást az ellenkező polaritású pólusok, mintha sok tekercs lenne. Ezt a következő megoldással érik el: A tekercset nem az ábrán rajzolt módon helyezik el, hanem körbeveszi a forgó mágnest. A tekercs vasmagja fésűsen kiképezett lemez, amit a tekercs két végén behajlítanak a tekercs belsejébe úgy, hogy a „fésű fogai" egymás közé csússzanak (64. ábra). Így az ellenkező mágnességű pólusok sűrűn követik egymást. A másik – hasonlóan megoldott - tekercset az előbbihez képest fél pólustávolságnyira elfordítva helyezik el.



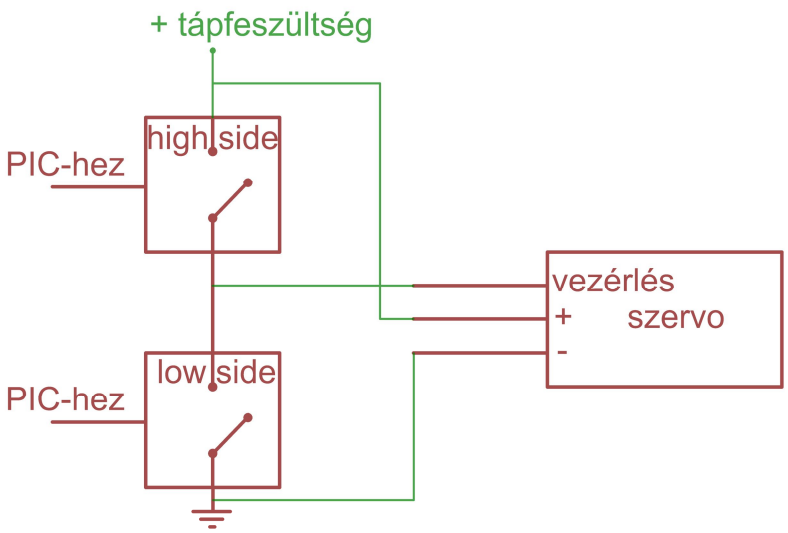
64. ábra

A szervo motor

A korábban részletezett léptetőmotor forgó részével a pillanatnyi helyzetéhez képest el tudunk mozdulni valahány lépést, de nem tudjuk a jelenlegi helyzetét lekérdezni. Tehát csak a jelenlegi helyzetéhez képest relatív helyzetet tudunk megadni, abszolút helyzetet nem.

Pont ellentétes célokat szolgál a szervomotor. Nem tudunk az éppen aktuális helyzetéhez képest relatív elmozdulást megadni, hanem csak a forgórész abszolút helyzetét tudjuk definiálni. A szervomotornak három vezetéke van. A tápfeszültség pozitív sarkát a piros, a negatív sarkát a fekete (néha barna) vezetékhez kötjük. A harmadik vezeték a vezérlő vezeték, amelynek a színe általában fehér vagy sárga. Ehhez a vezetékhez van kötve a motoron belül a motor forgórész tekercsének egyik vége. (A gyakorlatban általában nem közvetlenül, hanem egy kapcsoló áramkört vezérelünk.) A tekercs másik végét egy – a motorban lévő – belső elektronika a tápáramkör negatív illetve pozitív sarkához kapcsolja az alábbiakban leírtak szerint. Alapállapotban a kivezetett vezérlő vezetéket a negatív pólushoz (föld) kapcsoljuk. Ekkor e belső elektronika is ugyanehhez a pólushoz kapcsolja a tekercs másik végét. Így áram a tekercsben nem folyik, a motor forgórésze nem mozdul. A motor forgórésze egy potenciométer tengelyéhez van erősítve, ezért a belső elektronika a potenciométer ellenállásából „tudja” a forgórész aktuális helyzetét. Vezérlésként a külső vezérlő vezetéket az áramforrás pozitív sarkához csatlakoztatjuk rövid ideig. Ezt érzékeli a belső elektronika és abban a pillanatban a tekercs másik végét is ugyanehhez a pólushoz csatlakoztatja. Áram még mindig nem folyik, a forgórész nem mozdul. A belső elektronika a motor forgórészének aktuális helyzetétől, elfordulásától függő ideig tartja a tekercs általa vezérelt végét a pozitív pólushoz csatlakoztatva. Minimum 1 msec-ig (ha a forgórész a bal oldali végkitérésben áll) maximum 2 msec-ig (ha a forgórész a jobb oldali végkitérésben áll). Közbülső helyzetben 1 és 2 msec között. Pl. 1.5 msec, ha középen áll. Ezek után visszakapcsolja a negatív pólushoz. Ha a külső vezetéket is pontosan akkor kapcsoljuk vissza a negatív pólushoz, amikor a belső elektronika tette, akkor továbbra sem folyik a tekercsen áram, a forgórész nem mozdul. Ellenben ha mi előbb vagy később kapcsoljuk vissza a vezérlő feszültséget a negatív pólushoz, mint ahogy a vezérlő elektronika a forgórész pillanatnyi állapotától függően a tekercs másik végénél tette, akkor a forgórész tekercsében egyik vagy másik irányban áram folyik egy

rövid ideig. A forgórész elmozdul. Tehát egy 1 és 2 msec közötti vezérlő impulzust adunk a vezérlő vezetékre. Az impulzus szélessége, azaz a pozitív sarokhoz való csatlakoztatás időtartama határozza meg, hogy hova álljon be a motor forgórésze. A vezérlő impulzusokat 20 msec gyakorisággal szokás adni. A szervomotor tápfeszültsége a kis modellezési célú szervomotoroknál általában 4-6V. A vezérlő feszültség többnyire TTL szintű és kis áramú. Ez a feladat tipikusan jól végrehajtható PIC-kel. Előfordulnak olyan szervomotorok, amelyek a vezérlő vezetéken nagyobb áramot igényelnek, mint amit a PIC képes biztosítani. Ebben az esetben egy pozitív oldali (high side) és egy negatív oldali (low side) kapcsoló áramkör alkalmazása szükséges (65. ábra). Figyeljünk arra, hogy egyszerre véletlenül se legyen mindkét kapcsoló bekapcsolva még nagyon rövid ideig se. Ezért A PIC egyik utasításával kikapcsoljuk az aktuálisan bekapcsolt kapcsolót és csak a következő utasítással kapcsoljuk be a másikat. Így egy nagyon rövid időre egyik kapcsoló sincs bekapcsolva, tehát véletlenül sem következik be „összenyitás” miatti rövidzár a tápáramkör két sarka között.



65. ábra

Nyomtatott áramkör és kapcsolási rajz tervezése

A bemutatott mintapéldákat univerzális panelon (vero board) mutattam be. Ennek az előnye az olcsósága. A vezetékezést forrasztással végezzük, ami biztonságos érintkezést eredményez. Gyors és kényelmes a „dugdosós” próbapanel (bread board) használata is. Hátránya, hogy ha sok a vezeték, akkor gyakran előfordul, hogy valamelyik rosszul érintkezik, ami nagyon bosszantó és kezdő számára nehezen azonosítható hiba szokott lenni.

Ha áramkörökkel dolgozunk, célszerű a kapcsolási rajzokat valamilyen áramkör szerkesztő programmal elkészíteni. Lehetőleg olyat válasszunk, amely a kapcsolási rajzból a nyomtatott áramköri (NYÁK, panel) rajzolatot és a NYÁK gyártásához szükséges egyéb fájlokat is el tudja készíteni. Szempontként vegyük figyelembe, hogy lehetőleg legyen ingyenes verziója is. Ezeknek a kívánalmaknak jól megfelel az EAGLE tervező program ezért ennek a leírását mutatom be röviden.

EAGLE (Easily Applicable Graphical Layout Editor)[27]

Version 5.9.0 for Windows

Light Edition

Az EAGLE Layout Editor egy, a nyomtatott áramkörökhöz (PCB) egyszerűen használható szoftver. A név egy mozaikszó:

EAGLE (Easily Applicable Graphical Layout Editor)

(Könnyen Alkalmazható Grafikus Alaprajz Szerkesztő)

Az EAGLE használható Windows, Linux és Mac platformokon.

A program 3 fő modult tartalmaz, próbálom lefordítani őket, de eredeti elnevezésekkel megszokottabb, a magyar inkább magyarázat miatt szerepel itt.

- Layout Editor (Elrendezés szerkesztő)
- Schematic Editor (Vázlat szerkesztő)
- Autorouter (Automatikus láb összekötő, útvonalszerkesztő)

Egyetlen felhasználói felület van, nincs szükség Layout és Schematic közötti konvertálásra.

A Layout Editor tulajdonságai

- legnagyobb rajzterület: 1.6 x 1.6 m (64 x 64 hüvelyk)
- felbontás: 1 / 10 000mm (0.1 mikron)
- maximum 16 jelréteg
- hagyományos és SMT alkatrészek
- egy teljesen integrált szerkesztővel saját alkatrészeket is könnyen létrehozhatunk
- undo / redo (visszavonás/ újra) funkció bármely mélységben
- script fájlok a batch parancsok futtatására
- réz kitöltés
- kivágás és beillesztés funkció
- tervezési szabályok ellenőrzése
- interaktív Follow- me útvonaltervező

A Schematic Editor tulajdonságai

- akár 999 lap egy vázlatban
- elektronikai szabályok ellenőrzése
- kapu/láb csere (swap)
- NYÁK létrehozása a kapcsolási rajzból

Az Autorouter tulajdonságai

- útvonal bontás és újrarajzolás
- maximum16 jelréteg
- útvonal meghatározása a felhasználó által figyelembe vett költségekkel

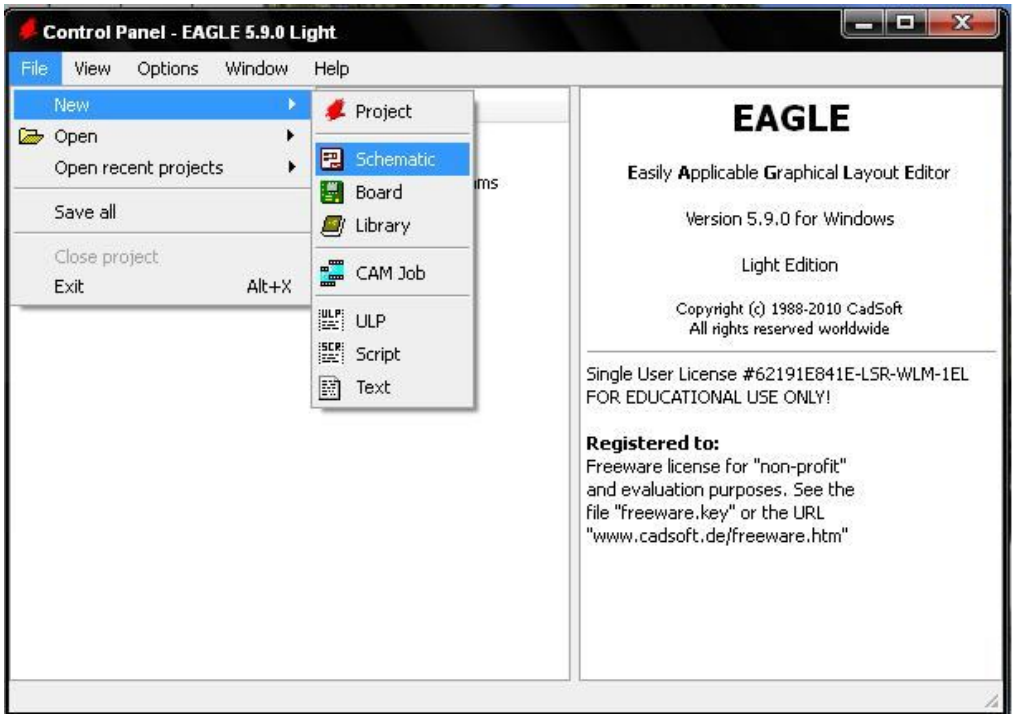
A program Lite verziója ingyenesen használható, de kizárólag non-profit célra a következő korlátozásokkal:

- a lap legnagyobb mérete 100 x 80 mm (4 x 3.2 inch)
- csak 2 vezető réteg használható
- a kapcsolási rajz csak 1 oldal terjedelmű lehet

A program bemutatása

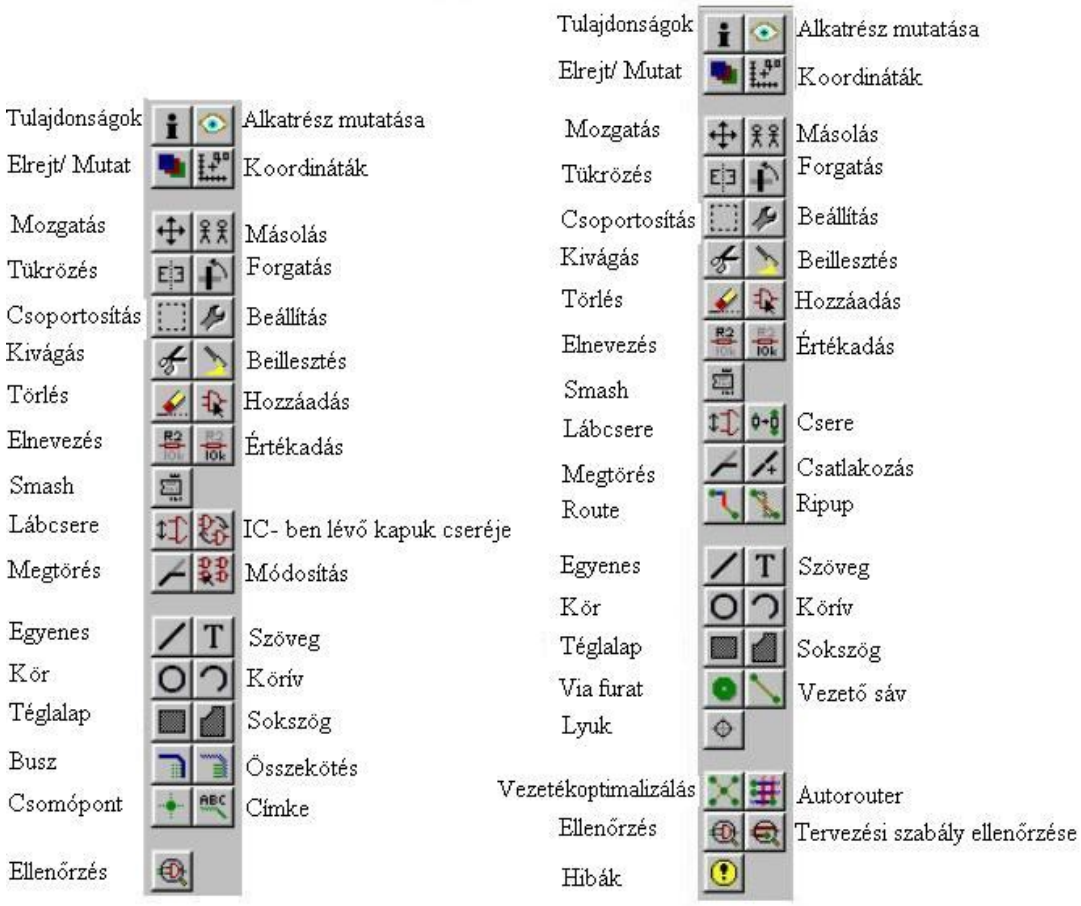
File/ New/ Schematic menüpontokra kattintva tudunk új kapcsolási rajzot kezdeni (66. ábra), a File/ New/ Board menüponttal pedig új panelt

File/ New/ Project menüponttal új projektet tudunk indítani, amivel létrejön egy mappa is (nekünk kell begépelnünk a nevét), ahová az új project összes adata kerül.



66. ábra

A következő ábrán az eszköztár látható (67. ábra):



A bal oldalon a kapcsolási rajz szerkesztő és a jobb oldalon a NYÁK-tervező ikon-eszköztára látható.

67. ábra

Amiket nem lehet röviden lefordítani, azokat itt külön hosszabban magyarázom.

SMASH

Megfigyelhetjük, hogy amennyiben egy alkatrészt elfordítunk, a megjelölése is együtt forog az alkatrésszel. A SMASH paranccsal ez kiküszöbölhető és az alkatrészek egyes jelölései

(ref. designator, name) az alkatrésztől függetlenül forgathatók, elmozdíthatók.

ROUTE

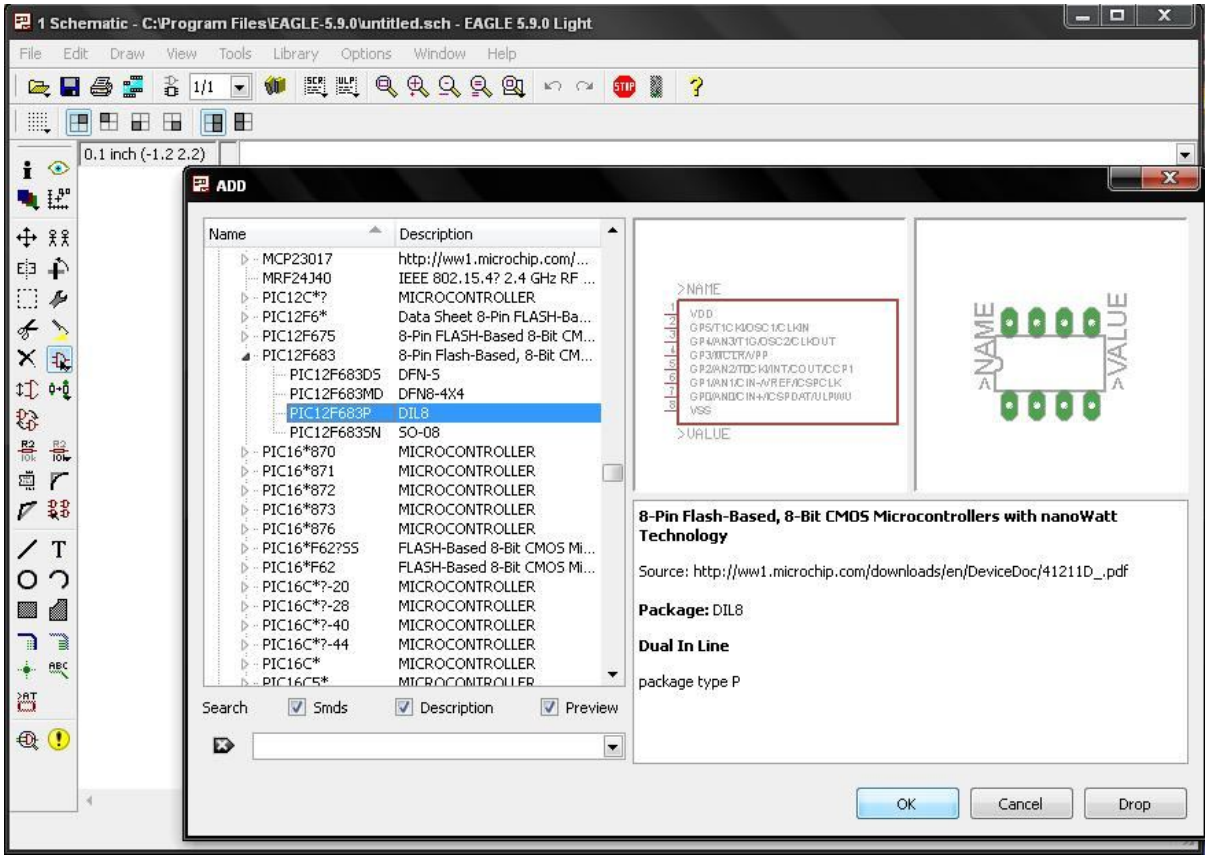
A paranccsal a légkötéseket (air wire) nyomtatott huzalozássá változtatjuk.

RIPUP

A paranccsal a már megrajzolt nyomtatott huzalt légkötésre állíthatjuk vissza.

ADD

Az Add ikonnal tudunk alkatrészt hozzáadni a rajzunkhoz. A könyvtárból válogathatunk alkatrészeket, amik a jobb oldalon láthatóak is Schematic és Board (Panel) nézetben is. A jobb oldalon alul információt láthatunk az alkatrészről, például hogy milyen a tokozása (68. ábra).



68. ábra

Miután kiválasztottuk a szükséges alkatrészt, ráklikkelünk a nekünk megfelelőre, majd a séma papírunkon is klikkelünk, ezzel rákerül a rajzra az alkatrész.

VIA

A másik rétegre való átmeneti lyuk.

A következő pár szóban a leggyakrabban használt ikonokat mutatom be.

WIRE (huzal)

Klikkeljünk a WIRE parancsra. A rajzolóréteg, dőlésszög és a vonal vastagsága a paramétersorban állítható be. A huzal kezdőpontját klikkeléssel adjuk meg. Húzzuk lassan a kurzort ferdén felfelé, elkezdődik a huzal rajzolása. A folytonos vonal befejezéséhez kétszer klikkeljünk.

CIRCLE (kör)

A kör rajzolásához a DRAW menüpont CIRCLE parancsát használjuk. Az EAGLE – ben a kör meghatározásához két klikkelésre van szükség – az első klikkeléssel a kör középpontját adjuk meg, a másodikkal pedig a kör sugarát. Helyezzük el a kurzort a kör középpontjának majdani helyén, klikkeljünk a bal egérgombbal, majd helyezzük át a kurzort a majdani körre és ismét klikkeljünk egyet a bal egérgombbal, megrajzolódik a kívánt átmérőjű és helyzetű körünk.

MOVE (mozgatás)

Az alkatrészek a rajzon szükség szerint áthelyezhetők a MOVE paranccsal. Az alkatrészre való klikkelés, a kurzor új helyre való áthelyezése majd a bal egérgombbal való újbóli klikkeléssel az alkatrész az új helyre helyeződik át. Az alkatrész az első klikkelésre színt változtat, jelezve, hogy ki lett választva és készen áll az áthelyezésre.

DELETE (törlés)

A DELETE (ikon, EDIT menüpont) paranccsal a rajzból objektumok törölhetők. Amennyiben összekötések vagy vonalak törlésére használjuk, csak a kiválasztott szegmensek törlődnek. Aktiváljuk a DELETE parancsot és klikkeljünk egy objektumra.

JUNCTION (kapcsolódás)

Két összekötés kapcsolódási pontján automatikusan megjelenik egy nagyobb pont, ami a kapcsolódást (junction) ábrázolja, de az automatikus pontmegjelenítés kikapcsolható. A kézi pontelhelyezést a JUNCTION paranccsal végezhetjük. A parancs aktiválása után a kurzornál megjelenik a pont, amelyet klikkeléssel az összekötésen bárhol (nemcsak a két összekötés kapcsolódási pontján) elhelyezhetünk. A kapcsolódási pont kizárólag az összekötésen (net) helyezhető el.

NYÁK lap generálása a kapcsolási rajzból

Ha elkészültünk a kapcsolási rajzzal, a BOARD parancs (ikon) segítségével automatikusan készülhet egy NYÁK. A program a NYÁK tervezéséhez szükséges információkat (alkatrészek és azok kapcsolásai, tehát a Partlist és Netlist) a kapcsolási rajzból automatikusan generálja. Elindítja a nyomtatott huzalozás szerkesztőt és az alkatrészeket, azok összekötéseivel a NYÁK lap körvonalán kívül helyezi el (ez egy fiktív áramkör). Az egyes alkatrészeket kézzel helyezzük a panelon abba a pozícióba, ahova kívánjuk. Az automatikus huzalozás készít egy elég jól használható áramköri tervet, amit kézzel módosíthatunk, ha úgy látjuk jónak. A kész rajzolatot kinyomtathatjuk és az amatőr technikában szokásos módszerekkel (pl. vasalás) elkészíthetjük a panelt. A program generálja mindazokat a fájlokat, amelyeket ki kell másolni és elvinni a panel gyártó műhelybe, ha szép és jó panelt akarunk gyártatni.

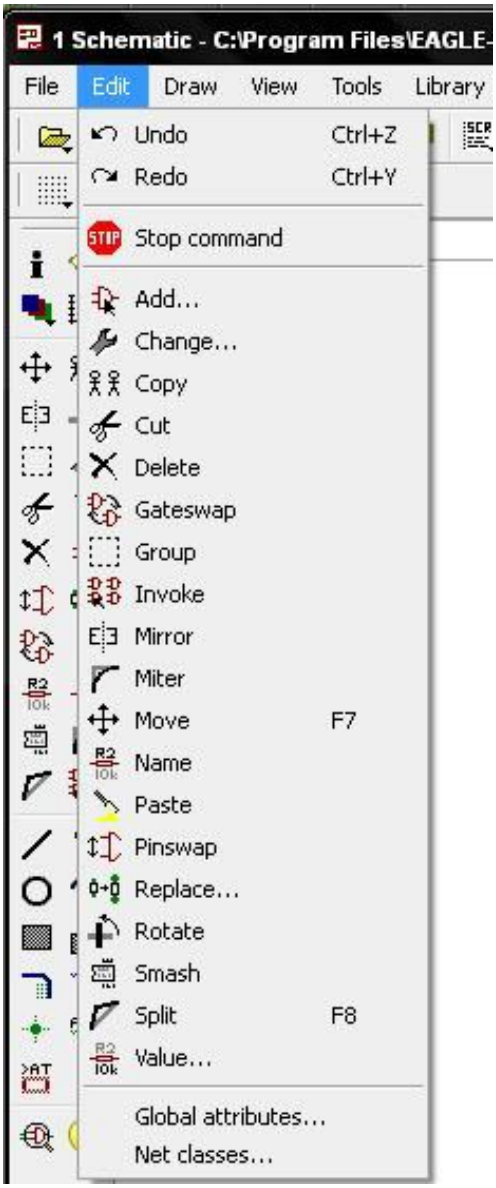
Új alkatrész rajzolása

Az Eagle program nagy előnye, hogy ha olyan alkatrészre van szükségünk, ami nincs benne gyárilag, akkor saját magunk is elkészíthetjük a rajzát. Nyissuk meg a kapcsolási rajz szerkesztőt. Menjünk a könyvtár/megnyitás menübe és nyissuk meg azt a könyvtárat, amelybe bele akarjuk helyezni a saját alkatrész, vagy meglévőt akarunk módosítani. Természetesen létrehozhatunk új könyvtárat is a saját alkatrészeinknek. A Symbol ikonra kattintva megnyílik az Edit szerkesztő ablak, amelyben fel vannak sorolva a megnyitott könyvtárban lévő alkatrész szimbólumok, mint rajzok. Kiválaszthatunk egy meglévőt, vagy létrehozhatunk egy újat. Az OK-ra kattintva megjelenik a szerkesztő ablakban a kiválasztott szimbólum, illetve üres ablakot kapunk, ha újat akarunk létrehozni. A szokásos szerkesztő eszközökkel szerkeszthetjük a saját

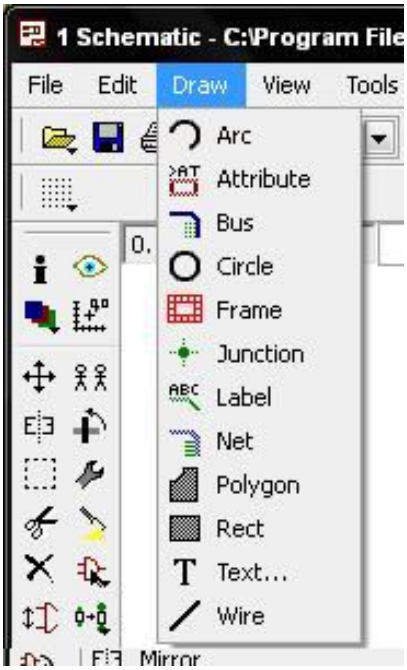
alkatrészünket. Előfordul, hogy olyan helyre szeretnénk behúzni vonalat, ahová nem engedi a túl nagyméretű raszter miatt. Ekkor a nézet/rács menüpontban vegyük kisebbre a raszter méretét és máris rajzolhatunk. Egyenes vonalat csak vízszintesen vagy függőlegesen enged rajzolni. A ferde vonal rajzolására egy lehetőség, hogy egy nagyon nagy sugarú körív rövid darabkáját vesszük, ami egyenesnek látszik. Ugyanígy rajzolhatunk az alkatrészünkhöz panel rajzolatot is (package). Ha kész a szimbólum és a panel rajzolat, összegezhetjük egy alkatrészben (device). Ha csak a kapcsolási rajz szimbólumra van szükségünk, a panel rajzolatot el is hagyhatjuk. Én is így rajzoltan meg ebben a dolgozatban pl. a szervomotort és a léptetőmotor tekercsét.

A rajzolás során leggyakrabban az alábbi menükkel találkozunk:

69. ábra


1 Schematic - C:\Program Files\EAGLE-
File
Edit
Draw
View
Tools
Library
Undo Ctrl+Z
Redo Ctrl+Y
Stop command
Add...
Change...
Copy
Cut
Delete
Gateswap
Group
Invoke
Mirror
Miter
Move F7
Name
Paste
Pinswap
Replace...
Rotate
Smash
Split F8
Value...
Global attributes...
Net classes...

70. ábra



71. ábra

Minden menühöz nem adnék fordítást (69. ábra, 70. ábra, 71. ábra), egyrészt mert általános számítástechnikai tevékenykedésünk kapcsán már ismerjük őket – gondolok itt olyanra, hogy File = Fájl, Save = Mentés stb. – másrészt meg egyik ábrán már szerepelnek ezeknek a fordításai, csak ezek nem az eszköztárra kihelyezett ikonok, hanem a menükben lévők.

72. ábra

Fent a menüsor található, lent az Eszköztárunk, ami más programokból már jól megszokott (72. ábra).

Balról jobbra haladva sorolom:

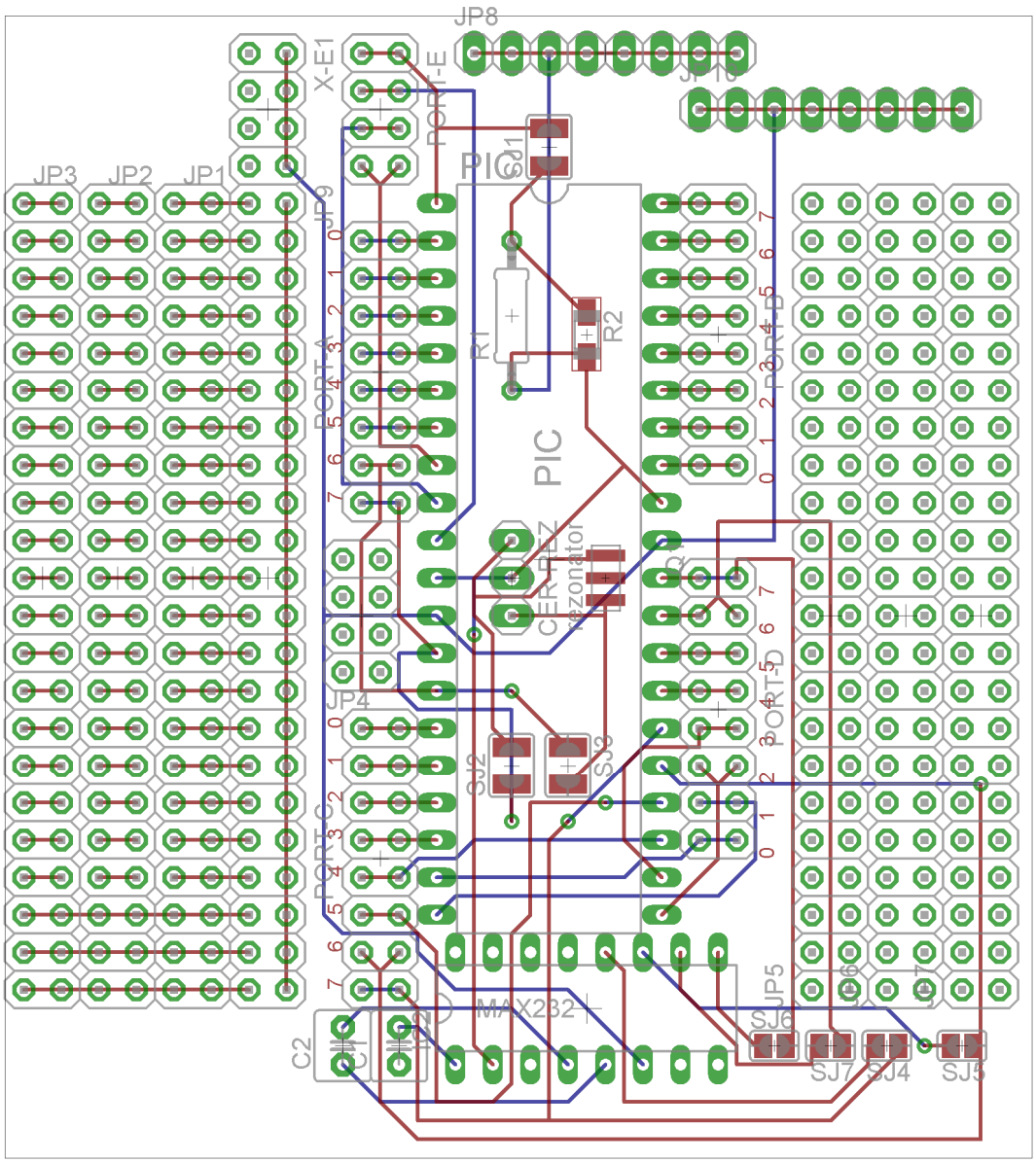
Megnyitás, Mentés, Nyomtatás, CAM, Schematic/ Board váltás, Oldal, Könyvtár, Script, Futatás, Nagyítás kitöltve, Nagyítás, Kicsinyítés, Újrarajzolás, Kiválasztás, Vissza, Előre, Mégse, Végrehajtandó parancs, Súgó

Egyszerűen megoldható mintafeladat a Microchip TC74 típusú hőmérő IC-vel való hőmérsékletmérés. Az IC az I²C buszon kommunikál a mikrovezérlővel. A mért hőmérsékletet digitális formában szolgáltatja. Az alábbi PICBASIC programrészlet beolvassa a 3 darab hőmérő IC által szolgáltatott hőmérsékleti értékeket és LCD kijelzőre kiírja. Az I²C busz és az LCD kijelző használatát nem részletezem, a PICBASIC kézikönyvében megtalálható.

```
DEFINE LCD_LINES 4          'LCD line numbers
DEFINE LCD_BITS  4          'LCD bus size (4 or 8)
DEFINE LCD_DREG PORTC       'LCD data port
DEFINE LCD_DBIT  4          'LCD data starting bit (0 or4)
DEFINE LCD_RSREG PORTB      'LCD register select port
DEFINE LCD_RSBIT 1          'LCD register select bit
DEFINE LCD_EREG PORTB       'LCD enable port
DEFINE LCD_EBIT  3          'LCD enable bit
TRISB=%00111101
LOW PORTB.2 'LCD R/W' LINE LOW (WR)
TRISA=255
CONTROL var byte
ADRESS var byte
hom1 var byte
hom2 var byte
hom3 var byte

 '3 db. TC74 tipusu homero IC olvasasa I2C buszon

        CONTROL=%10010000
        ADRESS=1
        I2CWRITE PORTC.2, PORTC.3, CONTROL, ADRESS, [0]
        ADRESS=0
        I2CREAD PORTC.2, PORTC.3, CONTROL, ADRESS, [hom1]

        CONTROL=%10010100
        ADRESS=1
        I2CWRITE PORTC.2, PORTC.3, CONTROL, ADRESS, [0]
        ADRESS=0
        I2CREAD PORTC.2, PORTC.3, CONTROL, ADRESS, [hom2]

        CONTROL=%10010110
        ADRESS=1
        I2CWRITE PORTC.2, PORTC.3, CONTROL, ADRESS, [0]
        ADRESS=0
        I2CREAD PORTC.2, PORTC.3, CONTROL, ADRESS, [hom3]

homerkiir:
        IF hom2<3 THEN
                LCDOUT $FE,1,"**Fagyveszely!**"
        ENDIF
        LCDOUT $FE, $C0,"Temp in:", #hom1, 223,"C"
        LCDOUT $FE, $90, "Temp out:", #hom2, 223,"C"
        LCDOUT $FE, $D0, "Water temp:", #hom3, 223,"C"
```

Egy kész nyomtatott áramkör

Végül bemutatok egy általam elkészített minta nyomtatott áramkört, ami a PIC elhelyezésére szolgáló furatokon kívül tartalmazza a legszükségesebbeket. Tartalmazza a kristály vagy rezonátor bekötési lehetőségét mind hagyományos, mind SMD alkatrészek esetén. Tartalmaz egy MAX232 szintillesztő áramkört a soros kommunikációhoz és sok furatot a további huzalozáshoz. A kapcsolási rajzot az (1. melléklet), az áramköri rajzolatot a (73. ábra) láthatjuk.



73. ábra

Összefoglalás

Másodéves hallgató koromban felvettem a Budapesti Műszaki és Gazdaságtudományi Egyetem (BME) Vegyészmérnöki és Biomérnöki Karon (VBK) vendéghallgatóként a számítástechnika III. című tárgy mikrovezérlőkkel foglalkozó kurzusát. Megtetszett a mikrovezérlők alkalmazása. A következő félévekben segítettem a tárgyat tartó tanárnak – későbbi témavezetőmnek – az órák tartásában. Láttam, hogy a hallgatók könnyen megtanulják az alapvető alkalmazásokat, ha egyszerűen magyarázzuk el nekik. Ott próbáltam ki többféle módszert a mikrovezérlőknek a kezdő szinten történő oktatására. Az órán bevált módszereket, példákat gyűjtöttem össze diplomamunkámban. A hallgatók érzékenyen reagáltak arra, hogy ne régi, elavult eszközökkel oktassunk.

Diplomamunkámban a PIC mikrovezérlők terén a jelenlegi legmodernebb eszközöket mutatom be olyan szinten, hogy az interneten fellelhető kézikönyvek, leírások remélhetően érthetővé váljanak kezdő számára is. A Microchip PIC10-es szériájától kezdve a PIC-24-es szériáig terjedő mikrovezérlőket és azok használatát írom le, mindig fő szempontnak tartva az olcsóságot és az amatőr körülmények közötti megvalósíthatóságot.

A diplomamunka terjedelmi korlátai miatt nem törekedhettem teljes részletességre, inkább a kezdő lépéseken való átsegítésre. Bemutatom a kezdő számára ingyenesen elérhető szoftver eszközöket, mint az MPLAB fejlesztői környezet, valamint a PICBASIC és a C nyelv ingyenesen is használható fordítói. Megadtam néhány egyszerűen megvalósítható konkrét áramköri megoldást, ami az oktatásban vagy az önálló tanulásban első lépésként megvalósítható.

Röviden bemutattam olyan módszereket is, ami a hallgatókat általában érdekelte, de nehezebben értették meg. Ezek közé tartozik a léptetőmotor és a szervomotor programozása. Általában az irodalomtól eltérő, de az órán bevált módszert alkalmaztam, miszerint a feladatokat általánosan kapcsoló áramkörökkel mutatom be és csak utána a konkrét megvalósítást.

Kapcsoló eszközként FET-et használtam, mert a bipoláris tranzisztornál kényelmesebben használhatók nagyobb „drain” áramuk és rendkívül kicsi „gate”

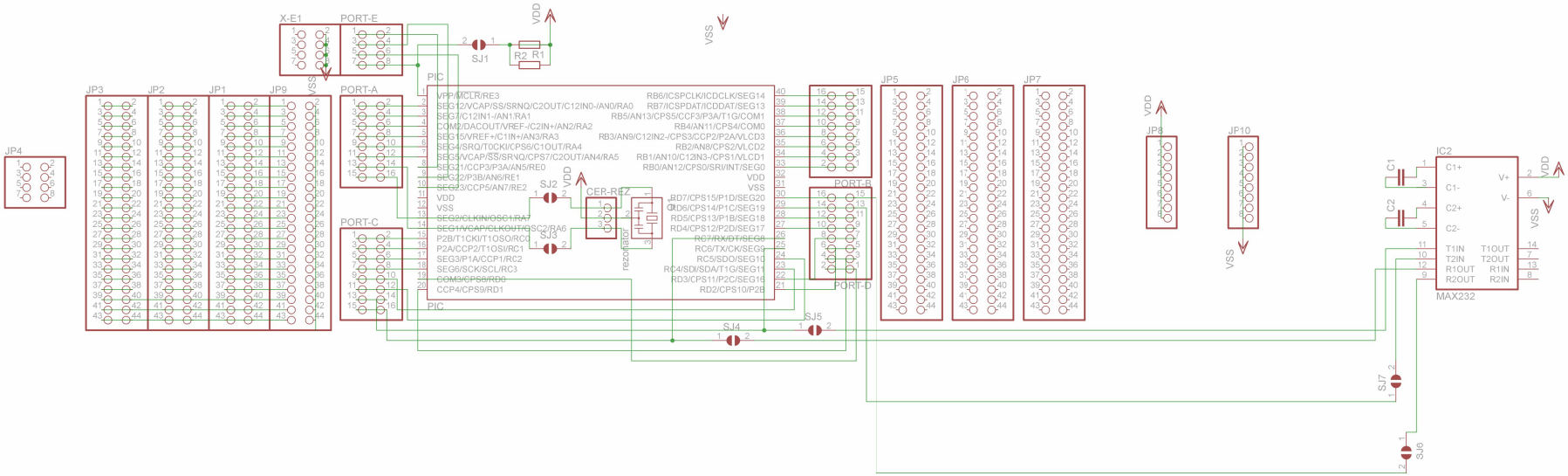
áramuk miatt. A fellelhető irodalmak születésének idején még a bipoláris tranzisztorok alkalmazása volt a logikus, de ma már – véleményem szerint – a FET.

Befejezésként egy elkészíthető vagy legyártatható kísérleti panelt terveztem és bemutatom a panel tervezés lépéseit.

Remélem, hogy diplomamunkámmal segíteni tudom a mikrovezérlőkkel most ismerkedőket.

PIC	lábszám	program memória (Kszó)	program memória (Kbyte)	adat memória (byte)	EEPROM byte	számláló 8 bites	számláló 16 bites	számláló 32 bites	A/D konverter	analog komparátor/CCP/ECCP	USART	I2c/SPI	működési freky. max. Mhz/MIPS	belső órajel generátor	RTC	ár (Ft)
12F683	8	2		128	256	2	1		+	+			20 Mhz	+		300
PIC 16F54	18	0.5		25		1	0						20 Mhz			150
PIC 16F57	28	2		72		1	0						20 Mhz			200
PIC 16F59	40	2		134		1	0						20 Mhz			250
PIC 16F84	18	1		68	64	1	0						20 Mhz			900
PIC 16F628	18	2		224	128	2	1			+	+		20 Mhz	+		400
PIC 16F648	18	4		256	256	2	1			+	+		20 Mhz	+		500
PIC16F877	40	8		368	256	2	1		+	+	+	+	20 Mhz			1500!
PIC 16F1938	28	16		1024	256	4	1		+	+	+	+	32 Mhz	+		500
PIC 16F1939	40	16		1024	256	4	1		+	+	+	+	32 Mhz	+		600
PIC18F2321	28	8		512	256	1	3		+	+	+	+	40 Mhz	+		800
PIC18F4321	40	8		512	256	1	3		+	+	+	+	40 Mhz	+		900
PIC18F26K22	28		64	3896	1024	3	4		+	+	+	+	16 MIPS	+		700
PIC18F46K22	40		64	3896	1024	3	4		+	+	+	+	16 MIPS	+		900
PIC24F04KA200	14		4	512			3		+	+	+	+	16 MIPS	+		400
PIC24F08KA102	28		8	1536	512		3		+	+	+	+	16 MIPS	+	+	600
PIC24FJ64GA002	28		64	8K			5		+	+	+	+	16 MIPS	+		900
PIC24FV32KA302	28		32	2K	512		5		+	+	+	+	16 MIPS	+	+	800
PIC24HJ12GP202	28		12	1K			3	1	+	+	+	+	40 MIPS	+		800
PIC24HJ32GP302	28		32	4K			5	2	+	+	+	+	40 MIPS	+	+	1000
PIC24HJ128GP502	28		128	8K			5	2	+	+	+	+	40 MIPS	+	+	1200

1. táblázat (PIC-ek összehasonlító táblázata)



1. melléklet (kapcsolási rajz)

Irodalomjegyzék

[1] http://www.aut.uni-obuda.hu/konya/mikro/pic_www/index.htm

[2] Kónya László-Kopják József: PIC mikrovezérlők alkalmazástechnikája, Chipcad, Budapest, (2009)

[3] http://plc.mechatronika.hu/mikrovez/mikrovezerlok.htm

[4] http://www.microchip.com

[5] http://www.chipcad.hu

[6] Milan Verle: PIC Microcontrollers, mikroElektronika, Beograd, (2006)

[7] http://itl7.elte.hu/~vella

[8] home.mit.bme.hu/~mersich/microchip/pic.ppt

[9] Myke Predko: PICMicro Microcontroller Pocket Reference, McGraw-Hill (2001)

[10] Di Jasio:PIC Microcontrollers - know it all, Newnes,Oxford (2008)

[11] John Morton: The PIC Microcontroller, Elsevier (2005)

[12] Sid Katzen: The Quintessential PIC Microcontroller, Springer (2000)

[13] John Iovine: PIC Microcontroller Project Book, McGraw-Hill (2004)

[14] Dogan Ibrahim: Picbasic Projects, Newnes,Oxford (2006)

[15] Dogan Ibrahim : Advanced PIC Microcontroller - Projects in C From USB to RTOS with the PIC18F Series, Newnes (2008)

[16] Julio Sanchez: Microcontroller Programming - The Microchip PIC, CRC Press (2007)

[17] Nebojsa Matic, PIC Microcontrollers for beginners, too! , Mikroelektronika, Beograd (2000)

[18] http://www.inetintel.com/embeddeddocs/EBooks/How-to%20Build%20a%20Little%20Autonomous%20Robot%20with%20PIC%2018F4520.pdf

[19] John Iovine: PIC Robotics, Newnes (2004)

[20] Doug Williams: PDA Robotics, McGraw-Hill (2003)

[21] http://kapcsolasok.hu/digitalis/86-pic-programozas-alapok

[22] Advanced PIC Microcontroller - Projects in C From USB to RTOS with the PIC18F Series; Dogan Ibrahim (Newnes, 2008).pdf

[23] http://www.ccsinfo.com

[24] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en534868&page=wwwCompilers#P29_2372

[25] http://melabs.com/pbpdemo.htm

[26] BME VBK számítástechnika III. című tárgy mikrovezérlő kurzusa, Neptun kód: BMEVEKIU305

[27] http://www.cadsoftusa.com