SZAKDOLGOZAT

Zámbó Lajos

Debrecen

2010

Debreceni Egyetem
Informatikai Kar

AZ INTELLIGENS OTTHON

Belső témavezető:
Dr. Kuki Attila
egyetemi adjunktus

Külső témavezető:
Szeleczki Szabolcs
villamosmérnök

Készítette:
Zámbó Lajos
mérnök informatikus

Debrecen

2010

Tartalomjegyzék

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani tanáraimnak, különösen Dr. Kuki Attila Tanár Úrnak, segítőkészségéért.

Szeleczki Szabolcs barátomnak, aki munkája mellet szakított időt a külső témavezetői munka elvégzésére. Szakmai irányítása nélkül nem készülhetett volna el ez a dolgozat.

És természetesen a családomnak, akik biztosították nekem a nyugodt hátteret.

Bevezetés

Bizonyára mindenki látott már sci-fi filmekben, sorozatokban (mint például: Star Trek, 2001. Űrodisszeia, Eureka) olyan űrhajókat, űrállomásokat, házakat, melyeket számítógépek vezéreltek. Az informatika és az elektronika gyors fejlődésének köszönhetően az alacsony költségű, kis teljesítményfelvételű mikrovezérlők elterjedése az átlagember számára is elérhetővé teszi a különféle épület-automatizálási rendszerek nyújtotta szolgáltatásokat, melyek nemcsak kényelmi, de elsősorban energia-takarékossági szempontból is egyre fontosabbá válnak.

Aki most építkezik, vagy ezt tervezi, annak érdemes már építkezéskor biztosítania egy ilyen rendszer későbbi kiépítésének lehetőségét. Ez elsősorban a villanyszerelési munkák anyag- és munkadíjának körülbelül 20-25%-os emelkedésével jár, mely később többszörösen megtérül, mikor nem kell utólagosan átalakítani a ház elektromos vezetékezését (vésés, kábelezés, festés) vagy ezt elkerülendő költségesebb rádiós vagy az erősáramú vezetéken működő kommunikációs hálózat kiépítését.

Habár ezek a rendszerek ma még szükségtelen luxusnak tűnnek, 10 – 20 év múlva valószínűleg éppen olyan elterjedtek lesznek, mint például ma a személyi számítógép.

Ebben a dolgozatban egy kevés anyagi ráfordítással megépíthető saját fejlesztésű rendszer alapjait szeretném bemutatni.

Az intelligens otthon

Mi is az az intelligens otthon?

Az épület villamos, gépészeti, informatikai és egyéb rendszereinek számítógépes felügyelete és vezérlése. Ez részben a felhasználó által előre definiált napi rutin feladatok (például: redőnyfelhúzás minden reggel), elvégzéséből, részben pedig eseményvezérelt tevékenységek (például: lámpa felkapcsolása felhasználói beavatkozás hatására) végrehajtásából áll.

A komplex, intelligens megoldások a következő alrendszerek működtetését fogják össze:

- biztonságtechnika
- videó megfigyelő rendszer
- szórakoztató elektronika
- világítás
- fűtés, hűtés
- számítástechnika
- telefon, kaputelefon
- redőny, függöny, reluxa, kapu
- öntöző
- szauna, jacuzzi és egyéb kényelmi berendezések

A beavatkozás legegyszerűbb módját a fali kapcsolókkal történő kezelés jelenti. Előnye, hogy a rendszer azok számára is kezelhető marad, akik hagyományos, megszokott módon szeretnék használni a házat. Ennél kényelmesebb mód egy kézi távirányító használata. (Ezek száma egy házon belül nincs korlátozva, így akár minden családtagnak lehet sajátja.) Az alkalmazható távirányítók skálája a kijelző nélküli egyszerűbbtől, a háttérvilágítással és érintőképernyővel is rendelkezőn át egészen a nagyfelbontású színes kijelzős érintőképernyősig terjed.

Az említett távirányítók közös jellemzője, hogy képesek vezérelni híradástechnikai készüléket, emellett parancsokat adhatunk a világítás beállítására és egyéb feladatokra. A vezeték nélküli számítógépes hálózatok elterjedésével az ún. PDA-k, vagyis hordozható

tenyérszámítógépek használata is lehetségessé vált. A vezérlés másik módját a különféle falra szerelhető érintőképernyők jelentik. Ezek léteznek fekete-fehér és színes kivitelben is, egészen SVGA felbontásig. Kényelmes megoldás a házunk alaprajzán feltüntetett eszközöket ily módon vezérelni. [1]

Már kereskedelmi forgalomban is kapható MAIA, a "Virtuális asszisztens", aki felismeri a hangutasításokat, akár beillesztett szövegkörnyezetben is, mert folyamatosan tanulja a szóhasználatokat. Alkalmas továbbá a multimédiás eszközök hangvezérlésére, e-mail írására (diktálva), internetes böngészésre hangutasítással (amennyiben szükséges, akkor ezt fel is olvassa), illetve a számítógépen futó alkalmazások hangvezérléssel történő működtetésre. Akár online póker játszására is. A demo video a következő címen tekinthető meg: http://www.domintell.hu/en/news.html [2]

Az eddig ismertetett kezelési megoldások közös jellemzője, hogy lokális jellegűek, a vezérlés ezekkel az eszközökkel csak a házon belül, esetleg annak közvetlen környezetéből történhet. Sokszor előnyös lehet otthonunk távoli irányítása, felügyelete. Népszerű lehetőség egy sms-en keresztül irányítani valamit, illetve állapotot lekérdezni. Ilyen módon hazaérkezésünk előtt bekapcsolhatjuk a klímát, fűtést.

Az Internet gyakorlatilag már mindenhol elérhető ezért igény, hogy a ház is vezérelhető legyen ezen keresztül. Hiszen ha egy üzletember külföldre utazik, szeretné tudni, hogy a lakásban minden rendben van-e, esetleg belső kamerák segítségével szeretné ezt vizuálisan is ellenőrizni, vagy éppen a fűtésbe beavatkozni. Ezt természetesen az Internet segítségével bárhonnan megteheti. [3]

Leggyakrabban a fűtési rendszer vezérlését oldják meg ilyen rendszerrel. Ezzel, sokkal komfortosabbá és gazdaságosabbá tehető a fűtés. A különböző helyiségekben különböző hőmérsékletet állíthatnak be. Ezzel a családtagok eltérő igényei sokkal inkább kielégíthetők. Lehetőség van időprogram, időjárás és igénybevétel szerinti szabályzásra. A rendszer hazaérkezésünkkor automatikusan komfort hőmérsékletre kapcsol.

A redőnyök, árnyékolók illetve a világítás vezérlésével a szobákban létrehozható a természetes és mesterséges megvilágítás ideális kombinációja, mely életritmusunkhoz és a külső fényviszonyokhoz igazodik. A redőnyök egyenként és csoportosan is vezérelhetők. Például

reggel évszaktól függően adott időben automatikusan felhúzza a redőnyöket, este leengedi. A napsütés és a belső hőmérsékleti igény figyelembevételével, míg nyáron a közvetlen napsütéstől óvja a szobát, addig az őszi és tavaszi időszakban ennek ellenkezőjével a fűtési költséget csökkenti.

A hangulatos és praktikus világítás egyre inkább terjed. Már nem csak egy nagy csillár van a nappaliban, hanem számtalan különféle világítótest. Melyek egyrészt az alap másrészt a hangulatvilágítást biztosítják. Ha a nappaliban akár 10 vagy még több áramkört is kialakítunk ezeknek a fényerő beállítása hagyományos módon már komoly feladat. De ha a rendszerre bízzuk, akkor akár egy gombnyomással mindig az éppen megkívánt beállítást tudjuk előhívni. A televízió nézéshez valami hangulatos derítőfényt, a vendégváráshoz egy a részleteket kiemelő világítást, míg egy kártyapartihoz az asztal kiemelését állíthatjuk be. Ezeket a beállításokat elmentve ezek a későbbiekben bármikor egy gombnyomással előhívhatók. [3]

Ha a kamerával megfigyelt kertben mozgás észlelhető, a tévé képernyőjén a kamera által látott kép jelenik meg, távollétünkben pedig ilyenkor a videó, vagy a napjainkra jellemzőbb digitális rögzítő automatikusan tárolja a képet, illetve sms, vagy az újabb generációs telefonok esetében képes üzenetet kapunk. A biztonságot szolgálja, hogy távozásunkkor a rendszer automatikusan áramtalanítja otthonunkat (természetesen csak a kívánt áramköröket, dugaljakat). Ezentúl tehát nem kell a bekapcsolva felejtett vasaló miatt aggódnunk. Távozáskor fentiek mellett élesedik a riasztó, a kamerák figyelő üzemmódba állnak, majd napnyugta után elindul a jelenlét szimuláció a világítások részben szokásunkhoz igazított, részben véletlenszerű kapcsolgatásával, de akár a tévé hangerejét is képes változtatni, ugyanúgy, mintha mi tartózkodnánk otthon. [2]

Az időjárás függvényében vezérelheti a kerti öntözőrendszert. Van, aki azt szeretné megoldani, hogy a medence hőmérséklet vagy éppen vegyszerszint állításához ne keljen a gépházba lemenni, hanem mindez egy könnyen kezelhető, akár a medence közelében elhelyezhető kijelzőről megoldható legyen. Mire hazaér a tulajdonos a szauna megfelelő hőmérsékletű legyen. Feleségek vagy gyerekek szeretik, ha az ágy mellett van egy úgynevezett pánik kapcsoló, mellyel, ha valami neszt észlelnek, a ház körül felkapcsolhatják azokat a világításokat melyek megvilágítják a házat, de az adott helyiséget nem. Ha a riasztó rendszer megszólal a házkörüli világítás, illetve ha a tulajdonosok nem tartózkodnak otthon a belső fények

is felkapcsolnak, akár villognak. Ezzel is felhívva a környék lakóinak figyelmét a betörésre. Mindeközben a tulajdonost SMS-ben értesíti az eseményről.

A házimozi rendszerek terjedésével igény jelentkezik ezek az automatizálásba bevonására is. Ha bekapcsoljuk a DVD-ét, akkor automatikusan a vetítővászon is menjen le, a világítás álljon be a megfelelő szintre, esetleg a redőnyöket engedje le és a fűtést állítsa egy kicsit feljebb a rendszer. Mindezt automatikusan, anélkül, hogy a felhasználónak bármihez is hozzá kellene nyúlnia. [3]

Előnyök

A fent felsorolt néhány alkalmazási példa is bizonyítja, hogy a mai otthonokban amúgy is megtalálható különálló rendszerek integrálásával milyen sokoldalú, komplex feladatok ellátására alkalmas rendszert kaphatunk.

A napi rutinfeladatok megkönnyítése, és a különféle egyedi rendszerek összefogott kezelése mellett az intelligens otthon fő célja, biztonságunk növelése, és az energia megtakarítás mellyel a környezeti terhelés is csökken. [1]

Például:

- A riasztó élesítésekor a távozásunk után szükségtelenné vált fogyasztók áramtalanítása.
- A hagyományos termosztát vezérlésű fűtési rendszerekkel szemben kisebb a hőingadozás, mely nem csak a komfortérzetet növeli, hanem a fűtési költséget is csökkenti.

Anyagi szempontból előnyös, hogy a rendszer modulárisan is kiépíthető, így az egyes alrendszerek eltérő időben, tetszés szerinti kiépítettségig valósíthatók meg.

Tanácsos a fűtési, és világítási/ áramellátási alrendszer kiépítésével kezdeni, mert az ezekkel az alrendszerekkel elérhető megtakarítás rövid idő alatt megtéríti a kiépítésre fordított költséget.

A PIC

Bevezető

Az egyes épületgépészeti alrendszerek érzékelő- és beavatkozó egységeinek vezérlése programozható logikai vezérlők (Programmable Logical Controller, PLC) alkalmazásával is megoldható. Ezek olyan ipari célú felhasználásra tervezett mikroszámítógépek, melyek ipari környezetben (por, légszennyezettség, rázkódás) képesek szabályozások és vezérlések megvalósítására. Bemeneti és kimeneti egységeinek feszültségszintje is az iparban használatos szintekhez illeszkedik (pl: a digitális be- és kimenetek feszültségszintje 24, 60, vagy akár 220 V is lehet.). ezen paramétereket a PLC-k ára is tükrözi. Otthoni felhasználásra célszerűbb az olcsó mikrovezérlők alkalmazása.

Egy-egy egység feladatának ellátásához viszonylag kicsi, egyszerű program szükséges. Például a hőmérő modul két fő funkciója a mérés (kommunikáció a szenzorral) és az adattovábbítás a vezérlőegység felé. Ezeket a feladatokat egy mikrovezérlő megbízhatóan és kis teljesítményfelvétel mellet képes ellátni. Ha az egyes szenzorokat és beavatkozó egységeket saját intelligenciával ruházzuk fel, akkor a vezérlőegységre is kevesebb feladat hárul.

A PIC mikrovezérlők gyökere a Harvard egyetemre (Harvard-architektúra) nyúlik vissza, a Védelmi Minisztérium projektjének keretében készült. A Harvard-architektúrát először a Signetics 8x300-ban használták, majd a General Instruments adaptálta és a periféria illesztő vezérlőiben (peripheral interface controller, azaz PIC) alkalmazta. A gyártás később (1985) az arizoniai Microchip Technology-hoz került, s a PIC lett a cég fő gyártmánya. A PIC mikrovezérlők típusválasztéka mára igen széles lett: PIC10x, PIC12x, PIC14x, PIC16x, PIC18x, PIC24x. Ezekben a sorozatokban szinte minden alkalmazásra találhatunk megfelelő mikrovezérlőt. A Microchip nagyon jó kézikönyveket és alkalmazási leírásokat bocsát a felhasználók számára (ezek természetesen angol nyelvűek). A fejlesztéseket mindig úgy végzi, hogy előtte kikéri a felhasználók véleményeit, s ennek alapján készül el a következő generációs PIC. Ezen mikrovezérlők ára nagyon kedvező, 500-2500Ft között mozog, így otthoni felhasználásra is megvásárolhatók nagyobb anyagi ráfordítás nélkül. A Microchip a programfejlesztéshez szükséges MPLAB programot ingyen biztosítja a felhasználóknak. [4]

A PIC-et kis teljesítményfelvétele, széleskörű alkalmazhatósága és alacsony ára teszi sikeres termékké, melyet előszeretettel alkalmaznak különféle berendezések beágyazott rendszereiben.

Beágyazott rendszerek

A beágyazott rendszer jellemzően a kis mikrovezérlők lehetőségeinek kiaknázására szolgál, ilyen például a Microchip PICmicro MCU-ja vagy a dsPIC Digital Signal Controller-e (DSCs). Ezek a mikrovezérlők egyazon chip-en elhelyezett mikroprocesszorból, perifériákból és egyéb kiegészítő áramkörökből állnak. Ezen áramkörök összessége olyan kisméretű vezérlőmodult alkot, melynek működéséhez csak néhány külső alkatrész szükséges. A mikrovezérlők egyéb elektronikus és mechanikus eszközökbe ágyazhatók, így azok digitális vezérlése olcsón megoldható.

A mikrovezérlő és a PC közötti különbségek

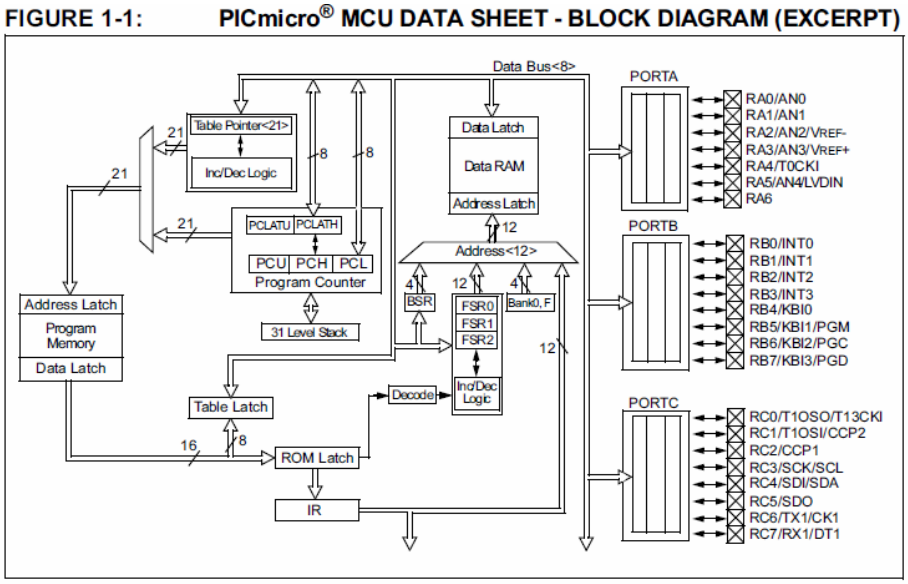
A legfőbb különbség a mikrovezérlő és a PC között az hogy, a mikrovezérlő egy specifikus feladat vagy feladat csoport elvégzésére hívatott. A PC-t pedig sokféle különböző típusú program futtatására és sokféle külső eszközhöz való csatlakoztatásra tervezték. Egy mikrovezérlőnek egy programja van. Ennek eredményeképpen olcsón előállítható, hiszen éppen annyi számítási kapacitása és hardver eleme van, ami ahhoz elegendő, hogy az adott feladatot ellássa. A PC viszonylag drága általános célú CPU-val és sok egyéb külső eszközzel (memória, lemezmeghajtók, video vezérlők, hálózati interfész, stb.) rendelkezik. A beágyazott rendszer egy kisköltségű mikrovezérlőt (MCU) tartalmaz, annak intelligenciájával és sok perifériájával egyazon chip-en és viszonylag kevés külső eszköz felhasználásával. A beágyazott rendszer gyakran láthatatlan része, vagy alegysége egy másik készüléknek, mint például egy vezeték nélküli fúrógép, hűtőszekrény vagy garázsnyitó. A mikrovezérlő ezekben az termékekben az egész eszköz funkcióinak csak egy apró részét végzi. Ez kisköltségű intelligenciát biztosít ezen eszközök kritikus alrendszerei számára. A füstjelző egy ilyen példa a beágyazott rendszerekre. Ennek feladata az érzékelő jeleinek kiértékelése és a riasztó megszólaltatása, ha a szenzorból érkező jelek füst jelenlétét jelzik. A füstjelzőben egy kis program fut végtelen ciklusban.

Mintavételezi a szenzor jeleit, vagy alacsony teljesítményfelvételű “alvás” módban várja, hogy a szenzortól érkező jel felébressze A program ekkor megszólaltatja a riasztót. A programnak lehet még néhány egyéb kiegészítő funkciója, úgymint a felhasználói teszt és a telep lemerülésének jelzése. Habár egy szenzorral és hangkimenettel rendelkező PC beprogramozható erre a feladatra, nem volna költséghatékony (és nem működne egy 9V-os elemről évekig felügyelet nélkül). A beágyazott rendszerek olcsó mikrovezérlők alkalmazásával intelligenciával ruházzák fel környezetünk mindennapi eszközeit, ilyenek a füstjelzők, fényképezőgépek, mobil telefonok, háztartási gépek, autók, chip kártyák és biztonsági rendszerek.

A mikrovezérlő komponensei

A PICmicro MCU tartalmaz egy program memóriát a firmware, vagy kódolt utasítások számára, melyek a program futtatásához szükségesek. Ezenkívül „file register”-eket, olyan memóriát is tartalmaz mely a program számítási műveleteihez szükséges változók tárolására vagy ideiglenes tárhelyként szolgál. Számos periféria áramköreit is tartalmazza egyazon chip-en. Néhány ilyen eszközt I/O portnak neveznek. Ezek olyan kivezetései a mikrovezérlőnek melyek magas, vagy alacsony logikai szintre állíthatók. Az I/O portokon keresztül lehet jeleket küldeni, lámpákat villogtatni, hangszórókat meghajtani. Ezek a kivezetések gyakran kétirányú adatforgalmat tesznek lehetővé és bemenetként is konfigurálhatók lehetővé téve ezzel, hogy a program egy külső kapcsoló, érzékelő jeleire reagáljon, vagy más külső eszközzel kommunikáljon. Az 1. ábrán egy MCU blokkvázlatának egy részlete látható. Egy ilyen rendszer tervezésekor el kell dönteni, hogy melyik perifériák szükségesek az adott alkalmazáshoz. Az A/D konverterek lehetővé teszik a mikrovezérlő számára, az analóg kimenetű szenzorok jeleinek fogadását. A soros portokon keresztül a mikrovezérlő egy másik mikrovezérlővel vagy PC-vel tud kommunikálni helyi hálózaton, vagy akár az Interneten keresztül. Az időzítők (timers) pontosan mérik a jelváltozások közötti időtartamokat, kommunikációs jeleket generálnak és fogadnak, precíz hullámformákat állítanak elő és még a mikrovezérlőt is újraindítják, ha „lefagy” vagy egy műszaki hiba miatt „eltéved”. Más perifériák detektálják, ha tápfeszültség veszélyesen alacsony szintre csökken, így a mikrovezérlő elmentheti a kritikus információkat és biztonságosan lekapcsolhat mielőtt a tápfeszültség teljesen megszűnik. Egy adott alkalmazás

futtatásához szükséges perifériák típusa és száma, valamint a memória méret nagymértékben meghatározza, hogy melyik PICmicro MCU kerüljön felhasználásra. [5]



1. ábra: MCU blokkvázlat (részlet) [5]

Architektúra

A PIC mikrovezérlők viszonylag nagy teljesítményüket a következő jellemzőknek köszönhetik:

- Harvard architektúra
- hosszú (16 bites) utasításkód
- egyszavas utasítások
- az utasítások egyetlen belső órajelciklus ($\mathbf{t}_{CY}$) alatt végrehajtódnak
- átfedéses utasításvégrehajtás
- csökkentett utasításkészlet
- különleges regisztermező

A Harvard architektúrájú számítógépekben a programmemória és az adatmemória külön van kialakítva. Ez nagyban gyorsítja a programfutást, hiszen a két különálló memórián egy

időben is végrehajthatóak a műveletek. Ezen felépítés másik előnye, hogy a memóriabuszoknak nem feltétlenül kell azonos szélességűeknek lenniük. Az adatbusz 8 bites, a programbusz 16 bites. A 16 bit szélességű programbusz természetesen 16 bites utasításhosszat jelent. Ez teszi lehetővé, hogy A PIC18-as sorozat szabványos utasításkészletében a legtöbb utasítás egyszavas legyen (létezik 4 kétszavas utasítás is, melyek két tárhelyet foglalnak a program memóriában). Egyetlen 16 bites utasításban ugyanis elhelyezhető 7, 8 vagy akár 11 bites konstans is.

A PIC mikrovezérlőkben az utasítások végrehajtása két belső ciklus ($\mathbf{t}_{CY}$) alatt megy végbe. Az elsőben az utasításbeolvasás, a másodikban pedig a tényleges végrehajtás történik meg. A programfutás gyorsítása érdekében a PIC mikrovezérlőkben a beolvasó és a végrehajtó egységet a szükséges mértékben szétválasztották. Így lehetővé vált az átfedéses utasítás végrehajtás (Pipelining), így az egyszavas utasítások 1 gépi ciklus alatt hajtódnak végre, kivéve azokat az elágaztató utasításokat, amelyeknél a feltétel igaz, vagy az utasítás eredményeképpen a PC tartalma módosul. Ezekben az esetekben a végrehajtás két gépi ciklust igényel. A kétszavas utasítás két gépi ciklus alatt hajtódik végre. Minden utasítás négy oszcillátor periódusból áll. Tehát, ha az oszcillátor frekvenciája 4MHz, akkor egy normál utasítás végrehajtási ideje 1μs. Azon elágaztató utasítások esetében, amelyeknél a feltétel igaz, vagy az utasítás eredményeképpen a PC tartalma módosul, a végrehajtási idő 2μs. Kétszavas elágaztató utasításoknál igaz feltétel esetén a végrehajtási idő 3μs.

A PIC mikrovezérlők egységes regiszterkialakítása és jól tervezett utasításkészlete lehetővé teszi, hogy 75 szabványos plusz 8 kiterjesztett utasítással (a PIC16xxx sorozat esetén mindössze 35) megoldható legyen minden elvárható programfeladat. Ezek a mikrovezérlők úgynevezett RISC (Reduced Instruction Set Computer), vagyis csökkentett utasításkészletű CPU-val vannak felépítve.

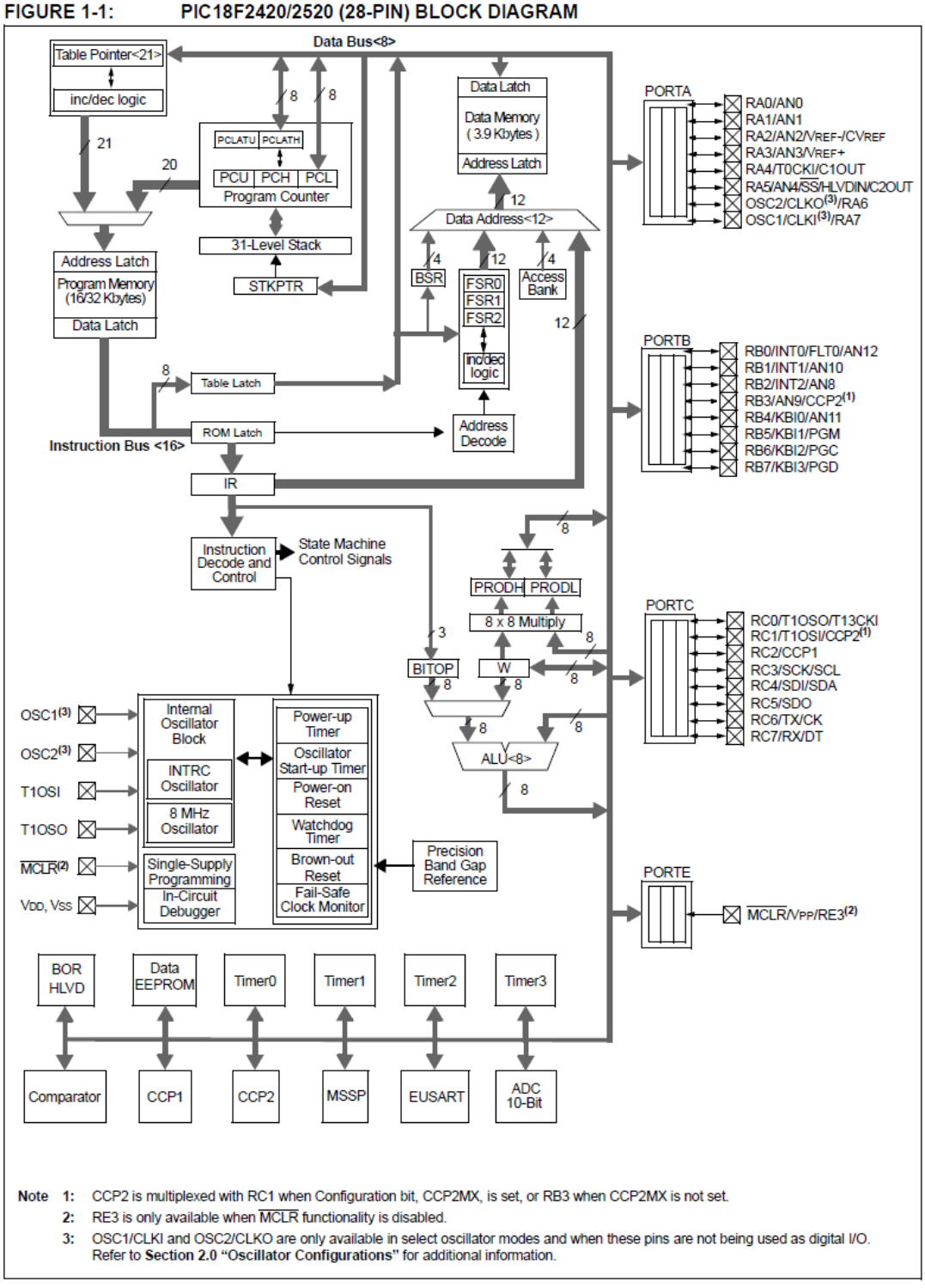
- A csökkentett utasításkészlet előnyei:
- könnyen megtanulható;
- gyors működés;
- az egy belső ciklus alatt végrehajtódó egyszavas utasítások miatt a program futási ideje könnyen kiszámítható.

A csökkentett utasításkészlet hátrányai:

- magas szintű nyelven történő programozás esetén komplex fordítóprogram szükséges és viszonylag hosszú a fordítási idő;
- a vezérlésorientált utasításkészlet miatt egyéb jellegű feladatok megoldása nehézkesebb.

Az egységes regiszterkialakítás főbb jellemzői: Az adatmemóriában két különböző funkciójú - általános és vezérlő - regisztermező van elhelyezve. Kezelés szempontjából azonban ezek nincsenek megkülönböztetve. A hardvervezérlési és címzési feladatok is az adatmemóriában található regiszterek segítségével vannak megvalósítva. Minden regiszter közvetlen és közvetett címzéssel is elérhető. Minden regiszter bit címezhető, tehát bitenként is módosítható, illetve vizsgálható. [10]

FIGURE 1-1: PIC18F2420/2520 (28-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when Configuration bit, CCP2MX, is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when MCLR functionality is disabled.

3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

2. ábra: a PC18F2420/2520 (28 lábú kivitel) blokkvázlata [6]

Programozás

Minden digitális számítógép, bármilyen bonyolult feladatot is hajtson végre, végül is a bináris számokat utasításként értelmezve végzi el a műveleteket a szintén bináris formátumú adatokon. Ez teljesen nyilvánvaló, mivel a gépben csak bináris 0 és 1 alakú információ feldolgozása lehetséges. Az ilyen formában előálló programot nevezzük gépi kódú programnak.

A számítógép programozásában rejlő összes lehetőség legjobb kihasználását a gépi kód teszi lehetővé. Ezen a szinten írhatók a gép működése szempontjából a leghatékonyabb programok, itt használhatók ki legjobban az egyes utasítások hatásai és mellékhatásai, itt alkalmazhatja a programozó a legszellemesebb megoldásokat és trükköket.

A gépi kódban történik programozás azonban az ember számára rendkívül nehézkes, készség szintű elsajátításához, a programozói rutin megszerzéséhez igen hosszú idő szükséges. Ezt a fáradtságos munkát még az is nehezíti, hogy meg kell tanulni az utasításoknak megfelelő bináris, oktális, vagy hexadecimális számrendszerbeli utasításkódokat, és ki kell számítani a programban szereplő címek abszolút, vagy relatív értékeit.

A gépi kódú programozás ezen hátrányait – az előnyök megtartása mellett – az assembly nyelven történő programozással küszöbölhetjük ki.. Az assembly a legegyszerűbb programozási nyelv, amely lehetővé teszi, hogy a gépi kódban számmal megadott utasítások helyett könnyen megjegyezhető neveket alkalmazzunk Ezeket, a kódneveket – melyek általában az utasítás funkciójának a rövidítései – *mnemonikoknak (emlékeztetőknek)* nevezzük. Az assembly nyelv lehetővé teszi azt is, hogy a program egyes belépési pontjaira nevekkel hivatkozhatunk, ún. *címkéket* rendelhetünk hozzá. Ezen kívül szintén nevekkel hivatkozhatunk különböző regiszterekre, bájtokra, bitekre, azaz *szimbólumokat* rendelhetünk hozzájuk. Így a programozó megszabadulhat a gépi kódú programozásnál szükséges címszámítástól. Ezzel a gépi kódú programozás elmlített két hátránya ki lett küszöbölve: nem kell az utasításokhoz tartozó számértékeket memorizálni és nem kell a programban címértékeket számolni. [9]

Az utasítások elnevezései természetesen kötöttek, ezt a gyártó definiálja. A PIC mikrovezérlőknél az adatmemória két részre oszlik:

SFR (Special Function Registers) – speciális funkciójú regiszterek

GPR (General Purpose Registers) – általános célú regiszterek

A speciális funkciójú regiszterek elnevezéseit a Microchip definiálta (pl. a portok elnevezései: PORTA, PORTB, stb.). Az elnevezések a <típus>.inc fájlban, az MPLAB program könyvtárában találhatók meg az egyes típusokhoz. Az általános célú regisztereknek tetszés szerinti betűvel kezdődő, akár bitenkénti elnevezést is adhatunk. A címkék, szimbólumok elnevezése tetszőleges, azonban érdemes rövid, lényegretörő (beszédes) elnevezéseket választani, ezek ugyanis megkönnyítik a forrásprogram „olvasását", ami a későbbiekben lényegesen egyszerűbbé teszi a program elemzését és a hibakeresést. A fordítóprogram feladata, hogy értelmezze, és gépi kódra fordítsa az általunk szimbolikus formában megírt programot. A tetszőleges szövegszerkesztővel megírt programot *forrásprogramnak* nevezzük, assembly-ben a kiterjesztése asm. Ezt a forrásprogramot alakítja át a fordítóprogram *tárgyprogrammá.* Az assembly programnyelvben a fordítóprogramot *assembler*-nek nevezik. Elnevezése az angol assemble (összeállítás) szóból származik.

Mint minden nyelvnek, az assembly nyelvnek is van nyelvtani formai szabályrendszere, ún. szintaktikája. Az assembly program sorokból áll, melynek felépítése a következő:

címke, művele,t operandus, megjegyzés (label, operation, operand, comment) pl: START: MOVLW .15 ; a W-be 15-öt töltünk.

Egy programsor tartalmazhat *utasítást,* amelyet a gép végrehajt, vagy *direktívát*, amely a fordítóprogramnak szól. Az egyes mezőket minimum egy szóközzel kell elválasztani, de sokkal áttekinthetőbbé válik a forrásprogramunk, ha tabulátorokkal igazítjuk a mezőket.

Az egyes mezők értelmezése a következő:

- Címkemező: ide írjuk azokat a neveket (címeket), melyekkel a program lényeges pontjait (pl. szubrutin belépési pont, ugrási cím, stb.) meg akarjuk jelölni. A fordítóprogram a fordítás során a címkéhez a memóriabeli elhelyezkedési címét rendeli. Abban az esetben tehát, ha a program során bárhol erre a címkére hivatkozunk, akkor a memóriabeli címének értéke helyettesítődik be. A címke csak betűvel kezdődhet, és általában kettőspontra végződik (a Microchip assemblerénél a kettőspont elhagyható).

- Művelet (utasítás) mező: ide írjuk a megfelelő utasításokat, valamint a direktívákat.

- Operandusmező: itt kell megadni az előző mezőben szereplő utasításhoz, vagy direktívához tartozó operandusokat. Az operandusmezőben egy, vagy két operandus állhat, illetve lehet olyan eset is, hogy nincs operandus. Két operandus esetén azok egymástól való

elválasztására vesszőt kell használni. Az operandusmezőben a következő elemek – melyek kifejezéseket is alkothatnak – szerepelhetnek:

• Konstans (literál, állandó), amely lehet decimális, hexadecimális, oktális, vagy bináris

• Szövegkonstans: idézőjelek, vagy aposztrófok közé írt karaktersorozat, amely ASCII kódját adja vissza a kifejezés

• Szimbólum: betűvel kezdődő elnevezés

• Regiszternév: a processzor belső regisztereinek, bitjeinek szimbolikus elnevezései (a gyártó definiálja)

• Feltétel (státusz) bit (Flag Bit): ezek valamilyen műveletvégzés eredményeképpen állnak be, az adott művelet után beállt állapotról adnak jelzést (pl. kinullázódott egy regiszter, átvitel történt összeadáskor, stb.), ezeket szintén a gyártó adja meg.

• Műveleti jelek:

+ összeadás

- kivonás

* szorzás

/ osztás

& logikai ÉS

^ logikai VAGY

• Speciális jelek

$ az utasításszámláló (programszámláló) aktuális értéke

- Megjegyzés mező: ide saját megjegyzésünket, magyarázó szövegünket helyezhetjük el. A megjegyzést mindig pontosvesszővel kell kezdeni. Minden pontosvessző után írt szöveg megjegyzésnek tekinthető, ilyen módon akár egy teljes sor is lehet megjegyzés. A programok megjegyzésekkel való ellátása, „kommentezése” nagyon fontos dolog. Ugyanis az assembly-ben írt programokban „első látásra” nagyon nehéz kitalálni az egyes programrészek funkcióit. Sokszor maga a programozó is elfelejti néhány nap múlva, hogy mit is akart itt csinálni! A jó kommentezés azonban nagyban megkönnyíti a hibakeresést, illetve a programjaink továbbfejlesztését.

A assemblyben megírt forrásprogramunkból az assembler állítja elő a tárgyprogramot. Ezt a műveletet fordításnak nevezzük. Az assembler a fordítást több menetben hajtja végre. Az első

menetben az assembler végigolvassa a forrásprogramot és felépíti a szimbólumtáblázatot. Az assembler két táblázatból dolgozik, az egyikben az állandó szimbólumok találhatók, a másikban pedig a felhasználói szimbólumok. Itt rendelődnek hozzá a szimbólumokhoz a megfelelő értékek a szimbólumok memóriabeli elhelyezkedése alapján. Az olvasás befejezésekor minden szimbólumnak értéke kell, hogy legyen. Azokat a szimbólumokat, amelyek nem kaptak értéket nem definiált szimbólumnak (undefined symbol) nevezzük. A második menetben történik a tulajdonképpeni tárgyprogram létrehozása. Ilyenkor az assembler újra végigolvassa a forrásprogramot és átalakítja a programsorokban szereplő utasításokat gépi kódra. A programsorok értelmezése során az assembler felismeri a különböző *szintaktikai* hibákat. Ezeket hibaüzenet formájában meg is jeleníti. A hibalista és a tárgyprogram mellett a második, vagy a harmadik menetben elkészül a fordítási lista is. Ez tulajdonképpen a forrásprogram másolata, amely azonban sorról sorra tartalmazza a helyszámláló aktuális értékét, valamint az adott sor fordítása révén nyert gépi kódot, illetve hibás sor esetén a hibaüzenetet. A fordítási listában megtaláljuk még a szimbólumok névsorba rendezett táblázatát is a hozzárendelt értékekkel együtt. A lista végén pedig egy statisztikát olvashatunk a programról. A fordítási lista tehát lényegében a fordítás dokumentuma.

A mikrovezérlők alkalmazásakor általában nem áll rendelkezésre az adott mikrovezérlőn futó assembler program, amivel a fordítást el tudnánk végezni. Ilyenkor az a megoldás, hogy egy másik gépen (pl. PC) végezzük el a fordítást.

Ehhez megfelelő fejlesztőkörnyezet szükséges, amely rendelkezik egy ún.kereszt-fordítóval (cross-assembler). A PIC mikrovezérlőkhöz a Microchip által kifejlesztett PC-n futó MPASM kereszt-fordító a leghatékonyabb megoldás. Az MPASM az integrált MPLAB fejlesztőkörnyezet része (lásd 3.1 fejezet). A kereszt-fordítóval történő programfejlesztés során valamilyen szövegszerkesztővel kell megírni a forráskódot (jelen estben .ASM kiterjesztésű fájl). A forrásprogram megírható az MPLAB beépített szövegszerkesztıjével, vagy bármilyen más szövegszerkesztővel, a lényeg az, hogy *text* formátumú legyen. A lefordításhoz ennek a fájlnak meg kell felelnie mind a formai (szintaktikai), mind az utasítások, operátorok, direktívák helyes használatát megkívánó szemantikai követelményeknek. A fordítás során többféle fájl keletkezik :

- A fordítási listát tartalmazó nyomtatható szöveges .LST kiterjesztésű fájl
- A hibaüzeneteket tartalmazó .ERR kiterjesztésű fájl

- A szimbólumokat és debug információkat tartalmazó .COD kiterjesztésű fájl

- A tárgyprogram .O kiterjesztéssel, amit az összefűző (linker) program használ fel a továbbiakban

- A gépi kódot tartalmazó, jelen esetben .HEX kiterjesztésű fájl, amit a linker hoz létre az előzőleg külön-külön lefordított modulokból [4] [9].

Utasításkészlet

A PIC 18F2420/2520/4420/4520 mikrovezérlők utsításkészlete nemcsak a PIC18-as sorozat 75 szabványos utasításból álló alap utasításkészletét tartalmazza, hanem még további 8 kiterjesztett utasítással is rendelkezik a rekurziv kódok optimalizálásához és a szoftververem kezeléséhez.

A PIC18-as sorozat szabványos utasításkészlete a korábbi PIC MCU utasításkészletének továbbfejlesztett változata, mely lehetővé teszi az egyszerű migrálást ezen korábbi verzióról. A legtöbb utasítás egyszavas (16 bit), de létezik 4 kétszavas utasítás is,melyek két tárhelyet foglalnak a program memóriában. Minden egyszavas utasítás két részből áll: egy műveleti kódból amely meghatározza az utasítás típusát és egy vagy több operandusból amelyre a művelet vonatkozik.

Az utasításkészlet erősen ortogonális és négy alapcsoportból áll:

- **Bájt orientált** műveletek
- **Bit orientált** műveletek
- **Konstans** műveletek
- **Vezérl**ő utasítások

A legtöbb **bájt orientált** utasításnak három operandusa van:

1. **A** fájlregiszter (“f” határozza meg)
2. Az eredmény helye (“d” határozza meg)
3. Az elért memóriaterület (“a” határozza meg)

A fájlregisztert jelölő „f” azt a regisztert jelenti, emelyiken az utasítást végre kell hajtani.

A célt jelölő (d) határozza meg, hogy hogy melyik regiszterbe kerüljön az eredmény. Ha d=0 az eredmény a WREG (working register, az AC regiszter PIC elnevezése) regiszterbe kerül ha pedig d=1 akkor az "f" által meghatározott fájlregiszterbe.

Minden **bit orientált** utasításnak három operandusa van:

1. A fájlregiszter ("f" határozza meg)
2. A fájlregiszter egy bitje ("b" határozza meg)
3. Az elért memóriaterület ("a" határozza meg)

A „b" jelöli ki, hogy melyik biten kell az adott műveletet elvégezni, "f" pedig azt a fáljregisztert határozza meg amelyik az adott bitet tartalmazza.

A **konstans** műveletek a következő operandusokat használhatják:

- A fájlregiszterbe töltendő konstans értéke ("k" határozza meg)
- A felhasznált FSR regiszter, amibe a konstanst töltjük ("f" határozza meg)
- Nincs operandus ("**-"** jelöli)

A vezérlő utasítások a következő operandusokat használhatják:

Programmemória cím ("n" határozza meg)

A CALL vagy a RETURN utasítás módja ("s" határozza meg)

Tábla írás és olvasás módja ("m" határozza meg)

Nincs operandus ("**-"** jelöli)

Fejlesztőkörnyezet

A fejlesztőkörnyezet a következő eszközökből áll:

- Fejlesztő program
- Programozó készülék
- Fejlesztőkártya

A fejlesztőkörnyezet legfontosabb szoftverelemét az **MPLAB IDE** program adja, amelyet a Microchip cég ingyenesen biztosít az általa gyártott eszközök szoftvereinek fejlesztéséhez. Ez a program tartalmaz egy szövegszerkesztőt a programok megírásához, egy keresztfordítót a forrásprogramok gépi kóddá alakításához, valamint egy beépített szimulátort a programok teszteléséhez. A program letölthető a Microchip honlapjáról (www.microchip.com). A programot

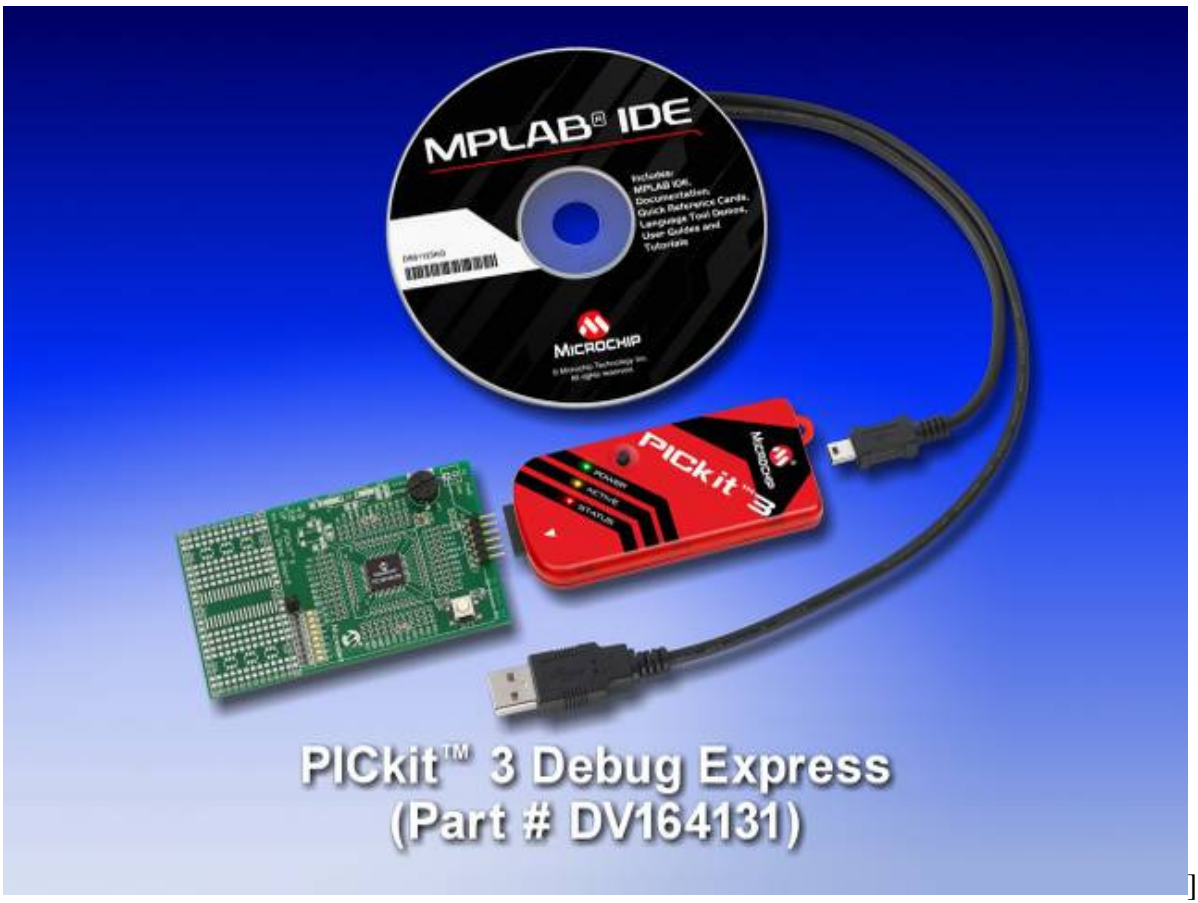
folyamatosan frissítik. Az MPLAB hasznos kiegészítője az Application Maestro program, mely előre megírt részprogramok tárháza. Ezeket a projektbe illesztve sok időt és munkát spórolhat meg a fejlesztő.

A PICkit 3 programozó a Microchip költséghatékony, PIC programozó és hibavadász (debug) készüléke. A PICkit USB 2.0 porton keresztül kapcsolódik a számítógéphez, mely lehetővé teszi a firmware frissítést ill. a fejlesztőeszköz és céláramkör tápellátását. A támogatott főbb in-circuit debug funkciók: valósidejű futtatás

- Megállítás
- egyesével léptetés
- adat és cím töréspont
- stopper

Az ICSP technológiának köszönhetően a PICkit 3 könnyen csatlakoztatható a fejlesztőkártyákhoz. [7]

A fejlesztőkártya, vagy próbapanel a felprogramozott PIC teszteléséhez szükséges környezetet biztosítja. Általában LED-eket, LCD kijelzőt, potmétereket, kapcsolókat tartalmaz.



]

3. ábra: fejlesztő készlet [8

A fűtési alrendszer

Házunk teljes lakóterületén padlófűtés van, melyet minden helyiségben radiátoros fűtés egészít ki. Egy jól működő padlófűtési rendszer kisebb energiafelhasználás mellett jobb hő eloszlást és nagyobb komfortot biztosít, mint a radiátoros fűtés. Ezért választottuk a padlófűtést elsődleges fűtésnek.

Mikor a rendszert átadták mindkét fűtési rendszert egyetlen termosztát vezérelte, melynek eredményeként egy radiátorral fűtött házat és hideg padlót kaptunk. A radiátorok és a padlófűtés egyidejűleg működött. A radiátorok 15-20 perc alatt felmelegítették a légteret és a kazán lekapcsolt. A padló pedig szép lassan lehűlt, hiszen nem csak a fűtési idő volt rövid, de a kazán

által leadott energia nagy részét is - a két rendszer eltérő hőátadási jellemzőinek következtében - a radiátorok vették fel. A padlófűtés a nagytömegű beton és kerámia burkolat miatt lassan melegszik fel és hosszasan sugározza ki a felvett energiát (tipikusan 200 W/ m²), míg a radiátor közvetlenül a légteret melegíti. Az energiaátadás is intenzívebb (tipikusan 2 kW/ m²).

Kézenfekvő megoldás volt a két fűtési rendszer függetlenítése. A padlófűtéshez készítettem egy 0,1 °C felbontású digitális termosztátot. Ennek beállítását a külső hőmérsékletnek megfelelően kézzel kellet elvégezni. Annak ellenére, hogy ez a termosztát 0,1 °C felbontású volt a padló hőmérséklete - a padlófűtés tehetetlenségének köszönhetően - 1,2-1,5 °C–ot is ingadozott. A radiátoros fűtést egy kereskedelemben kapható 0,5 °C felbontású digitális termosztát vezérelte. Ha a padlófűtés prioritása volt nagyobb, akkor hideg időben a hosszabb fűtési periódus alatt kellemetlenül lehűlt a légtér. Ellenkező esetben pedig a padló hűlt ki. Az eredő hőmérsékletingadozás meghaladta a 1,5 – 2 °C–ot is. Ez sajnos a fűtési számlán is meglátszott.

A hőmérséklet ingadozás szerepe jelentős az energiatakarékosság szempontjából. Az 1°C túlfűtés éghajlatonként eltérő mértékű energiafogyasztás növekedést jelent. Magyarországon 1°C túlfűtés 6% energia többlettel jár. [11]

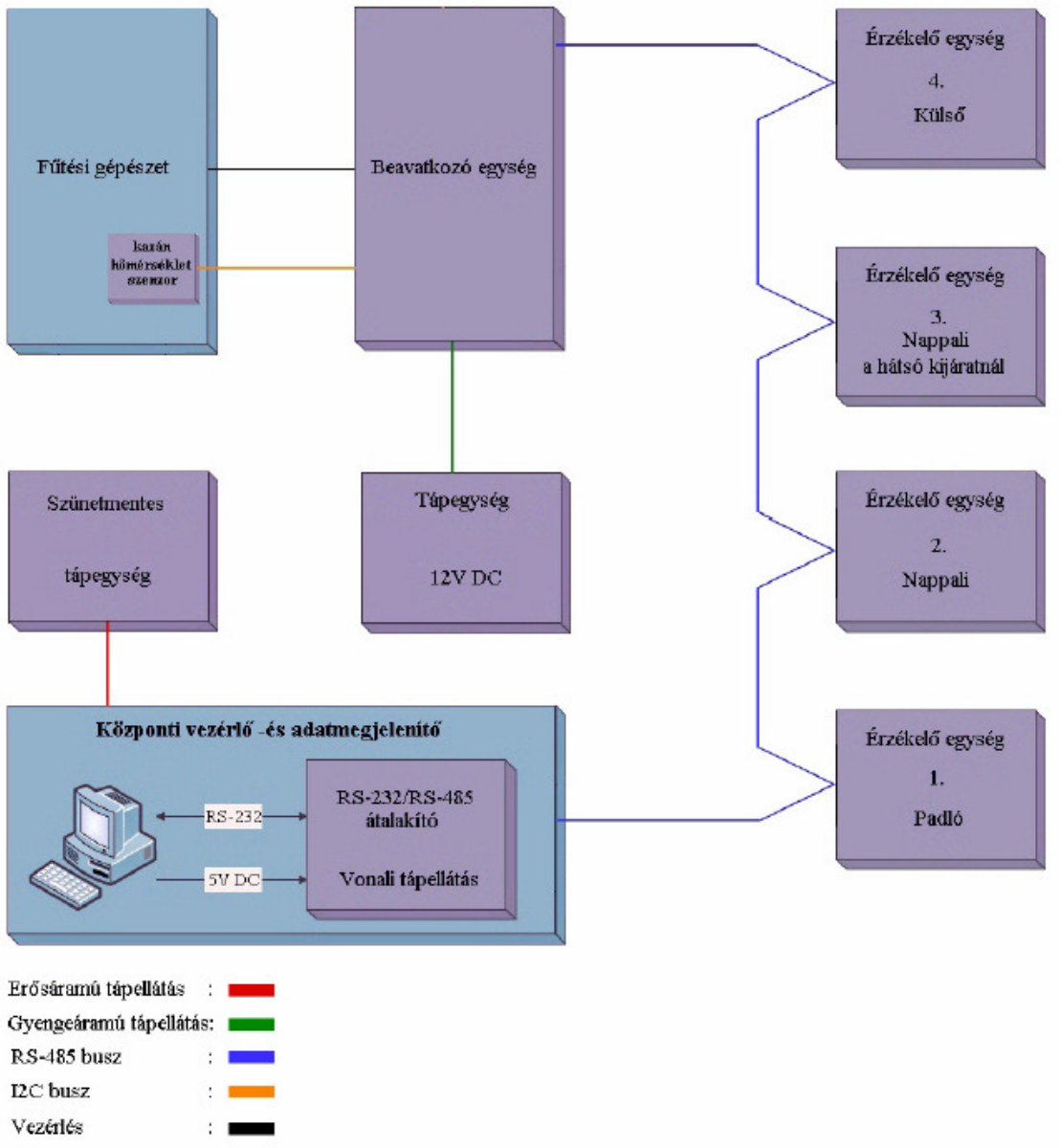
Ezért döntöttem egy olyan rendszer építése mellett, mely a két eltérő fűtési módot automatikusan vezérelve stabilabb hőmérsékletet biztosít.

Az itt bemutatásra kerülő rendszer korábbi kísérletek eredményeire és tapasztalataira épül. A kisebb hőingadozás eléréséhez egy nagyságrenddel jobb felbontású érzékelők alkalmazása és a külső hőmérséklet mérése is szükséges.

Négy hőmérőt alkalmaztam:

- A padló hőmérsékletének mérésére
- A nappali hőmérsékletének mérésére
- A nappali északi, „hideg” oldalán a hátsó kijárat mellett
- A ház északi homlokzatánál a külső hőmérsékletet mérésére

A fűtési alrendszer blokkvázlata a 4. ábrán látható.



4. ábra: a fűtési alrendszer

Működése

Az érzékelő egységek mérési ciklusideje nyolc másodperc, tehát új mérési eredmény nyolc másodpercenként áll elő.

A beavatkozó egység vezérli a kazán teljesítményét, a padlófűtés keringető szivattyúját. Ezen kívül képes még hat elektromos radiátor szelep vezérlésére, áramszünet detektálására és hibaüzenet küldésére, ha a kazán nem melegít. PC hiba esetén, ha több mint 5 percig (320s) nem érkezik parancs a PC-től, akkor tartalék üzemmódra kapcsol. Ebben az üzemmódban a hiba elhárításáig egy hagyományos termosztát vezérli a padlófűtést.

A központi vezérlő és adatmegjelenítő egység feladata az érzékelő egységek adatai alapján a beavatkozó egységen keresztül a fűtési gépészet vezérlése, valamint a mért hőmérsékleti értékek, üzemmódok (pl. padlófűtés) és üzenetek (pl. szellőztetés) megjelenítése. A vezérléshez alkalmazott PC-be TV kártya és TV kimenettel rendelkező grafikus vezérlő is van építve. A videó jelet egy - a videó magnókban is megtalálható - modulátorra vezetve a házban található bármelyik televízión ellenőrizhető a rendszer állapota, miközben kedvenc TV csatornánkat, DVD filmünket vagy a házunk előtti kamera képét nézzük.

Az érzékelők és a beavatkozó egység mikrovezérlői tápfeszültségüket közvetlenül a központi egységtől kapják. A szünetmentes tápellátásnak köszönhetően az egyes egységek áramszünet idején is működőképesek maradnak.

Kommunikáció

Az egységek közötti kommunikáció fizikai rétegének az RS-485-ös szabvány kétvezetékes változatát választottam.

Az 1. táblázatban a soros kommunikációs szabványok főbb jellemzőit hasonlíthatjuk össze.

EIA SZABVÁNY	RS-232	RS-422	RS-485
Működési mód	aszimmetrikus átvitel	szimmetrikus átvitel	szimmetrikus átvitel
Meghajtók és vevők száma egy vonalon	1 meghajtó 1 vevő (pont-pont)	1 meghajtó 10 vevő (pont-pont)	32 meghajtó 32 vevő (multidrop)
Max. kábelhosszúság	15 m	1200 m	1200 m
Max. adatsebesség	20 kBd	10 MBd	10 MBd
Max. Common Mode	Nincs megadva	+7 V , -7 V	+12 V , -7 V
Meghajtó feszültség	± 5 V tól ± 15 V	± 2 V min.	± 1,5 V min
Meghajtó terhelés	3 kΩ tól 7 kΩ	100 Ω min.	60 Ω min
Meghajtó Slew Rate	30 V/µs	Nincs megadva	Nincs megadva
Meghajtó kimeneti rövidzárási áram limit	500 mA Vcc vagy Test felé	150 mA Test felé	150 mA tól Test felé 250 mA Vcc felé
Vevő érzékenység	± 3 V	± 200 mV	± 200 mV
Vevő hiszterézis	1,15 V	50 mV	50 mV
Vevő bemeneti ellenállás	3 kΩ - 7 kΩ	4 kΩ	12 kΩ

1. táblázat: soros kommunikációs szabványok

A szabvány szimmetrikus átvitelt használ sodrott érpáron keresztül a vonalon több adó és vevő is lehet. Két vezetéken fél-duplex, négy vezetéken teljes-duplex összeköttetés valósítható meg, és 32 résztvevő lehet egy buszon (master vagy slave). A kommunikáció multi drop (üzenetszórásos) rendszerű. A maximum kábelhossz 1200 m. Ez a szabvány alkotja több, ipari környezetben is alkalmazott kommunikáció fizikai rétegét. Az RS-485 rendszer tipikus felhasználása, hogy egy számítógép (Master) több címezhető készüléket (Slave) vezérel ugyanazon a vezetéken keresztül. Annak elkerülésére, hogy több adó kezdjen ugyanabban az időpontban a vonalon (buszon) adni, egy kitüntetett eszköznek, a Master-nek kell az adási jogot biztosítania, csak egy adó lehet aktív állapotban, minden más adónak magas impedanciájú állapotban kell lennie. A Master címzett parancsokkal (protokollal) szólítja fel a Slave-eket az esetleges adásra.

A kétvezetékes rendszer fél-duplex hálózat, mivel csak egy átviteli út áll az adatcsere rendelkezésére, ezért egy időben mindig csak egy résztvevő tud adatot küldeni. Csak az

adatküldés befejezése után tud válaszolni egy másik résztvevő. A legismertebb két vezetéken alapuló rendszer a PROFIBUS. [12]

Jelátvitelre a szabvány által ajánlott sodrott érpárt (UTP .Unshielded Twisted Pair) alkalmaztam.

Az adaátvitel paraméterei:

- sebesség: 9600 bps
- adatbitek: 8
- paritás: nincs
- stopbitek:1

A protokoll Master – Slave rendszerű: A központi vezérlő-és adatmegjelenítő egység a Master. Minden kommunikációt ő kezdeményez: címzett parancsot küld a vonalon, melyet minden egység vesz. Csak a címzett válaszol. Minden megszólított Slave köteles válaszolni. A Master innen tudja, hogy a parancs célba ért. Ha egy Slave-től nem érkezik válasz, a Master újraküldi a parancsot. A harmadik próbálkozás után a vezérlőegység hibaüzenetet küld a felhasználónak.

A vezérlőegység parancsának formátuma

1. kezdő karakter (@)
2. cím (00–FF)
3. parancs (csak a beavatkozó egység esetében)
4. ellenőrző szám (00-FF)
5. záró karakterek (CR LF)

Válasz:

1. kezdő karakter (@)
2. a megszólított egység címe (00–FF)
3. adapt, vagy adatok (pl: 14 01)
4. ellenőrző szám (00-FF)
5. záró karakterek (CR LF).

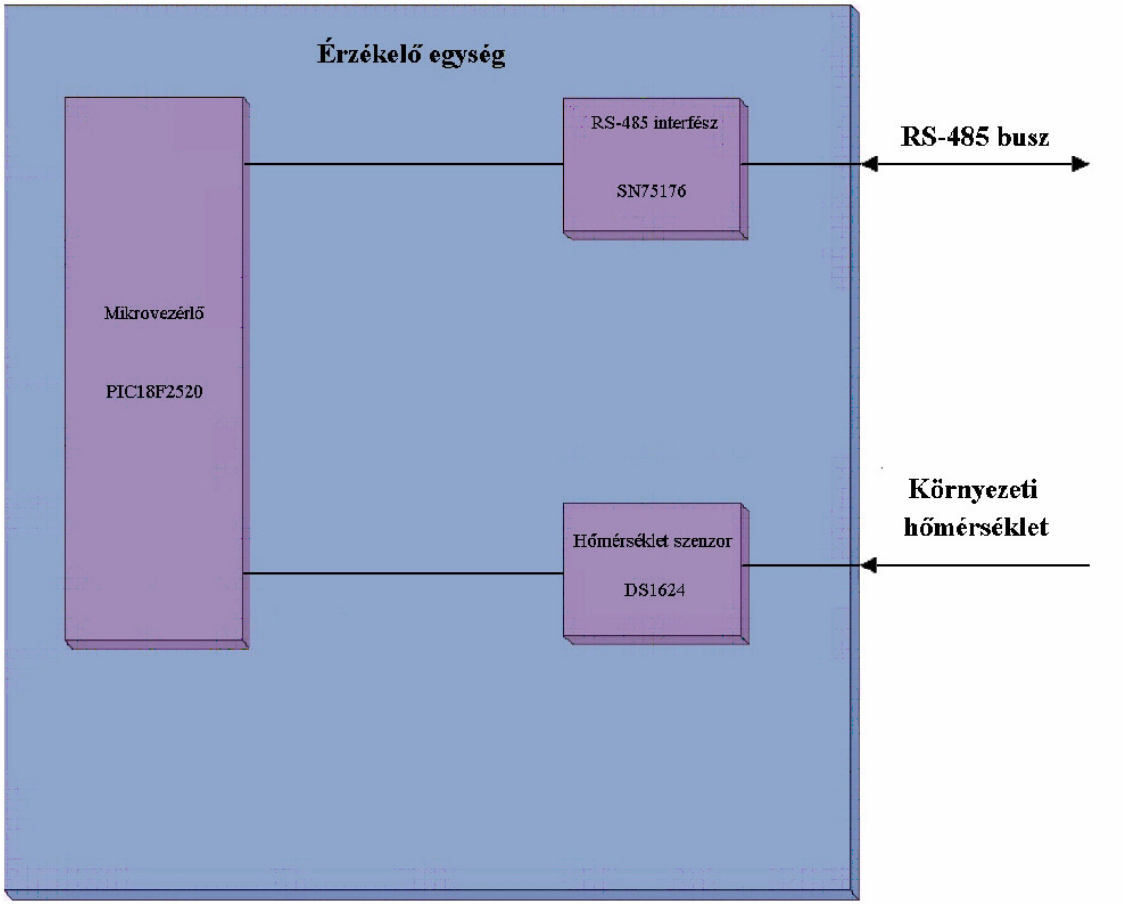
A hexadecimális adatok az üzenetekben ASCII karakterekké vannak konvertálva. Ez ugyan 100%-os redundanciát jelent, de az átvitelre kerülő adatmennyiség mindössze néhány bájt, így ez még a 9600 bps adatátviteli sebességnél sem okoz gondot, de két előnyt is biztosít:

1. Az adatforgalom egy egyszerű terminal programmal is monitorozható.
2. Az ASCII karakterré konvertálás bizonyos hibadetektálási lehetőséget is nyújt, mert a beérkező adatoknak 0-9 és A-F (ASCII) közé kell esniük.

Az ellenőrző szám a cím és parancs/adat bájtjaiból XOR művelettel áll elő. Ez a módszer egy gyenge hibadetektálást tesz lehetővé. Az OMRON PLC-k is ilyen hibadetektálást alkalmaznak. Az egység csak akkor fogadja el az üzenetet, ha egyezik a cím és egyik hibaellenőrző módszer sem mutat eltérést.

Mivel a kommunikáció fizikai rétege megbízható átvitelt biztosít még ipari környezetben is, ezért otthoni körülmények között igen alacsony hibaarányú adatátvitel valósítható meg.

Érzékelő egység



5. ábra: az érzékelő egység

Felépítése

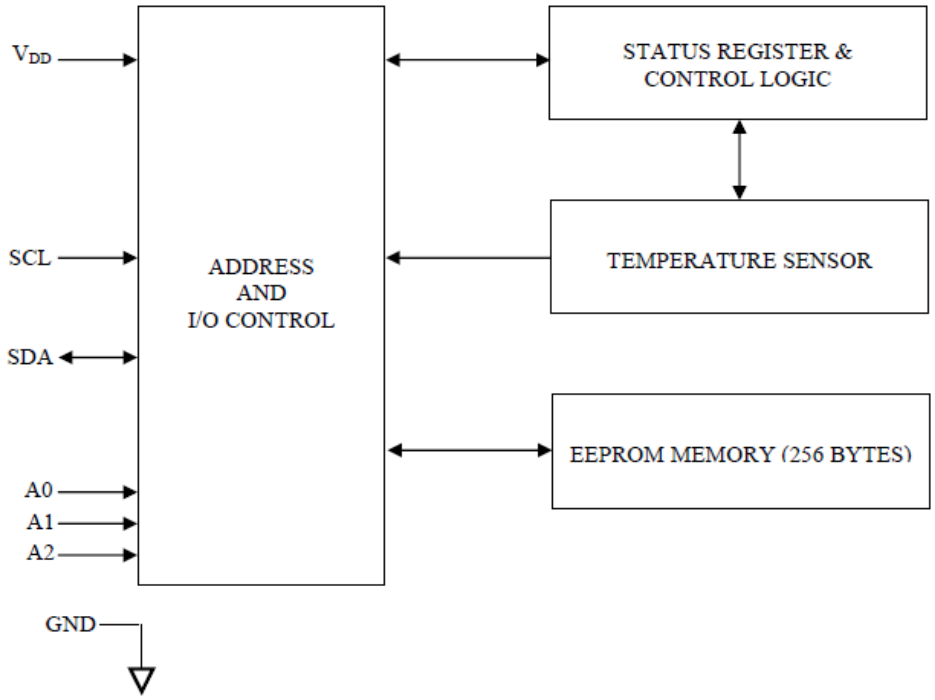
Az érzékelő egység (5. ábra) legfontosabb eleme a hőmérséklet szenzor.

Erre a feladatra kedvező ára és 13 bites felbontása (egy lépés 0,03125°C) miatt a MAXIM gyártmányú DS1624 digitális hőmérsékletmérő IC-t választottam. Az eszköz mérési tartománya -55°C -tól +125°C –ig terjed, ami a mi éghajlatunkon az év bármely szakában épületen belül és kívül egyaránt alkalmassá teszi hőmérsékletmérésre.

Az áramkör a hőmérsékletméréshez nem igényel külső alkatrészt.

Funkcionális felépítése a 6. ábrán látható

DS1624 FUNCTIONAL BLOCK DIAGRAM Figure 1



6. ábra a DS1624 funkcionális felépítése

A tok a hőmérséklet szenzoron kívül még egy 256 bájtos szabadon felhasználható, nem felejtő memóriát is tartalmaz.

A memória és a hőmérsékletmérő szenzor egy 2-vezetékes soros interfészen keresztül érhető el. A DS1624 az I2C kommunikációt támogatja. A tok 7 bites címmel rendelkezik, melyből a felső 4 bitet a gyártó definiálta: 1001. A maradék 3 bittel szabadon rendelkezünk (A2, A1, A0 bemenetek). Ebből adódik, hogy egy I2C buszra nyolc darab DS1624 IC csatlakozhat egy időben. [13]

Az I2C kommunikáció

Az Inter-IC (I2C vagy IIC) busz Az I2C kétvezetékes (SCL: órajel és SDA: adat) szinkron adatátviteli rendszer, melyet a Philips cég dolgozott ki, integrált áramkörök összekapcsolására. A soros adatkezelésű memóriák, A/D konverterek között sok ezzel az illesztővel van felszerelve. A részvevők címezhetőek, a cím lehet rögzített vagy programozható.

Alapvetően egy master és egy vagy több slave kommunikál egymással, de a rendszerben több master is lehet. A master-ek a busz feletti vezérlés jogáért versenyeznek egymással. Mindig a master küldi az órajelet az SCL vonalra. Az eredeti leírásban az adatátvitel sebessége 100 KHz volt, később ezt kiterjesztették 400 KHz-re, ma pedig általános az 1 MHz átviteli frekvencia alkalmazása. Az adattranszfert a master kezdeményezi, a Start feltétel kialakításával, amit egy cím követ, a cím utáni egy bites vezérlő jel mutatja meg, hogy a megjelölt slave-et a master írni vagy olvasni kívánja. A slave ACK (Acknowledge) jellel visszaigazolja a vételt, és ezután következik az írási vagy olvasási ciklus. Az adattranszfer végét a master a Stop feltétellel jelzi. A rendszer eredetileg 7 bites címekkel működik, az újabb igényeknek megfelelően később bővítették ki 10 bites címekre. [14]

A DS1624 Slave-ként működik az I2C kommunikációban, tehát szüksége van egy master-re, ami a kommunikációt vezérli (ez lesz a PIC). Az eszköz a hőmérsékletet egy alacsony hőmérsékleti együtthatójú (hőfok függetlennek tekinthető) és egy magas hőmérsékleti együtthatójú (erősen hőfokfüggő) oszcillátor frekvenciakülönbségének segítségével méri.

Az eredmény 13 biten kettes komplemens kódban olvasható ki, melyre a következő táblázatban láthatunk néhány példát.

TEMPERATURE/DATA RELATIONSHIPS Table 2

TEMP	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	01111101 00000000	7D00h
+25.0625°C	00011001 00010000	1910h
+½°C	00000000 10000000	0080h
+0°C	00000000 00000000	0070h
-½°C	11111111 10000000	FF80h
-25.0625°C	11100110 11110000	E6F0h
-55°C	11001001 00000000	C900h

2. táblázat

Mint a táblázatból is látható az eredmény két bájton kerül ábrázolásra, ahol a három legalacsonyabb bit mindíg nulla. A felső bájt (MSB) a hőmérséklet egész értékét tartalmazza. Az áramkör adatlapja a következő helyről tölthető le: http://datasheets.maxim-ic.com/en/ds/DS1624.pdf

A mikrovezérlő feladata a hőmérséklet szenzor vezérlésén túl a központi egységgel való kommunikáció is.

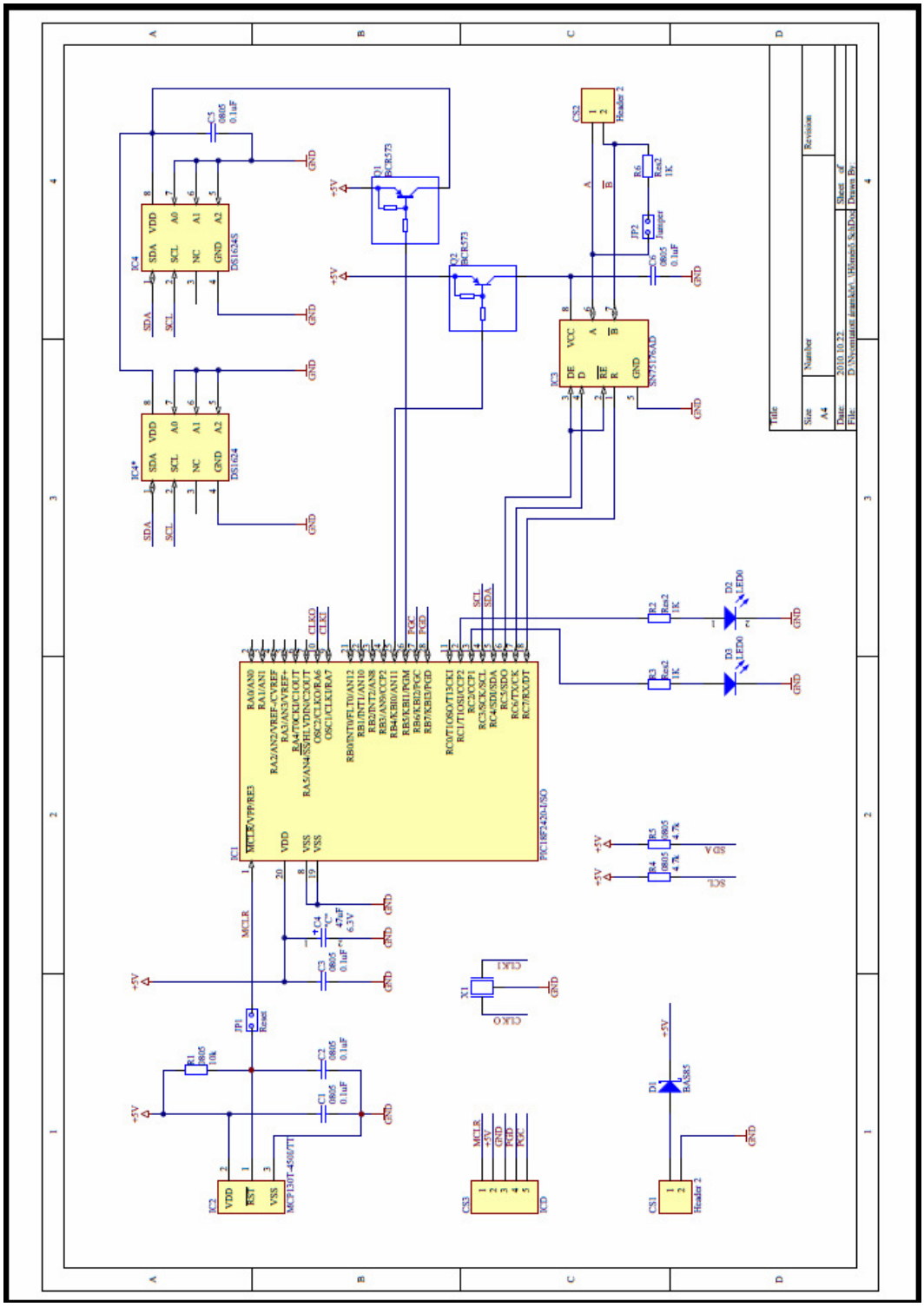
A DS1624 vezérléséhez a PC18F2420 mikrovezérlőt választottam. Ez a PIC külön porttal rendelkezik a vezérlő egységgel való soros kommunikációhoz és a hőmérséklet szenzor I2C kommunikációjához.

A szenzor másodpercenként méri a hőmérsékletet. A központi egység felé küldendő mérési eredményt nyolc mérés átlaga adja, tehát új mérési eredmény nyolc másodpercenként áll rendelkezésre. Ezzel a módszerrel kiküszöbölhető a mérési bizonytalanság (a DS1624 0,03125°C felbontással mér). Számítani is könnyű, hiszen bináris számrendszerben a nyolccal való osztás nem más, mint három bittel való eltolás jobbra. Átlagszámítás után a kétbájtos hexadecimális eredményt négybájtos ASCII kóddá konvertálja, majd kiszámítja az ellenőrző számot.

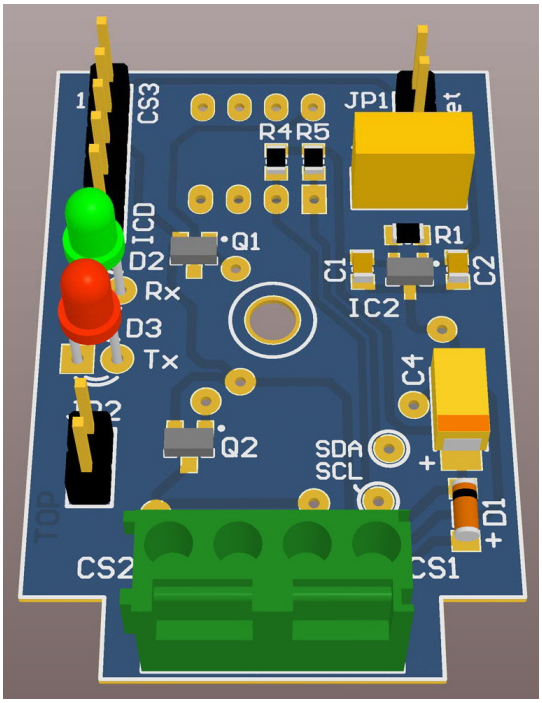
A PIC soros portja és az RS-485 busz közé a TEXAS gyártmányú SN75176 típusú differenciális busz adó-vevő került. Adatlapja a következő helyen található: http://focus.ti.com/lit/ds/symlink/sn75176a.pdf

Az egység kapcsolási rajza a 7. ábrán, beültetési rajza a 8. és 9. ábrán látható.

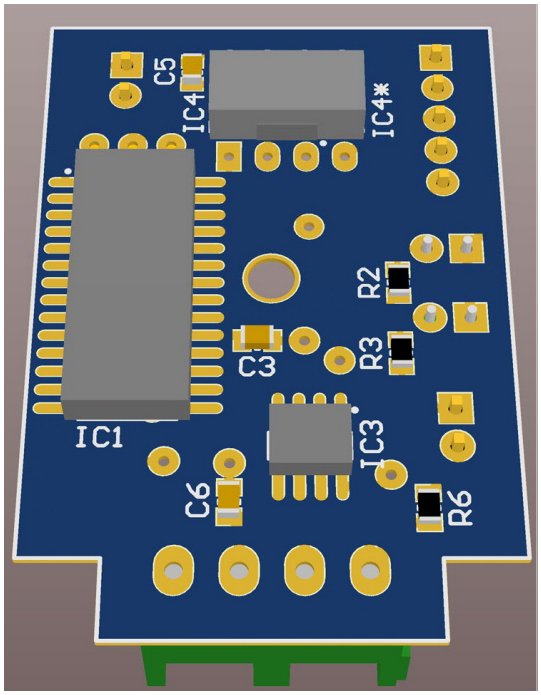
Az elkészült modul képei a 10.-13. ábrán láthatóak.



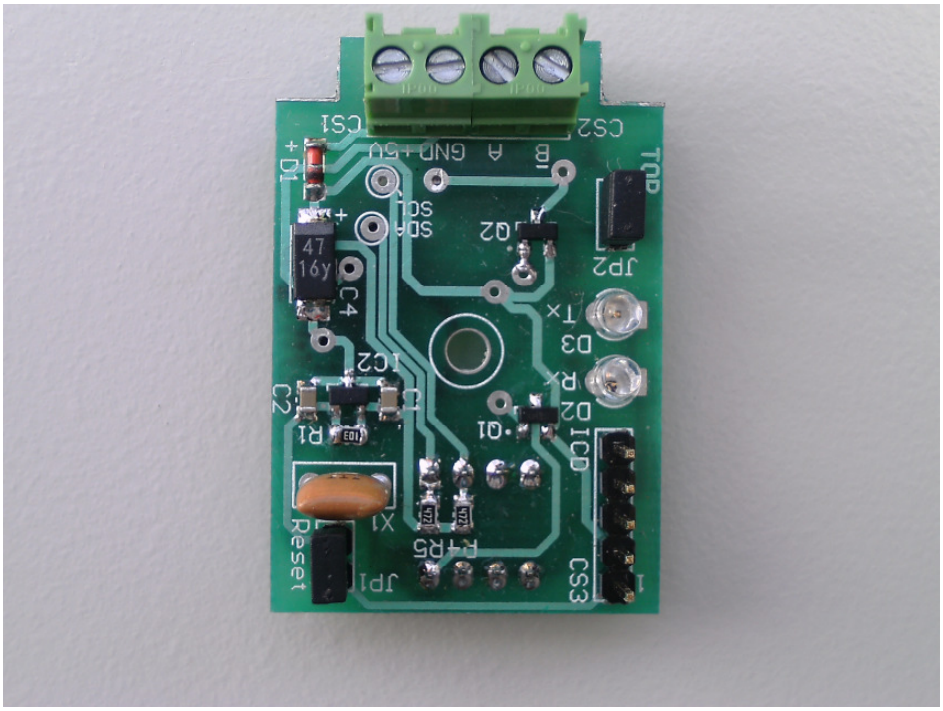
7. ábra: az érzékelő egység kapcsolási rajza



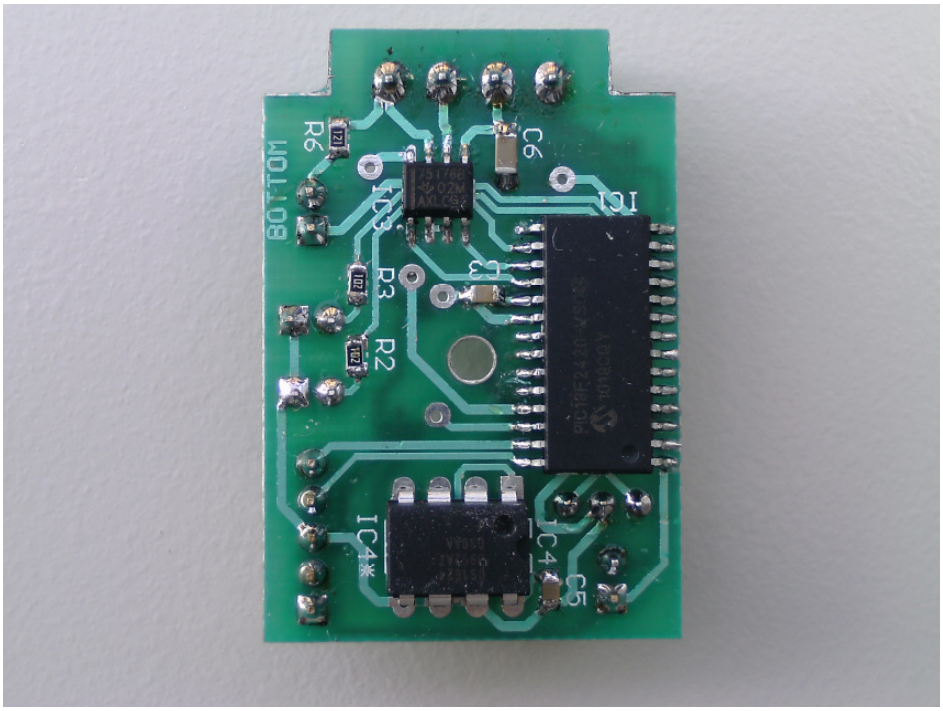
8. ábra: beültetési rajz (Top)



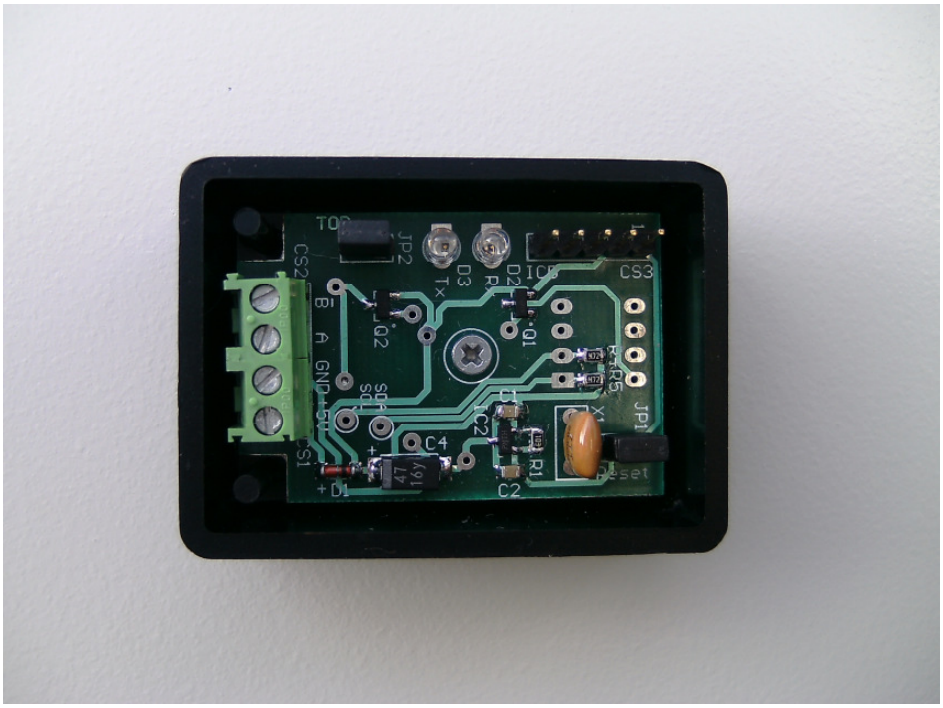
9. ábra: beültetési rajz (Bottom)



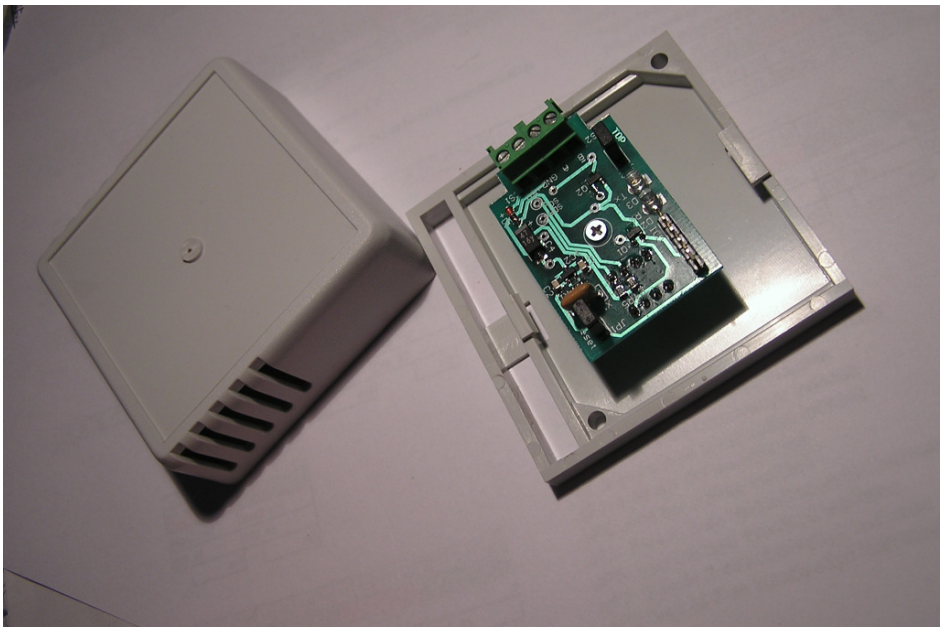
10. ábra



11. ábra



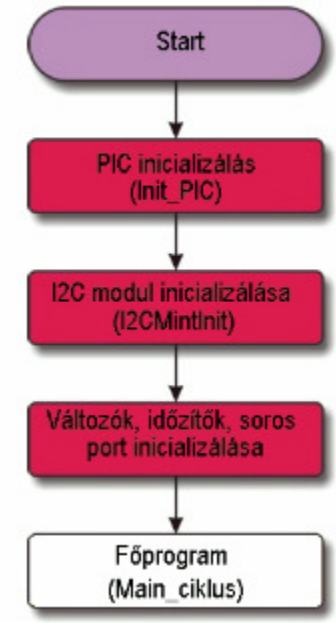
12. ábra



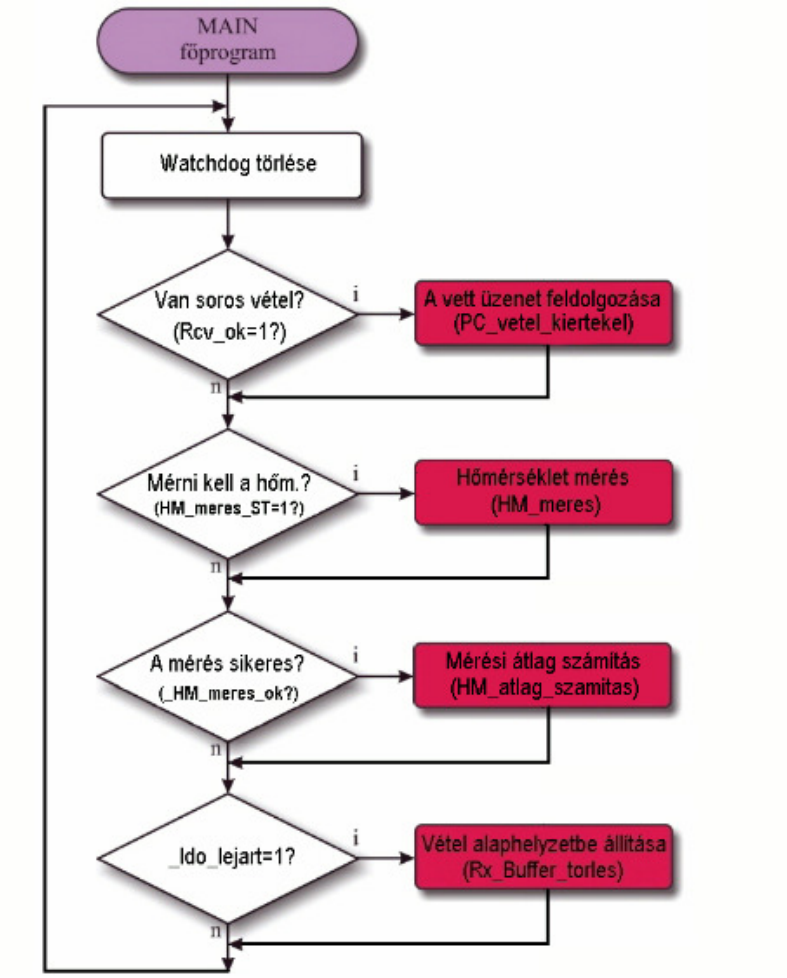
13. ábra

A program

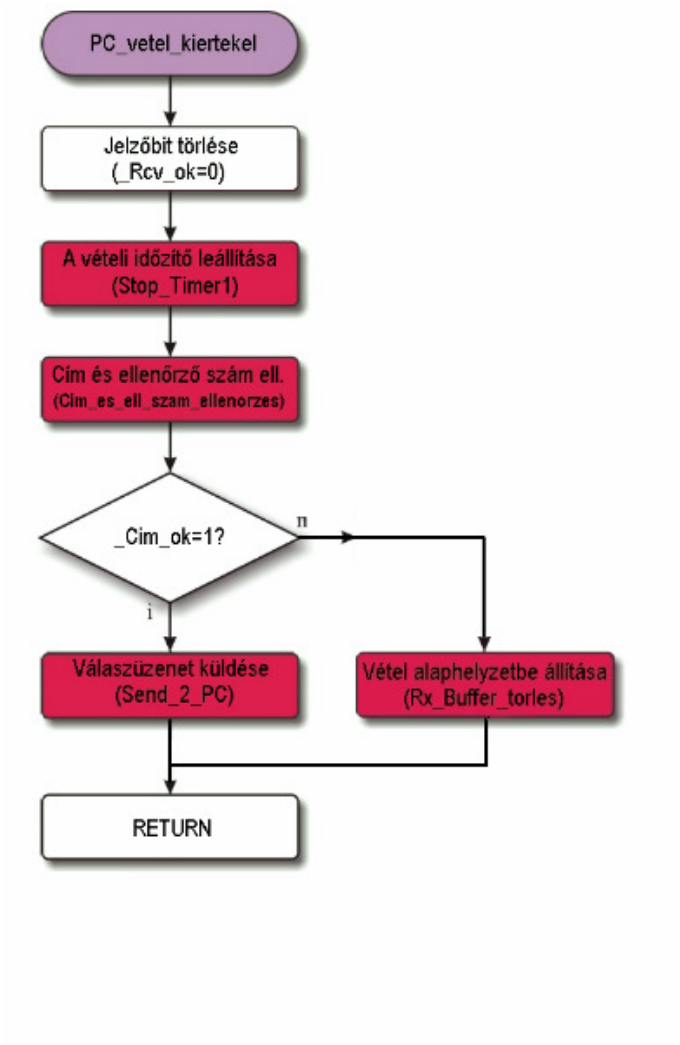
A program futása során bekövetkező különböző események (időzítő, vétel a soros porton) megszakítást generálnak. A megszakítás kezelő rutin a megszakítást okozó eseménynek megfelelően állítja be a jelzőbiteket. A main eljárás végtelen ciklusban fut. A beállított jelzőbitnek megfelelő eljárást hívja meg. A hőmérsékletmérési eljárást másodpercenként hívja meg a Timer0 időzítő. A mérés sikeres befejezése indítja az átlagszámítást, mely nyolc mérés átlagát számítja, tehát elküldendő mérési adat nyolc másodpercenként keletkezik. Az új átlagérték ellenőrző szám számítás és ASCII konverzió után bekerül az adási pufferbe. Így a bekapcsolás utáni nyolcadik másodperctől kezdve mindig van érvényes adat az adási pufferben Az első nyolc másodpercben olyan értéket küld (0x80), ami nem fordulhat elő normális működés esetén. Ezt a központi egység programjában kell kezelni. A soros porton történő vétel esetében a Receive_char eljárás ellenőrzi a kezdő és záró karakterek meglétét, valamint az üzenetvétel időtartamát, melynek rövidebbnek kell lennie 100ms-nál. A vett üzenet kiértékelése a cím és az ellenőrző szám ellenőrzésével kezdődik. Mivel a parancs csak a modul címét tartalmazza, ezért nincs szükség külön ellenőrző szám számításra, mert az megegyezik a címmel és még ASCII-HEX konvertálás is elhagyható. Így mindkét ellenőrzés a megfelelő ASCII karakterek összehasonlítására vezethető vissza. Az érzékelő egységek címe a 00-0F tartományba esik. Ez 16 egység együttes alkalmazását teszi lehetővé. A mostani rendszer 4 modult tartalmaz. Ezt a későbbiekben még 3 egységgel szeretném kibővíteni (így minden szobában lenne). A modul csak akkor válaszol, ha a vett parancs a megfelelő kezdő-és záró karaktereket tartalmazza, neki szól és az ellenőrző szám is egyezik. A program folyamatábrája 14.-18. ábrákon látható.


Start
PIC inicializálás
(Init_PIC)
I2C modul inicializálása
(I2CMintInit)
Változók, időzítők, soros
port inicializálása
Főprogram
(Main_ciklus)

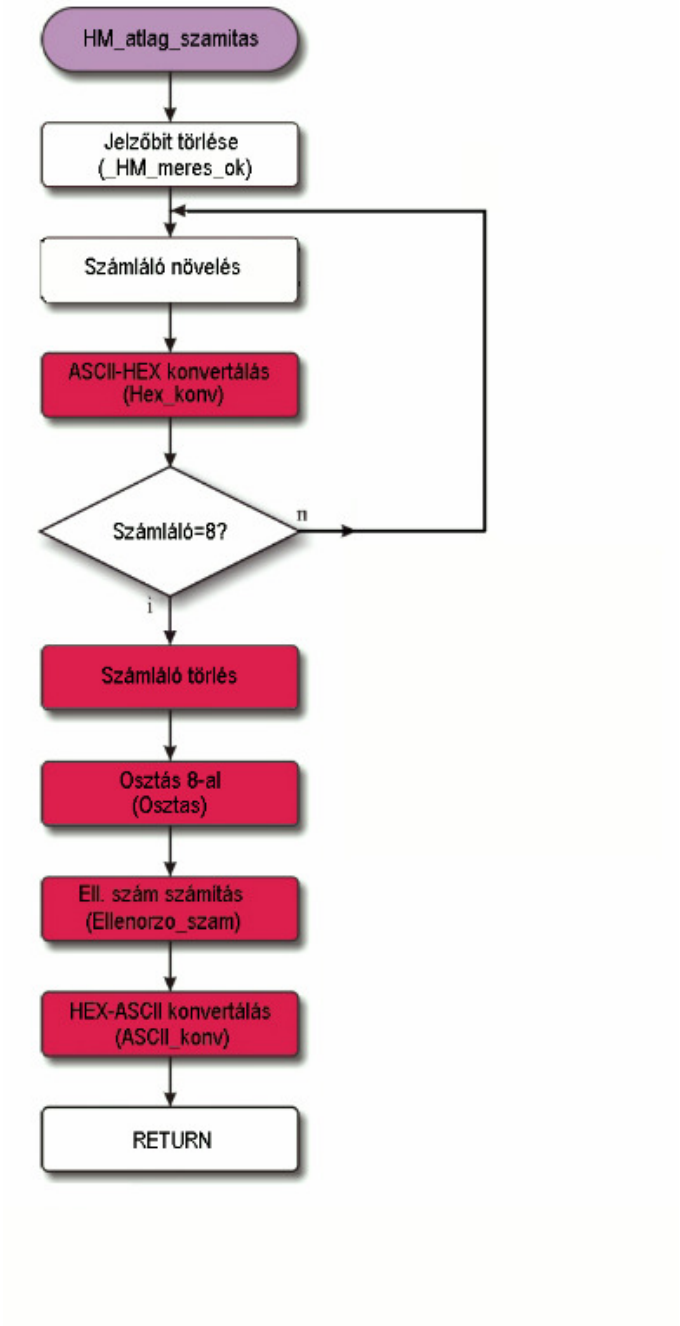
14. ábra


MAIN
főprogram
Watchdog törlése
Van soros vétel?
(Rcv_ok=1?)
i
A vett üzenet feldolgozása
(PC_vetel_kiertekel)
n
Mérni kell a hőm.?
(HM_meres_ST=1?)
i
Hőmérséklet mérés
(HM_meres)
n
A mérés sikeres?
(_HM_meres_ok?)
i
Mérési átlag számítás
(HM_atlag_szamitas)
n
_Ido_lejart=1?
i
Vétel alaphelyzetbe állítása
(Rx_Buffer_torles)
n

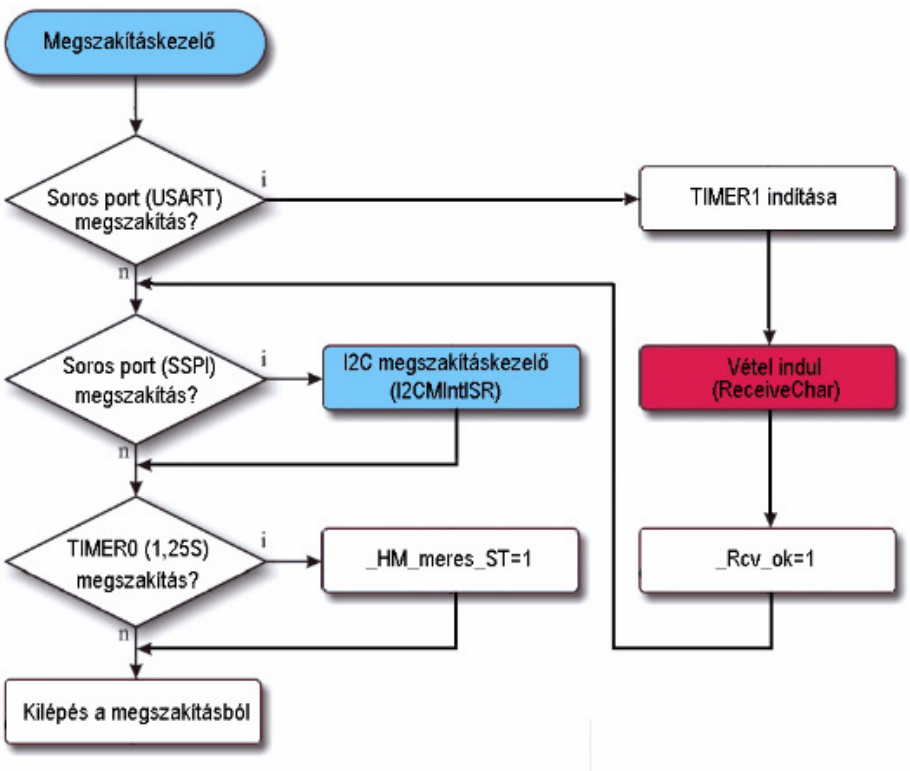
15. ábra


PC_vetel_kiertekel
Jelzőbit törlése
(_Rcv_ok=0)
A vételi időzítő leállítása
(Stop_Timer1)
Cím és ellenőrző szám ell.
(Cim_es_ell_szam_ellenorzes)
_Cim_ok=1?
n
i
Válaszüzenet küldése
(Send_2_PC)
Vétel alaphelyzetbe állítása
(Rx_Buffer_torles)
RETURN

16. ábra

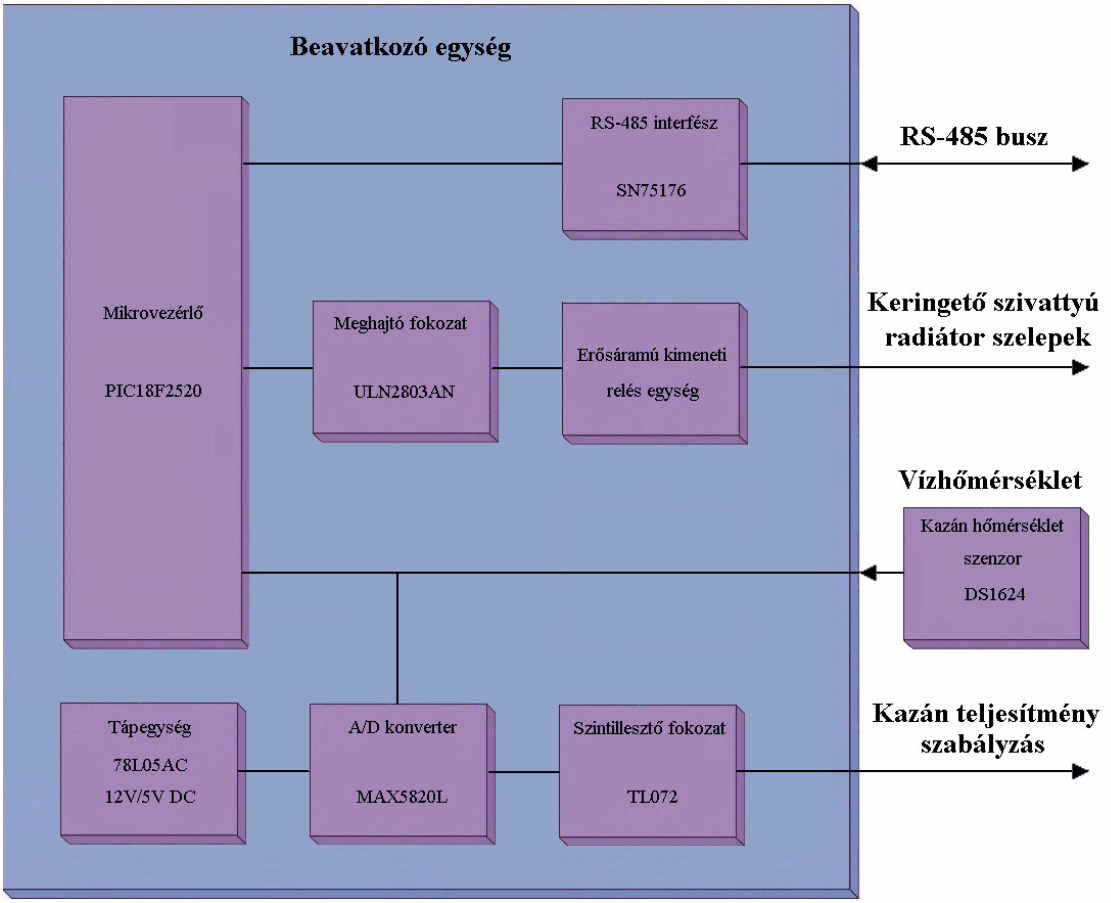

HM_atlag_szamitas
Jelzőbit törlése
(_HM_meres_ok)
Számláló növelés
ASCII-HEX konvertálás
(Hex_konv)
Számláló=8?
n
i
Számláló törlés
Osztás 8-al
(Osztas)
Ell. szám számítás
(Ellenorzo_szam)
HEX-ASCII konvertálás
(ASCII_konv)
RETURN

17. ábra


Megszakításkezelő
Soros port (USART) megszakítás?
i
n
TIMER1 indítása
Soros port (SSPI) megszakítás?
i
n
I2C megszakításkezelő (I2CMIntISR)
Vétel indul (ReceiveChar)
TIMER0 (1,25S) megszakítás?
i
n
_HM_meres_ST=1
_Rcv_ok=1
Kilépés a megszakításból

18. ábra

Beavatkozó egység



19. ábra: beavatkozó egység

Felépítése

A beavatkozó egység (19. ábra) feladata a központi vezérlő egységtől érkező parancsok végrehajtása, hibaüzenet küldése áramszünet és kazánhiba (ha nem fűt) esetén.

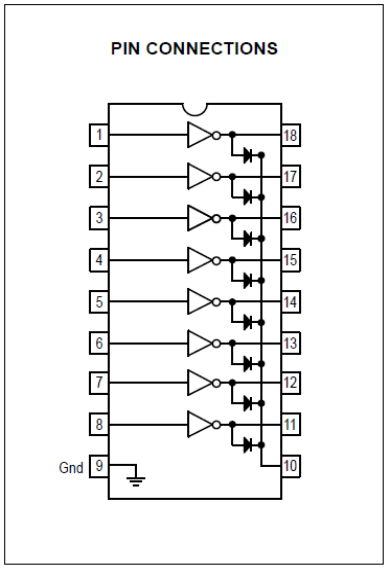
A kommunikációt mindig a központi egység kezdeményezi. A beavatkozó egység köteles válaszolni. A válaszüzenet tartalmazza az esetleges hibaüzeneteket. Biztonsági okokból a modult a központi egységnek rendszeresen meg kell szólítania. Ugyanis ha lefagy a vezérlő egység és a kazán éppen fűt, akkor a beavatkozó egység öt perc leteltével tartalék üzemmódra kapcsol. A tartalék üzemmód termosztátos padlófűtést biztosít a hiba elhárításáig. A beavatkozó egység

analóg, teljesítményvezérlő kimenetével sorosan egy hagyományos termosztát van kapcsolva. Ezt a rendszerben beállított értéknél egy kicsivel magasabb hőfokra kell állítani. Tartalék üzemben a beavatkozó egység analóg kimenete maximális értékre áll, és a padlófűtés keringető szivattyúja teljes sebességre kapcsol. Ezzel átadja a vezérlést a termosztátnak.

Az egységben alkalmazott PIC18F2520 mikrovezérlő az érzékelő egységben használt PIC18F2420 kétszer akkora memóriával ellátott változata. Bár a feladatot a PIC18F2420 is el tudná látni, azért döntöttem a PIC18F2520 alkalmazása mellett, mert ennek DIP tokozású változatából korábban már vásároltam néhány darabot. A hőmérő modul nyomtatott áramkörének mérete nem teszi lehetővé a DIP tokozású mikrovezérlő használatát, de kényelmesen elfér a beavatkozó egységben.

A PIC a központi vezérlő egységtől érkező parancsoknak megfelelően kapcsolja a padlófűtés keringető szivattyúját és a radiátorok elektromos szelepeit. A mikrovezérlő kis terhelhetőségű TTL szintű logikai kimenetei nem képesek közvetlenül meghajtani az erősáramú kimeneti egység reléit. Ezért van szükség az ULN2803AN integrált áramkörre, mely nyolc darab meghajtó fokozatot tartalmaz a 12V-os relék vezérléséhez. Az IC blokkdiagramja a 20. ábrán látható.

Az ULN2803AN integrált áramkör adatlapja a következő helyről tölthető le: http://pdf1.alldatasheet.com/datasheet-pdf/view/12687/ONSEMI/ULN2803.html



20. ábra: ULN2803AN IC

A kazánunk teljesítménye folyamatosan szabályozható 10,3-28kW között. Ehhez 10-16V közötti egyenfeszültségre van szüksége. Ezt a vezérlőfeszültséget a mikrovezérlő digitális jeléből a MAX5820L típusú kettős nyolc bites D/A konverter (DAC) egyik fele analóg feszültséggé alakítja a 0-5V tartományban (adatlap: http://datasheets.maxim-ic.com/en/ds/MAX5820.pdf). A TL072 kettős műveleti erősítővel (adatlap: http://focus.ti.com/lit/ds/symlink/tl071.pdf) épített szintillesztő fokozat gondoskodik arról, hogy a vezérlőfeszültség a kazán számára megfelelő 10-16V-os tartományba essen.

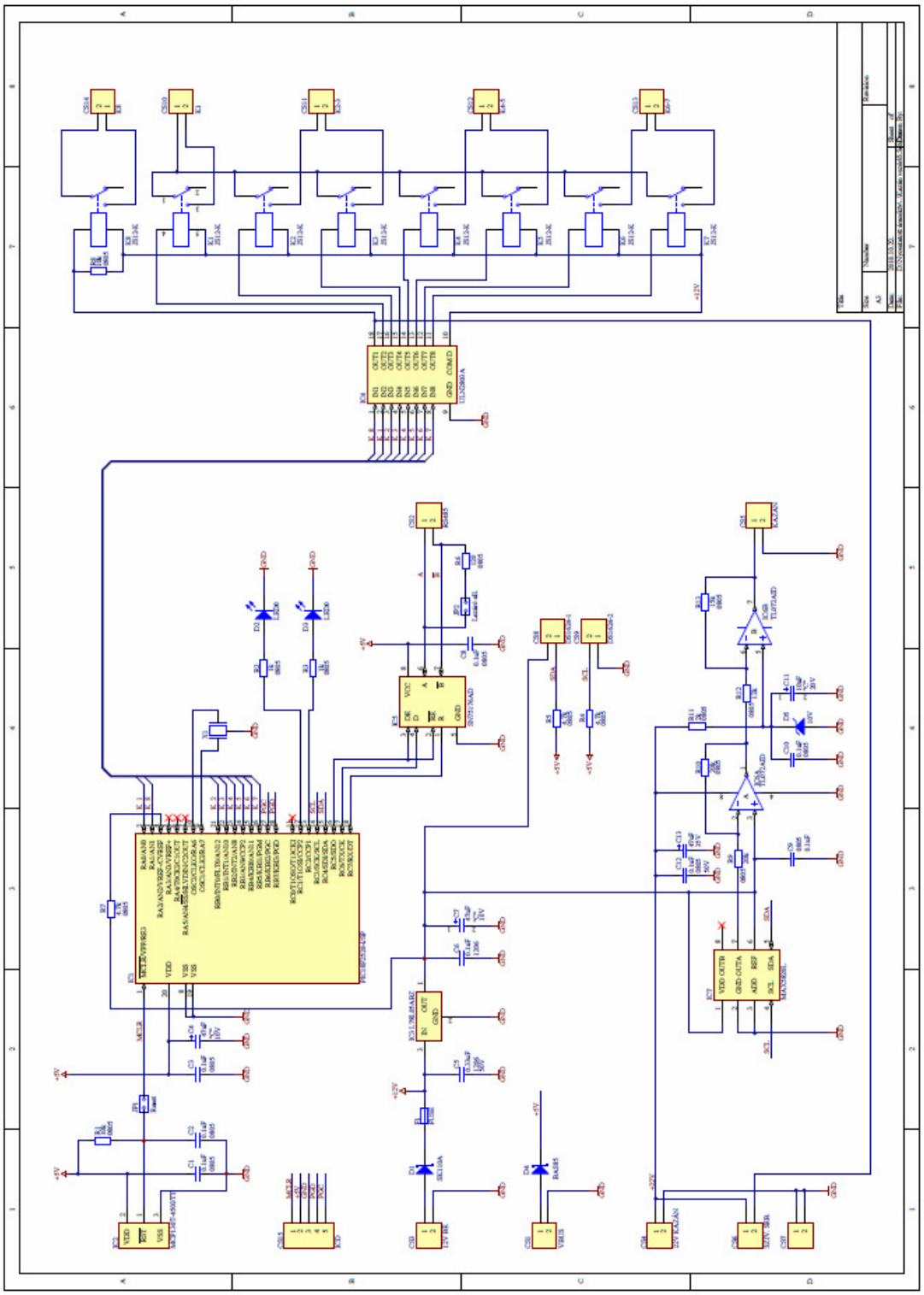
A korábban már ismertetett DS1624 méri a kazánból a fűtőtestek felé áramló víz hőmérsékletét. Ezzel a visszacsatolással tudja a mikrovezérlő ellenőrizni, hogy fűt-e a kazán.

A PIC figyeli a 12V-os tápfeszültség meglétét, áramszünet esetén megszűnik az egység 12V-os tápellátása. Ekkor csak a mikrovezérlő és az RS-485 interfész működik, hiszen tápellátásukat a központi vezérlő egységtől kapják. A modul a következő üzenetváltáskor hibaüzenetet küld a központi egységnek.

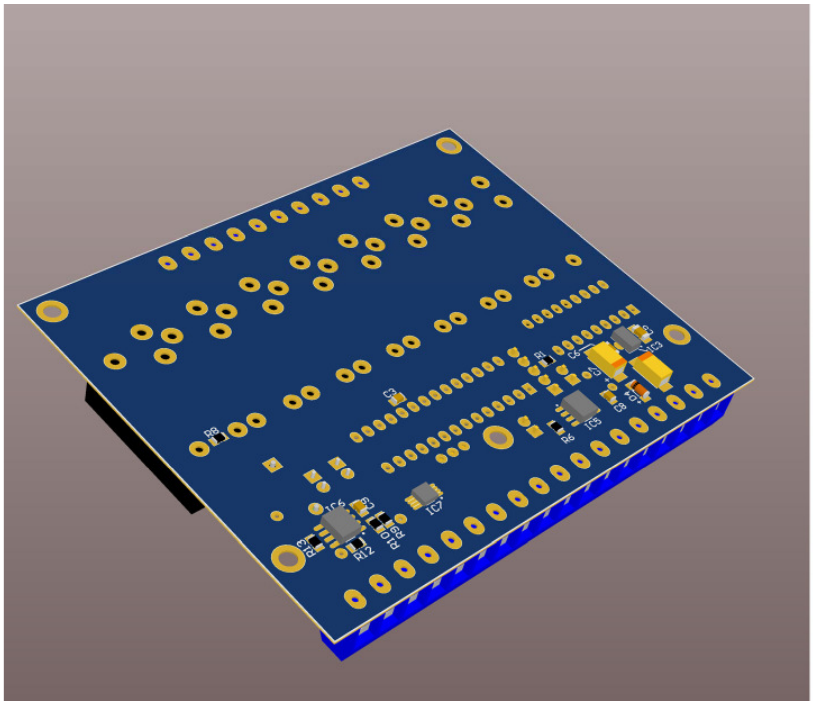
Az RS-485 interfészt itt is az SN75176 IC alkotja.

A beavatkozó egység kapcsolási rajza a 21. ábrán, beültetési rajza a 22. és 23. ábrán látható.

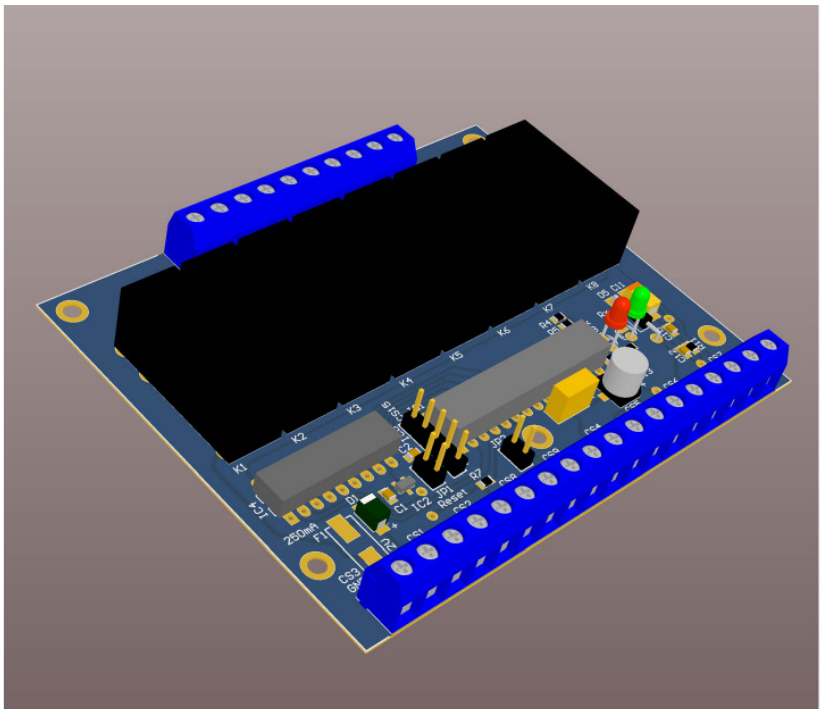
Az elkészült modul képei a 24.-27. ábrán láthatóak.



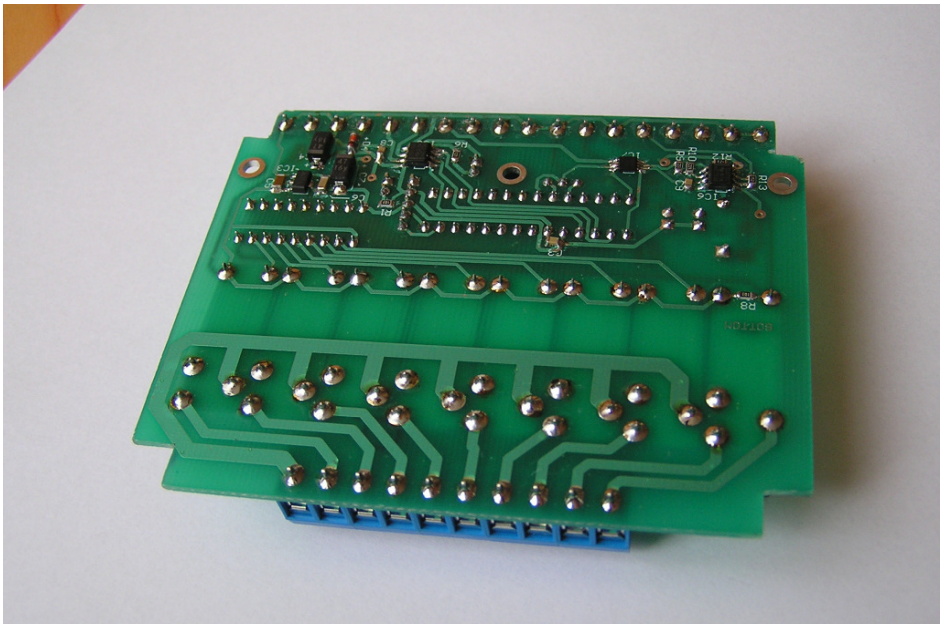
21. ábra: a beavatkozó egység kapcsolási rajza



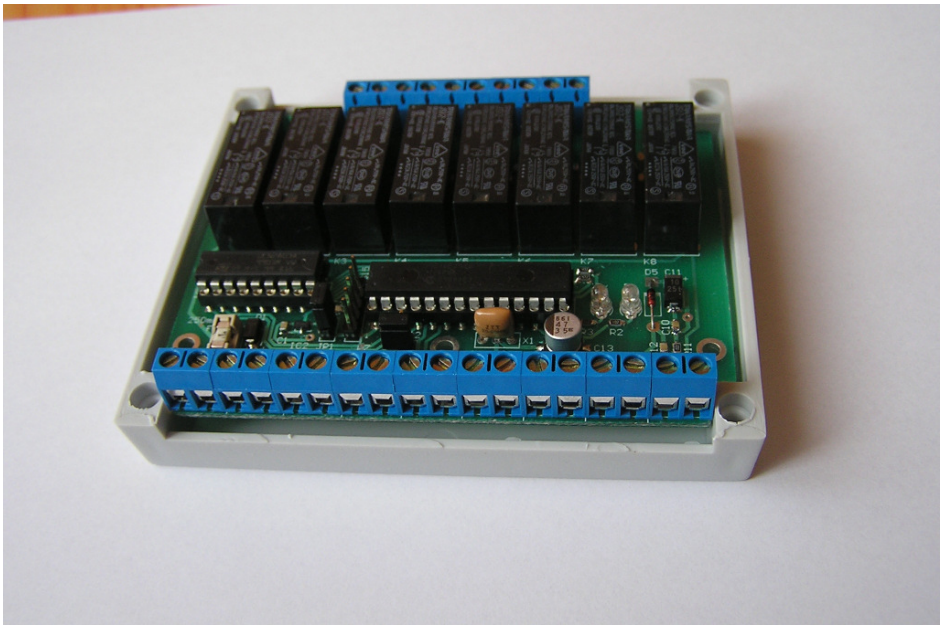
22. ábra: beültetési rajz (Bottom)



23. ábra: beültetési rajz (Top)



24. ábra



25. ábra

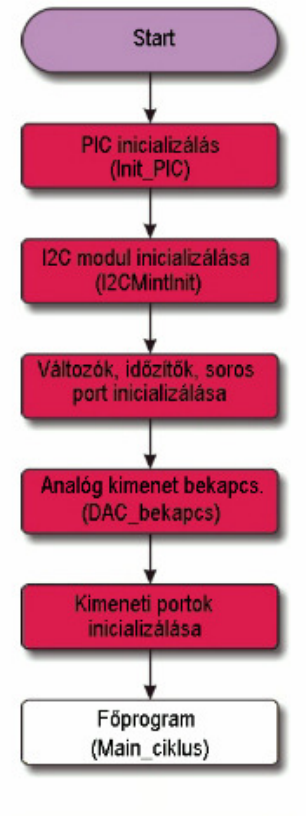
26. ábra

A program

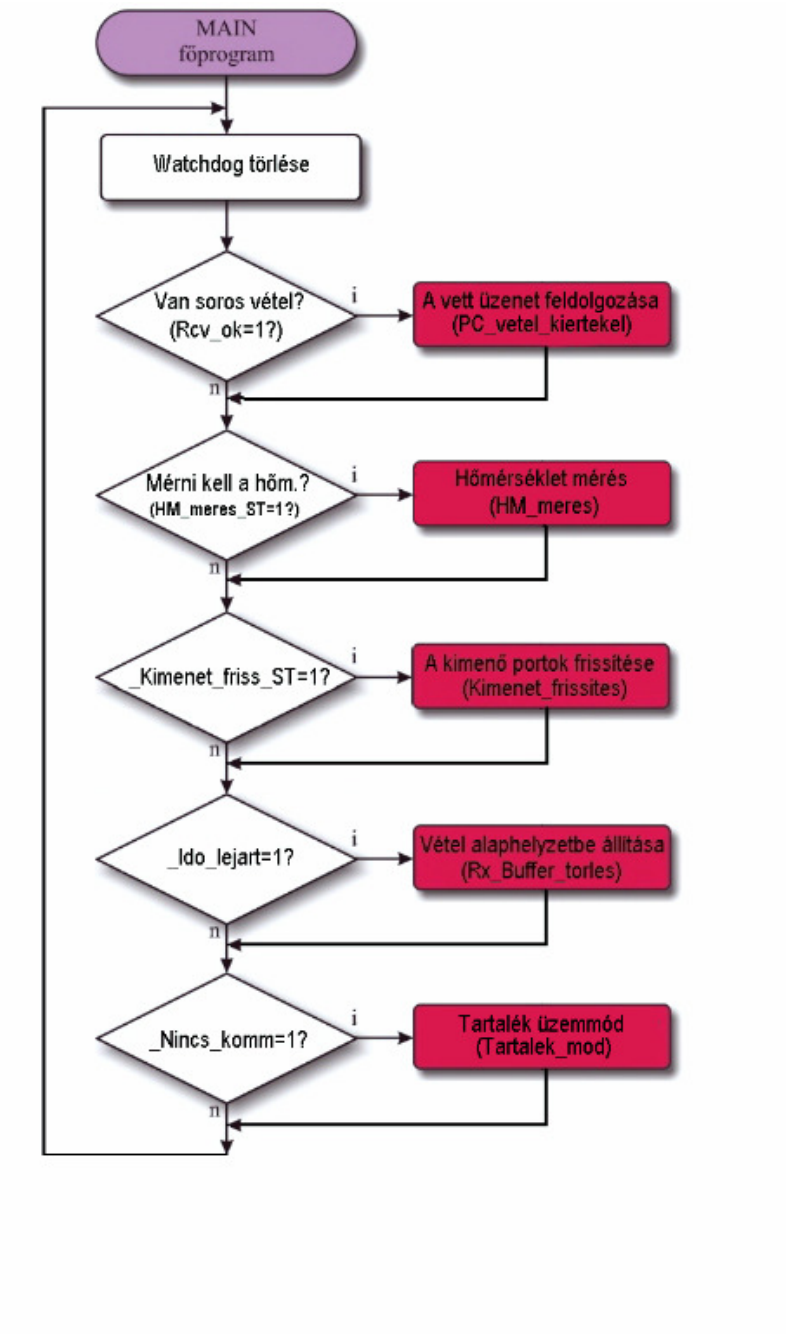
A beavatkozó egység programfelépítése hasonló az előzőekben bemutatotthoz. Az időzítés alapja itt is a Timer0, de 1,25s-os időzítéssel. A kazán előremenő hőmérsékletét 10 másodpercenként mérjük 1C-os felbontással. A rosszabb felbontás miatt az átlagszámítás elhagyható. Mivel ez a modul két I2C egységet is vezérel és a DS1624 konverziós ideje 1s körül van, ezért elegendő időt kell biztosítani a hőmérsékletmérésre, nehogy ütközés lépjen fel az I2C vonalon. Itt a központi egységtől érkező parancs a címen és ellenőrző számon kívül két utasítást is tartalmaz. Az első a kazán hőmérsékletének értékét, a második a nyolc relés kimenet beállítási módját (radiátor szelepek és keringető szivattyú) adja meg. Ezeket az utasításokat az egység visszaküldi a PC-nek, ami ezzel egyfajta hibaellenőrzést végez a modulon. Ezért nem hagyható el sem vett adatok ASCII-HEX konvertálása, sem pedig az ellenőrző szám vizsgálata. A vett utasításokon kívül elküldi a mért hőmérséklet értékét , és két hibaüzenetet: ha nem fűt a kazán (ha a hőmérséklet a fűtés bekapcsolása után 80s-al sem éri el a 40C-ot), illetve, ha áramszünet van. Ezen üzenetek értelmezése a központi egység feladata. Az összeállított válaszüzenet ellenőrző szám számítás és HEX-ASCII átalakítás után kerül az adási pufferbe. Válasz esetén ennek tartalma kerül a soros portra.

Mindkét típusú modul két LED-et tartalmaz, egy zöldet és egy pirosat. A zöld LED a Timer0 megszakításait , a piros pedig a soros port vételét jelzi. Ezek segítségével egyszerű szemrevételezéssel ellnőrizhető a modul alapvető működése.

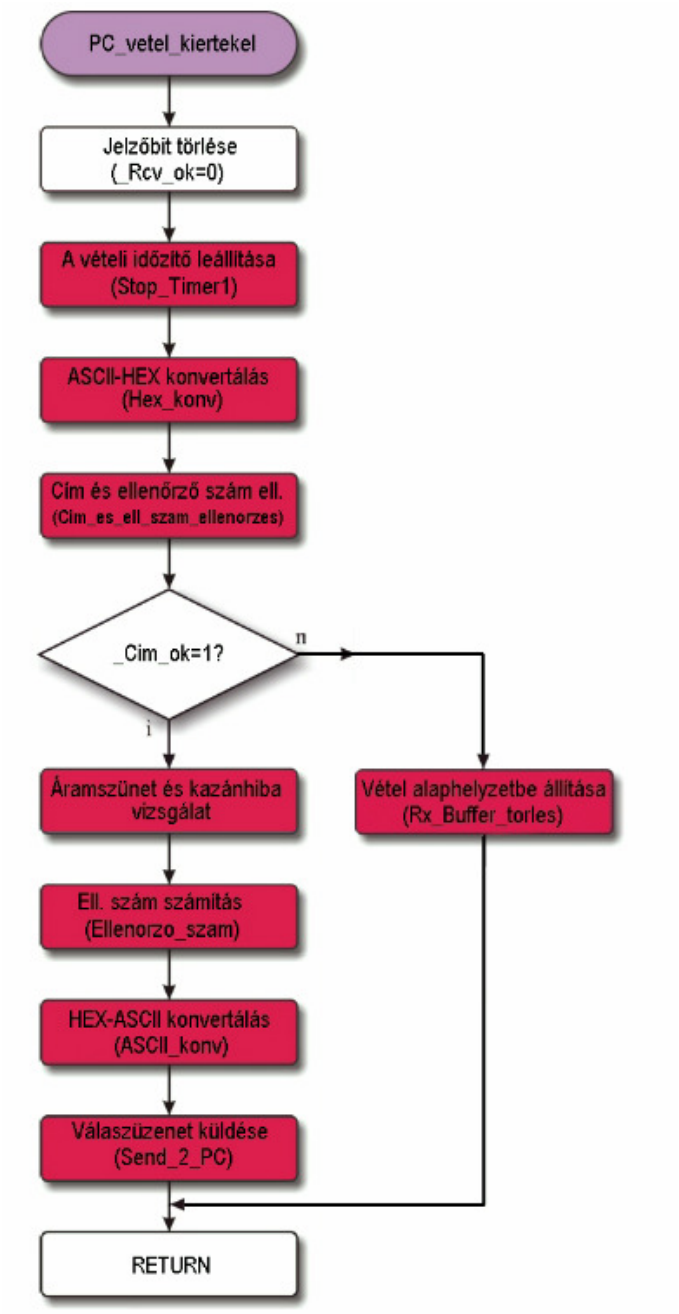
A program folyamatábrája 27.-31. ábrákon látható.



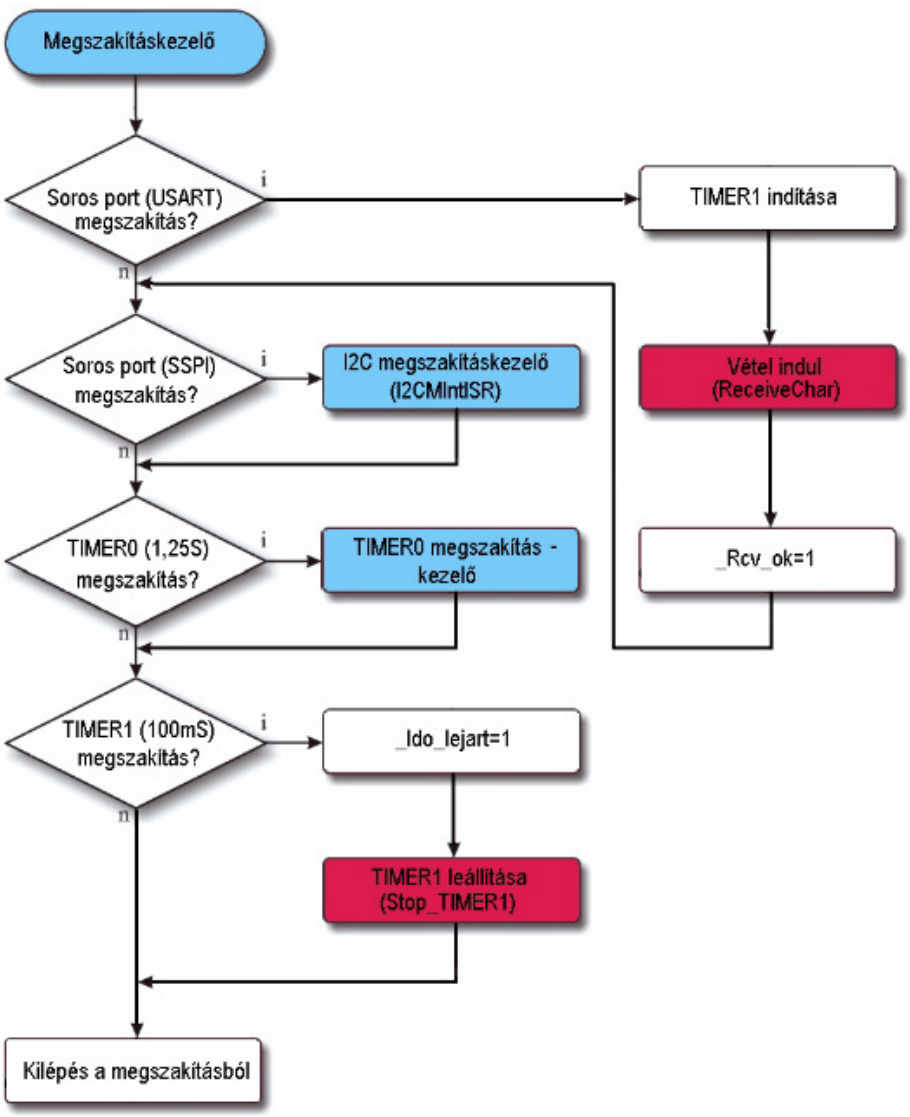
27. ábra


MAIN
főprogram
Watchdog törlése
Van soros vétel?
(Rcv_ok=1?)
i
A vett üzenet feldolgozása
(PC_vetel_kiertekel)
n
Mérni kell a hőm.?
(HM_meres_ST=1?)
i
Hőmérséklet mérés
(HM_meres)
n
_Kimenet_friss_ST=1?
i
A kimenő portok frissítése
(Kimenet_frissites)
n
_Ido_lejart=1?
i
Vétel alaphelyzetbe állítása
(Rx_Buffer_torles)
n
_Nincs_komm=1?
i
Tartalék üzemmód
(Tartalek_mod)
n

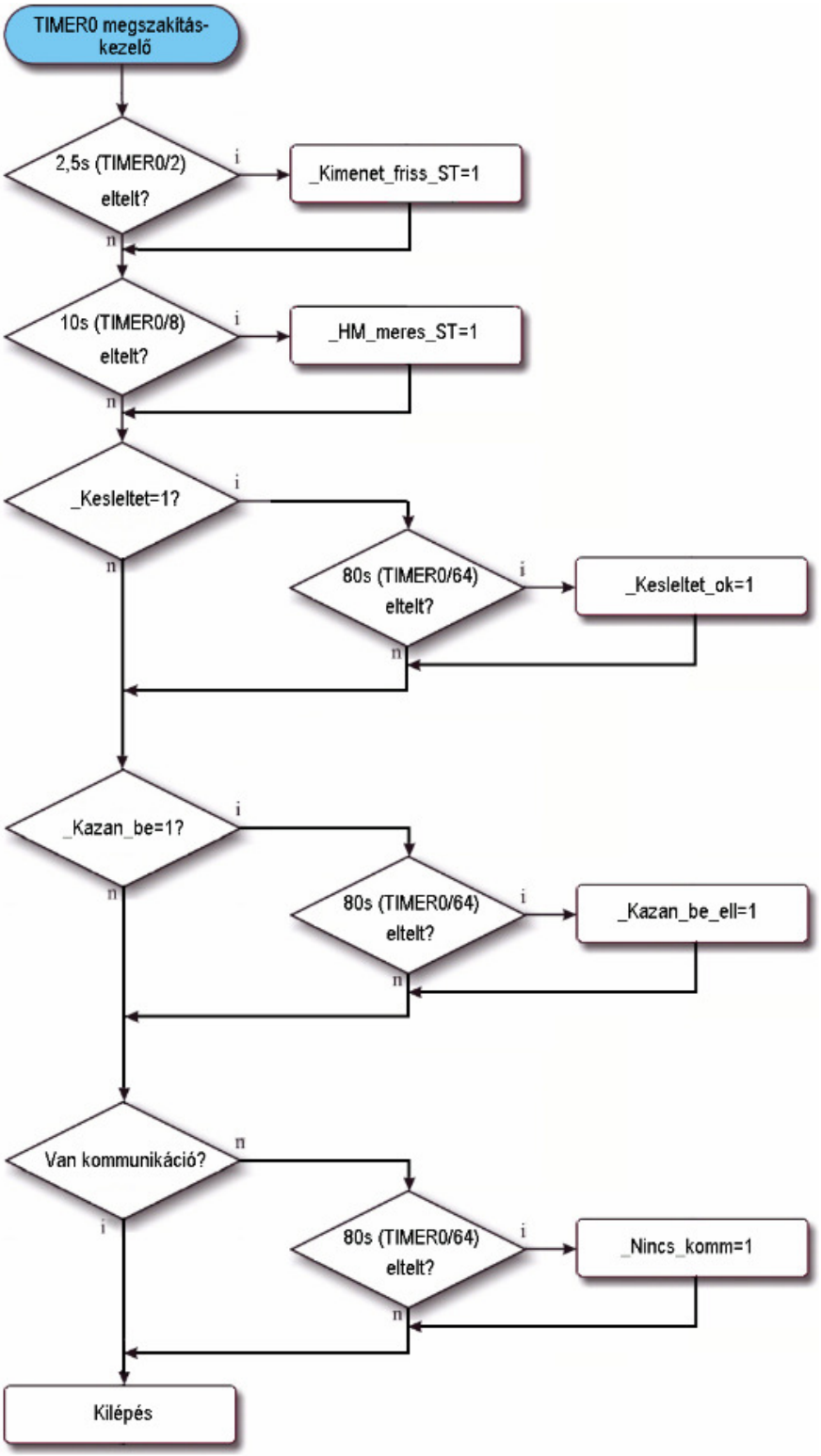
28. ábra


PC_vetel_kiertekel
Jelzőbit törlése
(_Rcv_ok=0)
A vételi időzítő leállítása
(Stop_Timer1)
ASCII-HEX konvertálás
(Hex_konv)
Cím és ellenőrző szám ell.
(Cim_es_ell_szam_ellenorzes)
_Cim_ok=1?
n
i
Áramszünet és kazánhiba
vizsgálat
Vétel alaphelyzetbe állítása
(Rx_Buffer_torles)
Ell. szám számítás
(Ellenorzo_szam)
HEX-ASCII konvertálás
(ASCII_konv)
Válaszüzenet küldése
(Send_2_PC)
RETURN

29. ábra.


Megszakításkezelő
Soros port (USART) megszakítás?
i
n
TIMER1 indítása
Vétel indul (ReceiveChar)
_Rcv_ok=1
Soros port (SSPI) megszakítás?
i
n
I2C megszakításkezelő (I2CMIntISR)
TIMER0 (1,25S) megszakítás?
i
n
TIMER0 megszakítás - kezelő
TIMER1 (100mS) megszakítás?
i
n
_Ido_lejart=1
TIMER1 leállítása (Stop_TIMER1)
Kilépés a megszakításból

30. ábra


TIMER0 megszakítás-kezelő
2,5s (TIMER0/2) eltelt?
i
_Kimenet_friss_ST=1
n
10s (TIMER0/8) eltelt?
i
_HM_meres_ST=1
n
_Kesleltet=1?
i
n
80s (TIMER0/64) eltelt?
i
_Kesleltet_ok=1
n
_Kazan_be=1?
i
n
80s (TIMER0/64) eltelt?
i
_Kazan_be_ell=1
n
Van kommunikáció?
n
i
80s (TIMER0/64) eltelt?
i
_Nincs_komm=1
n
Kilépés

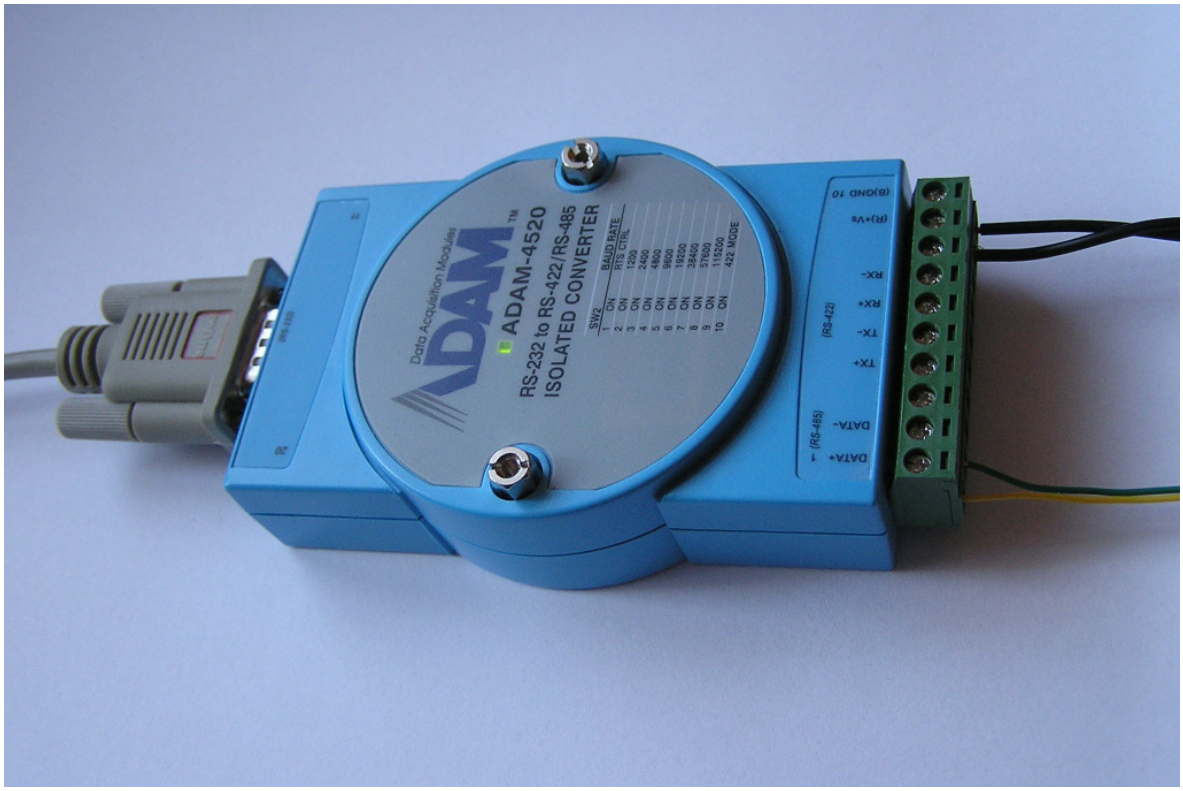
31. ábra

A központi vezérlő és adatmegjelenítő egység

Felépítése

A központi egységet egy szünetmentes tápegységről üzemelő PC és egy RS-232/RS-485 átalakító alkotja. A PC egy kis fogyasztású Intel Atom alaplapra épül, így a teljes teljesítményfelvétele mindössze 44W.

Az RS-232/RS-485 átalakítást egy ezen dolgozat megírásához kölcsönkapott ADAM-4520 típusú ipari kivitelű konverter végzi. Képe a 32. ábrán látható.



32. ábra: ADAM-4520 RS-232/RS-485 átalakító

A program

A vezérlő egység programját Microsoft VB6 fejlesztő környezetben írtam. Két okból választottam ezt. Egyrészt korábban már végeztem néhány kísérletet a fűtési rendszer automatizálására és ez a kísérleti program képezi az alapját a vezérlő egység programjának. Másrészt ebből egy jogtiszta példánnyal is rendelkezem.

A központi vezérlő és adatmegjelenítő egység egyik feladata a modulokkal való kommunikáció. A rendszerben alkalmazott Master-Slave protokollban a vezérlő egység (PC) a Master.

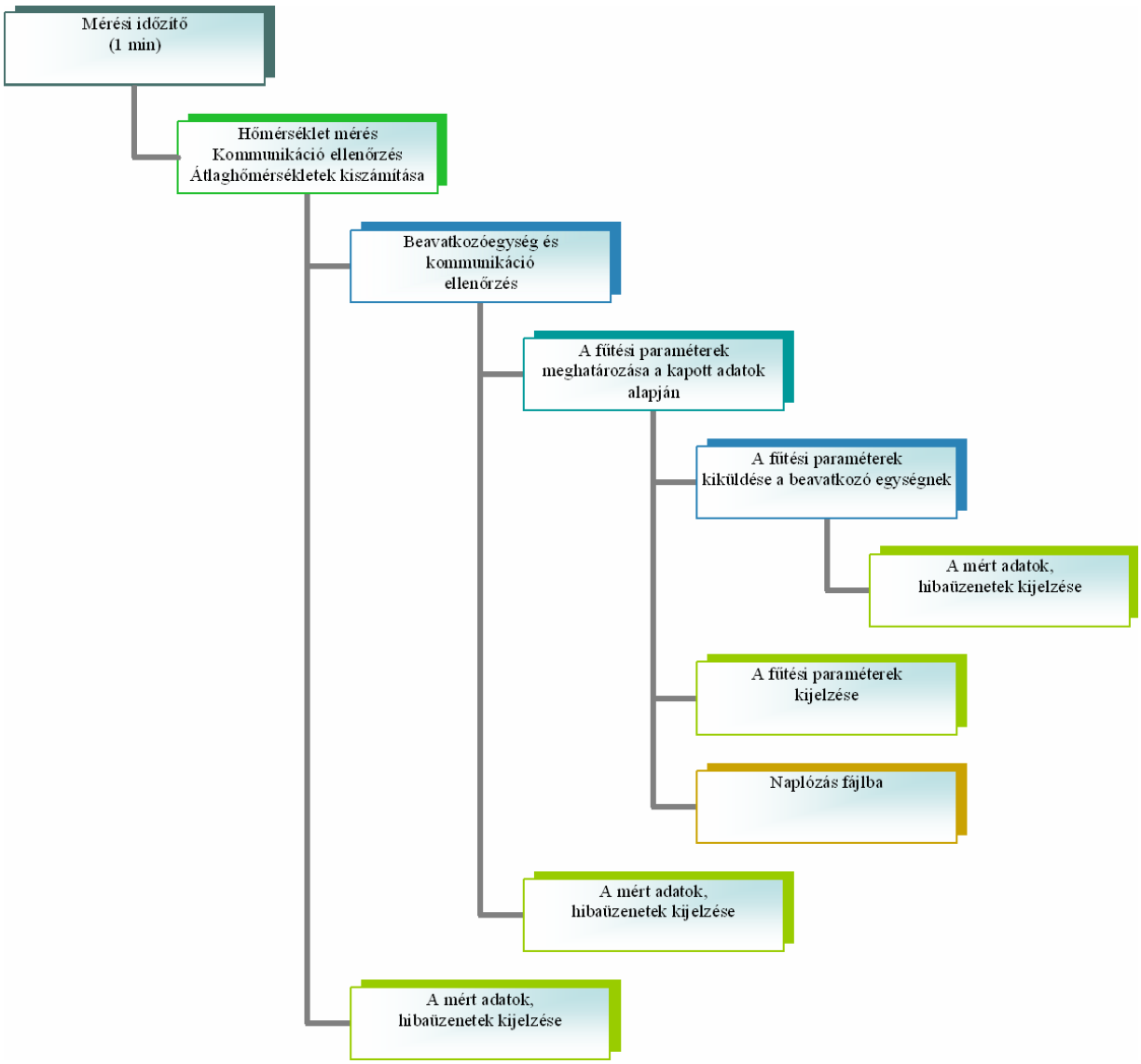
Mivel a PC kezdeményezi a kommunikációt és a megszólított modul is neki válaszol, így a központi egységnek nem szükséges saját címmel rendelkeznie. A PC az általa küldött parancsra érkező válaszon cím és ellenőrző szám ellenőrzést hajt végre (az válaszolt, akit kérdezett, nem sérült az üzenet?) Ha nem érkezik válasz, akkor újraküldi a parancsot. Három próbálkozás után hiba üzenetet küld a felhasználónak, de percenként egyszer újra megpróbálja.

Az érzékelő egységek esetében a beérkezett adatokat decimálissá konvertálja, átlagot számít belőlük és megjeleníti őket. A beavatkozó egység esetében azt is ellenőrzi, hogy visszakapta-e az általa küldött üzenetet. Ezzel egyfajta ellenőrzést végez, hiszen ezt az üzenetet a modulnak fel kell dolgoznia, majd a válaszüzenetben vissza kell küldenie. Ha a válasz a formai követelmények (kezdő és záró karakterek ASCII karakterek 0-9 illetve A-F tartományban és érvényes ellenőrző szám) teljesülése esetén nem egyezik a kiküldött paranccsal, akkor azt a modul hibásan dolgozta fel. Ekkor leáll minden kommunikáció és a beavatkozó egység hibájáról üzenet tájékoztatja a felhasználót. Ilyenkor felhasználói beavatkozás szükséges. Ha minden rendben van akkor a beavatkozó egységből érkező adatokat megjeleníti.

Ha a rendszer úgy működne, hogy a vezérelt berendezések (szivattyú, szelepek) mindegyike visszajelzést küldene a modulnak a saját állapotáról és a modul ezeket az állapotokat küldené vissza a PC-nek, akkor ez a módszer igen magas szintű megbízhatóságot eredményezne, mely nemcsak a modul, hanem a berendezések működését is ellenőrizné. Ilyen ipari szintű megbízhatóságra (és költségre) szerintem nincs szükség ebben az esetben, hiszen a kazán önmagában is rendelkezik olyan védelmi rendszerrel, mely hiba esetén (pl. túlmelegedés) leállítja a fűtést.

A mérési eredmények alapján meghatározza a fűtési paramétereket, naplózza és elküldi a beavatkozó egységnek. A naplófájl formátuma Excell kompatibilis.

A központi vezérlő egység funkció diagramja a 33. ábrán látható.



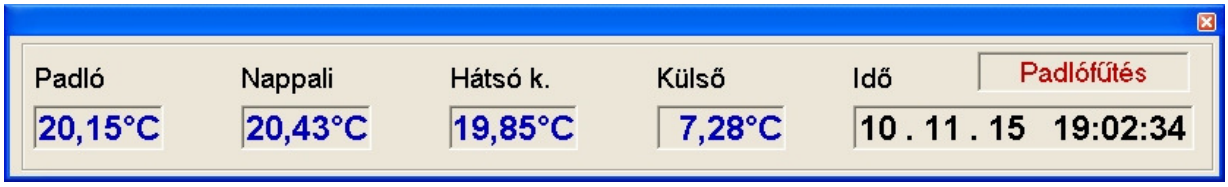
33. ábra: a központi vezérlő egység funkció diagramja

A programablak megnyitás után a képernyő alján a 34. ábrán láthatókompakt formában jelenik meg. Itt a legfontosabb paraméterek láthatóak úgy, hogy ne takarják a mögötte futó TV

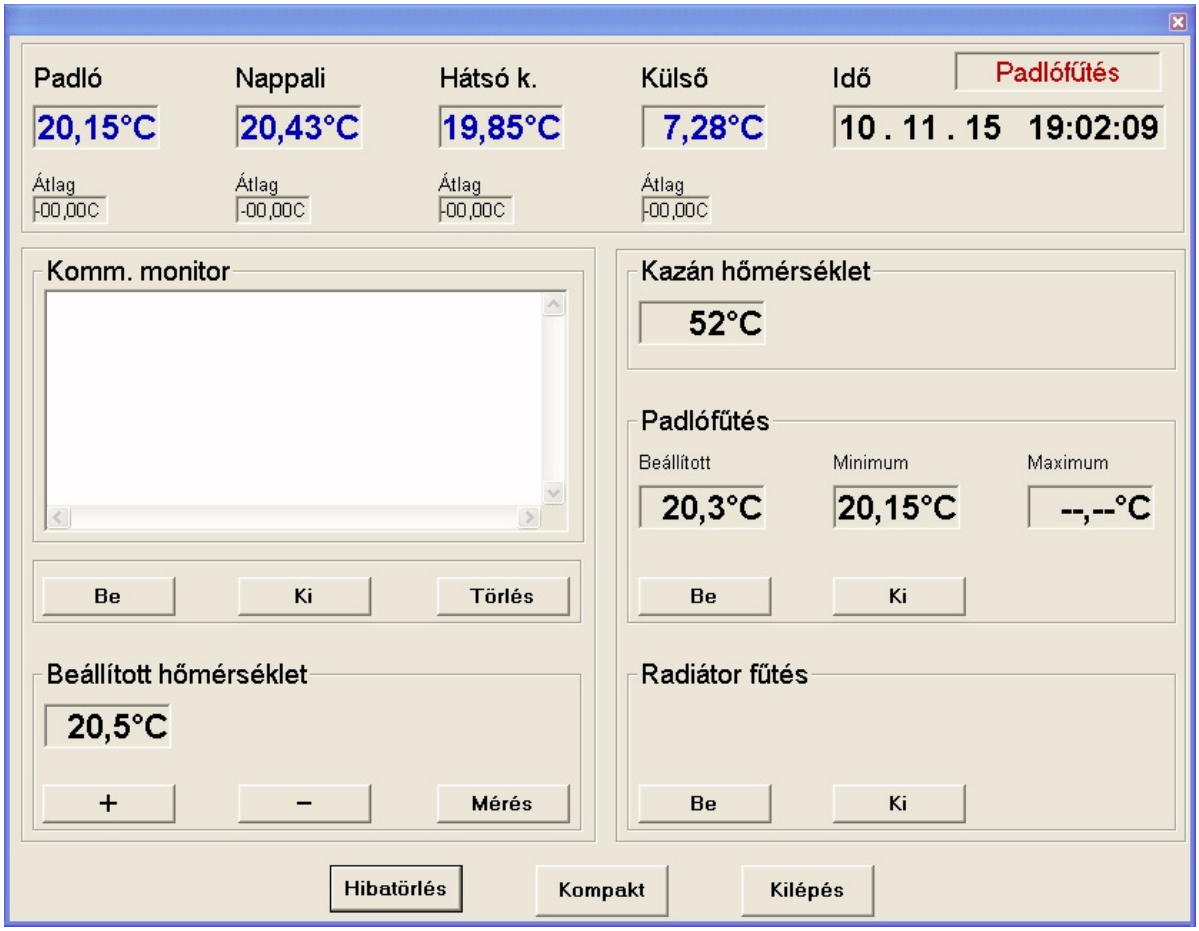
műsor képét. Ha duplán kattintunk erre az ablakra, akkor a 35. ábrán látható ablakban jelennek meg a rendszer ellenőrzéséhez és teszteléséhez szükséges vezérlőelemek.

Az egyes hőmérsékletkijelzők alatt látható a hozzájuk tartozó egy órás átlag, a kazán hőmérséklete, a padlófűtés adatai és a beállított hőmérséklet. Ez utóbbi értékét is itt lehet beállítani. A beállított hőmérséklet értékét a program a save_preset.txt fájlba menti Mindkét fűtési mód be-és kikapcsolható. A kommunikációs monitorral pedig a soros kommunikációt lehet ellenőrizni. A kompakt gomb szolgál az előző ablakméret visszaállítására. A hibatörlés gombbal minden hibaüzenet törlődik és újraindul a kommunikáció is.

A program saját felhasználásra készült, így nem készültek hozzá telepítő fájlok felhasználói leírások.



34. ábra: kompakt nézet



35. ábra: teljes nézet

Összegzés, és a továbbfejlesztés lehetőségei

Mikor nyilvánvalóvá vált, hogy házunk komfortos és energiatakarékos fűtése nem valósítható meg a kereskedelemben kapható hagyományos termosztátokkal végeztem néhány kísérletet, hogy kiderítsem számítógépes vezérléssel meg tudom-e oldani a problémát.

A kísérletek eredményeként kb. 15%-al lehetett az éves gázfogyasztást csökkenteni ugyanakkora beállított hőmérséklet és nagyobb komfort mellett. Ez elég jelentős az éves 3300m³-es gázfogyasztás mellett. Ennek egyik oka, hogy a nagyobb felbontású hőmérőknek köszönhetően jelentősen csökkenthető a túlfűtés. Az általam elért hőmérsékletingadozás (ΔT) 0,5°C volt 24 óra alatt.

Mint fentebb már idéztem: Magyarországon 1°C túlfűtés (1°C ΔT) 6% energia többlettel jár. [11]

Erre pedig még egy 0,5°C felbontású digitális termoszát sem képes. Ha a beállított hőmérséklet pl.: 20°C, akkor a termosztát 19,5°C-nál kapcsolja be a fűtést, és 20,5°C-nál pedig ki. Ez már önmagában is 1°C hőmérsékletingadozás, de ehhez még hozzájön az adott fűtési rendszer tehetetlensége is. A fűtés bekapcsolásakor csak egy bizonyos idő elteltével kezd emelkedni a hőmérséklet, előtte még tovább hűl. Kikapcsoláskor fordított a helyzet. Ennek mértékét az adott fűtési rendszer típusa és kiépítésének módja (hőszigetelés, fűtőtestek méretezése, stb.) és a külső hőmérséklet határozza meg. Tehát egy ilyen termosztáttal valójában a ΔT könnyen elérheti a 2°C-ot is. A nagy hőmérsékletingadozás nem csak kényelmetlen, de még sokba is kerül.

A másik ok az, hogy a számítógépes rendszerrel megoldható, hogy a kevésbé gazdaságos radiátoros fűtés még alacsony külső hőmérséklet esetén is csak kiegészítő fűtés maradjon.

Ezen tapasztalatok alapján döntöttem úgy, hogy érdemes egy ilyen rendszert építenem.

Mivel ezelőtt még nem foglalkoztam sem PIC programozással, sem az assembly nyelvvel, így kihívást jelentett a PIC programozás alapjainak elsajátítása, a magas szintű programnyelvektől eltérő filozófiája. A beavatkozó egység programjának írásánál problémát jelentett, hogy az i2C kommunikációs portra két eszköz is csatlakozik és ezek valamelyike néha látszólag véletlenszerűen folyamatosan foglalta az I2C Buszt. A hibát csak az eszköz ki-és bekapcsolásával lehetett megszüntetni. Sikerült kideríteni, hogy mivel a DS1624 konverziós ideje

1s körül van, a PIC által küldött olvasási parancs nyugtázása után ugyan nem foglalja tovább a Buszt, de a konverzió befejeztével (valamikor 1s-on belül) elkezdi küldeni az adatokat a PIC-nek. az I2C Busz foglaltságát ellenőrző rutin már a nyugtázás után szabadnak nyilvánítja a Buszt. Ha a PIC és a DAC kommunikációja nem fejeződik be a DS1624 konverziós idején belül, akkor ütközés lép fel a Buszon. Ezt úgy sikerült megoldanom, hogy mindkét eszköz kommunikációját a Timer0 vezérli. A DAC-al minden páratlan megszakítás esetén (2,5s-onként), a hőmérővel pedig páros megszakítás esetén 10s-onként kommunikálhat a PIC. A modulok áramköreinek megtervezése után a paneleket megterveztettem és legyártattam, majd összeszereltem. Ezek a feladatok ugyan nem kapcsolódnak szervesen ezen dolgozat témájához, de ezeket is el kellett végeznem, hogy a rendszer működőképességét ellenőrizni tudjam. A próbapaneles tesztelés ugyanis még nem jelenti azt, hogy a megépített rendszer „élesben" is megfelelően működik.

A központi vezérlő egység programjának azon része, amelyik a fűtési paramétereket határozza meg gyakorlatilag ugyanaz, mint amelyik egyszer már bevált a kísérleti rendszer esetében. Új fejlesztés a két PIC vezérlésű modul, azok programjai, a központi egység programjának kommunikációs modulja és kezelői felülete. Az új fejlesztésű eszközök és programok megfelelően működnek.

Sikerült a dolgozatban kitűzött célokat megvalósítanom:

- olyan automatikus fűtési rendszert építeni, amelyik gazdaságosan vezérli a padlófűtést és a radiátoros fűtést.
- a megépített rendszer bővíthető legyen. A ház elektromos rendszerének vezérlése (a megfelelő érzékelő és beavatkozó egységek megépítése után) a már meglévő rendszerbe integrálható.

A fejlesztés egyik következő lépéseként ki lehetne hagyni a hagyományos termosztátot a rendszerből és tartalék üzemmódban a beavatkozó egység kérne adatot a nappaliban elhelyezett érzékelő egységtől. A hiba elhárításáig egy egyszerűsített, de a hagyományos termosztátos vezérlésnél hatékonyabb szabályzást tenne lehetővé.

A rendszert további érzékelőkkel bővítve akár minden egyes szobában külön lehetne vezérelni a radiátoros fűtést. A beavatkozó egység tervezésekor figyelembe vettem ezt a lehetőséget és még 4 radiátorszelep vezérlésére van lehetőség.

Külön lehetne választani a központi vezérlő és az adatmegjelenítő funkciókat. A központi vezérlő szerepét egy kis fogyasztású PIC-el megépített modul vehetné át, a PC feladata mindössze az adatok képernyőn való megjelenítése lenne. A központi vezérlő éjszakára lekapcsolná a PC-t. Napközben csak akkor kapcsolná be, ha otthon tartózkodik valaki. Így jelentősen csökkenteni lehetne a rendszer energiafogyasztását.

Irodalomjegyzék

[1] http://www.intelligensotthon.hu/

[2] http://www.domintell.hu/en/news.html

[3] http://www.elektro-kamleithner.hu/download/lakaskultura.pdf

[4] Juhász Róbert: Mikrovezérlők oktatása Tananyag
http://plc.mechatronika.hu/piclei/mikrovez_okt_honlap.pdf

[5] MICROCHIP: MPLAB IDE Quick Start Guide
http://ww1.microchip.com/downloads/en/DeviceDoc/51281d.pdf

[6] MICROCHIP: PIC18F2420/2520/4420/4520 Data Sheet
http://ww1.microchip.com/downloads/en/DeviceDoc/39631e.pdf

[7] http://www.chipcad.hu/letoltes/PICkit_2_programozo_csomagok.pdf
http://www2.chipcad.hu/tartalom.aspx?hir=360

[8] http://www.modtronix.com/images/pickit3debugexpress.jpg

[9].Dr. Kónya László: PIC mikrovezérlők alkalmazástechnikája ChipCad Elektronikai Disztribúció Kft. 2000

[10] Közepes teljesítményű PIC mikrovezérlőkFelhasználói Kézikönyv
http://www.t-es-t.hu/elokep/pic.htm

[11] http://www.beryl.hu/

[12] RS-232, RS-422, RS-485 Kisokos –útmutató az ipari adatkommunikációhoz
www.s-e.hu

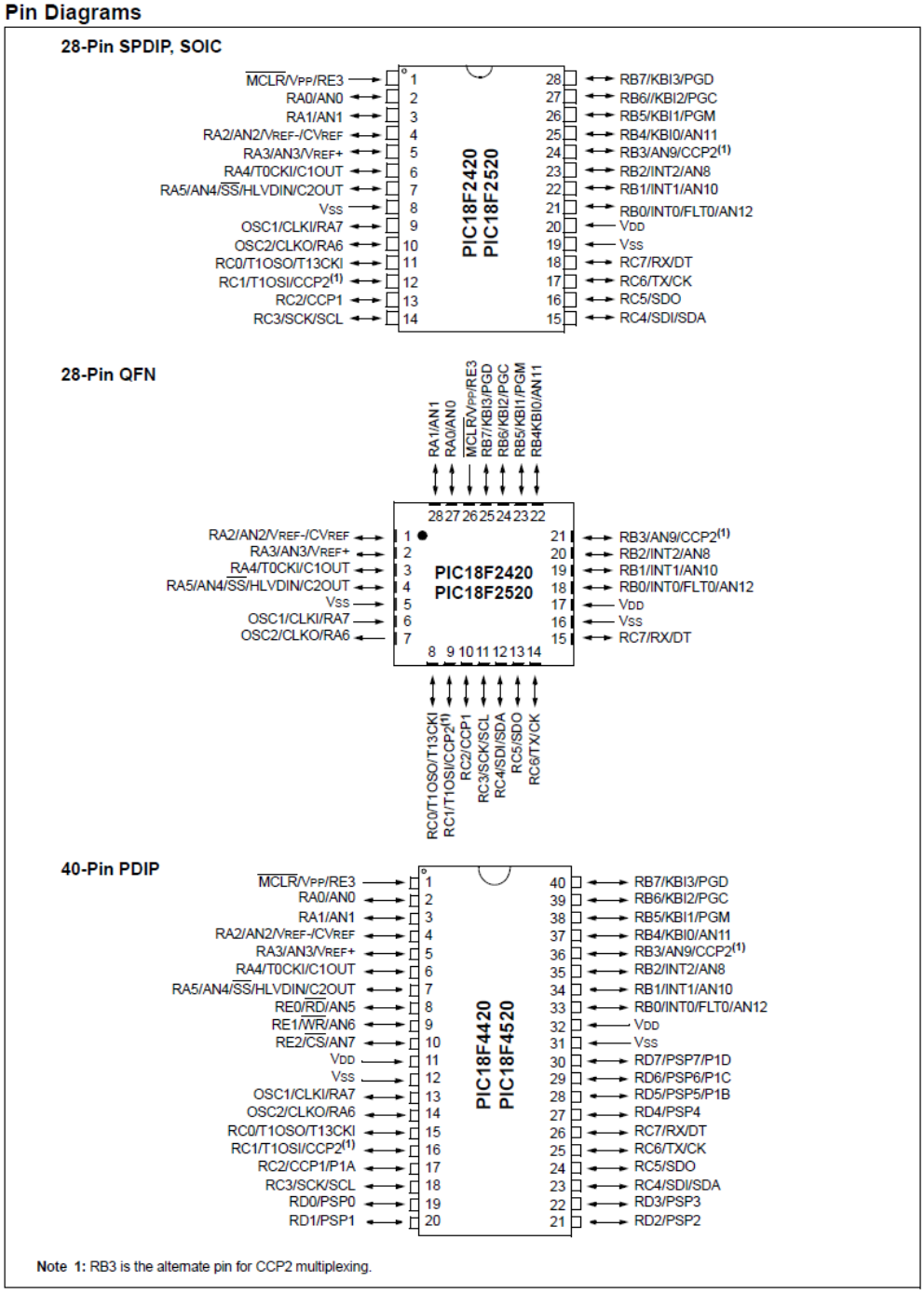
[13] http://datasheets.maxim-ic.com/en/ds/DS1624.pdf

[14] Kommunikáció alapjai - Soros adatátvitel
http://www.hobbielektronika.hu/cikkek/kommunikacio_alapjai_-_soros_adatatvitel.html?pg=5

Függelék

A PIC18F2420/2520/4420/4520 lábkiosztása

Pin Diagrams



[6]

A PIC18F2420/2520 kivezetéseinek funkciói

TABLE 1-2: PIC18F2420/2520 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC	QFN			
MCLR/VPP/RE3	1	26			Master Clear (input) or programming voltage (input).
MCLR			I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP			P		Programming voltage input.
RE3			I	ST	Digital input.
OSC1/CLKI/RA7	9	6			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise.
CLKI			I	CMOS	External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7			I/O	TTL	General purpose I/O pin.
OSC2/CLKO/RA6	10	7			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6			I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
CMOS = CMOS compatible input or output
I = Input
P = Power

Note 1: Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2: Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.

TABLE 1-2: PIC18F2420/2520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC	QFN			
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	11	8			
RC0			I/O	ST	Digital I/O.
T1OSO			O	—	Timer1 oscillator output.
T13CKI			I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	12	9			
RC1			I/O	ST	Digital I/O.
T1OSI			I	Analog	Timer1 oscillator input.
CCP2(2)			I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
RC2/CCP1	13	10			
RC2			I/O	ST	Digital I/O.
CCP1			I/O	ST	Capture 1 input/Compare 1 output/PWM1 output.
RC3/SCK/SCL	14	11			
RC3			I/O	ST	Digital I/O.
SCK			I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL			I/O	ST	Synchronous serial clock input/output for I^2C™ mode.
RC4/SDI/SDA	15	12			
RC4			I/O	ST	Digital I/O.
SDI			I	ST	SPI data in.
SDA			I/O	ST	I^2C data I/O.
RC5/SDO	16	13			
RC5			I/O	ST	Digital I/O.
SDO			O	—	SPI data out.
RC6/TX/CK	17	14			
RC6			I/O	ST	Digital I/O.
TX			O	—	EUSART asynchronous transmit.
CK			I/O	ST	EUSART synchronous clock (see related RX/DT).
RC7/RX/DT	18	15			
RC7			I/O	ST	Digital I/O.
RX			I	ST	EUSART asynchronous receive.
DT			I/O	ST	EUSART synchronous data (see related TX/CK).
RE3	—	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VSS	8, 19	5, 16	P	—	Ground reference for logic and I/O pins.
VDD	20	17	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2: Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.

TABLE 1-2: PIC18F2420/2520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC	QFN			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0/FLT0/AN12	21	18			
RB0			I/O	TTL	Digital I/O.
INT0			I	ST	External interrupt 0.
FLT0			I	ST	PWM Fault input for CCP1.
AN12			I	Analog	Analog input 12.
RB1/INT1/AN10	22	19			
RB1			I/O	TTL	Digital I/O.
INT1			I	ST	External interrupt 1.
AN10			I	Analog	Analog input 10.
RB2/INT2/AN8	23	20			
RB2			I/O	TTL	Digital I/O.
INT2			I	ST	External interrupt 2.
AN8			I	Analog	Analog input 8.
RB3/AN9/CCP2	24	21			
RB3			I/O	TTL	Digital I/O.
AN9			I	Analog	Analog input 9.
CCP2(1)			I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
RB4/KBI0/AN11	25	22			
RB4			I/O	TTL	Digital I/O.
KBI0			I	TTL	Interrupt-on-change pin.
AN11			I	Analog	Analog input 11.
RB5/KBI1/PGM	26	23			
RB5			I/O	TTL	Digital I/O.
KBI1			I	TTL	Interrupt-on-change pin.
PGM			I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	27	24			
RB6			I/O	TTL	Digital I/O.
KBI2			I	TTL	Interrupt-on-change pin.
PGC			I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	28	25			
RB7			I/O	TTL	Digital I/O.
KBI3			I	TTL	Interrupt-on-change pin.
PGD			I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2: Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.

[6]

A PIC18FXXXsorozat utasításkészlete

Mező	Leírás
a	RAM hozzáférési bit a=0: a RAM terület az ún. „Access” RAM-ban van (a BSR-t nem kell figyelembe venni) a=1: a RAM bankot a BSR jelöli ki
bbb	Bit cím a 8 bites fájlregiszterben
d	Célkiválasztó bit: d=0(W): az eredmény a WREG-be kerül d=1(F): az eredmény a fájlregiszterbe (f) kerül
dest	Cél, akár a WREG, akár egy fájlregiszter
f	8 bites fájlregiszter neve (pl PORTB) v. címe(0xFF)
f_s	12 fájlregiszter címe (0x000-0xFFF). A forráscím tkp.
f_d	12 fájlregiszter címe (0x000-0xFFF). A célcím tkp.
k	Konstans mező, konstans adat, vagy címke (8, 16, ill. 20 bites lehet)
label	Címke neve
mm * *+ *- +*	A TBLPTR regiszter kezelésének módja tábla íráskor és olvasáskor: Nem változik a TBLPTR értéke (se íráskor, se olvasáskor) Az olvasás vagy írás után 1-gyel növekszik a TBLPTR Az olvasás vagy írás után 1-gyel csökken a TBLPTR Az olvasás vagy írás előtt 1-gyel növekszik a TBLPTR
n	Relatív cím (2-es komplemensben adott szám) a feltételes elágaztató utasításoknál, vagy közvetlen cím szubrutin hívásnál, visszatéréskor, ugráskor.
PRODH	A szorzás eredményének felső bájtja
PRODL	A szorzás eredményének alsó bájtja
s	Fast call/return módot kiválasztó bit s=0: nincs mentés az árnyékregiszterekbe s=4: mentés/töltés az árnyékregiszterekbe (fast módus)
u	Nem használt, vagy nem változik
WREG	Munkaregiszter (akkumulátor)
x	Figyelmen kívül hagyható (0 v. 1) A fordító nullát generál. Ez a többi Microchip szoftverrel való kompatíbilitáshoz szükséges.
TBLPTR	21 bites táblamutató (egy programmemória beli címre mutat)
TABLAT	A táblából kiolvasott értéket tárolja
TOS	TOP OF STACK, a verem teteje
PC	Programszámláló
PCL	Programszámláló alsó bájtja
PCH	Programszámláló felső bájtja
PCLATH	Programszámláló felső bájtját tároló regiszter
PCLATU	Programszámláló legfelső bájtját tároló regiszter
GIE	Általános megszakítás engedélyezés
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
C, DC, Z, OV, N	Az ALU státuszbitjei: átvitelbit, fél átvitel, zéróbit, túlcsordulásbit, előjelbit
[ ]	Feltételes
( )	Tartalom
→	Hozzárendelés
< >	Bitterület a regiszterben
€	Halmaz eleme
Dőlt betű	A felhasználó által definiált

Mnemonik Operandus	Leírás	Ciklus	16 bites kód MSB LSB	Állított jelzőbitek	Megjegyzés
Bájt orientált fájlregiszter műveletek					
ADDWF f,d,a	W és f összeadása	1	0010 01da ffff ffff	C,DC,Z,OV,N	1,2
ADDWFC f,d,a	W, f és az átvitelbit összeadása	1	0010 00da ffff ffff	C,DC,Z,OV,N	1,2
ANDWF f,d,a	W és f ÉS kapcsolata	1	0001 01da ffff ffff	Z,N	1,2
CLRF f,a	f törlése	1	0110 101a ffff ffff	Z	2
COMF f,d,a	f komplementálása	1	0001 11da ffff ffff	Z,N	1,2
CPFSEQ f,a	WREG és f összehasonlítása, átlép ha =	1 (2 v.3)	0110 001a ffff ffff	nincs	4
CPFSGT f,a	WREG és f összehasonlítása, átlép ha f>	1 (2 v.3)	0110 010a ffff ffff	nincs	4
CPFSLT f,a	WREG és f összehasonlítása, átlép ha f<	1 (2 v.3)	0110 000a ffff ffff	nincs	4
DECF f,d,a	f csökkentése	1	0000 01da ffff ffff	C,DC,Z,OV,N	1,2,3,4
DECFSZ f,d,a	f csökkentése, átlép ha 0	1 (2 v.3)	0010 11da ffff ffff	nincs	1,2,3,4
DCFSNZ f,d,a	f csökkentése, átlép ha nem 0	1 (2 v.3)	0100 11da ffff ffff	nincs	1,2
INCF f,d,a	f növelése	1	0010 10da ffff ffff	C,DC,Z,OV,N	1,2,3,4
INCFSZ f,d,a	f növelése, átlép ha 0	1 (2 v.3)	0011 11da ffff ffff	nincs	4
INFSNZ f,d,a	f növelése, átlép ha nem 0	1 (2 v.3)	0100 10da ffff ffff	nincs	1,2
IORWF f,d,a	WREG és f VAGY kapcsolata	1	0001 00da ffff ffff	Z,N	1,2
MOVF f,d,a	f mozgatása	1	0101 00da ffff ffff	Z,N	1
MOVFF f_s, f_d	f_s (forrás)mozgatása 1. szó f_d -be (cél) 2. szó	2	1100 ffff ffff ffff 1111 ffff ffff ffff	nincs	
MOVWF f,a	WREG mozgatása f-be	1	0110 111a ffff ffff	nincs	
MULWF f,a	WREG és f összeszorzása	1	0000 001affff ffff	nincs	
NEGF f,a	f kettes komplemensének képzése	1	0110 110a ffff ffff	C,DC,Z,OV,N	1,2
RLCF f,d,a	f forgatása balra átvitelbiten keresztül	1	0011 01da ffff ffff	C,Z,N	
RLNCF f,d,a	f forgatása balra átvitelbit kihagyásával	1	0100 01daffff ffff	Z,N	1,2
RRCF f,d,a	f forgatása jobbra átvitelbiten keresztül	1	0011 00da ffff ffff	C,Z,N	
RRNCF f,d,a	f forgatása jobbra átvitelbit kihagyásával	1	0100 00da ffff ffff	Z,N	
SET f	f 1-be állítása	1	0110 100a ffff ffff	nincs	
SUBFWB f,d,a	f kivonása a WREG-ből áthozattal	1	0101 01da ffff ffff	C,DC,Z,OV,N	1,2
SUBWF f,d,a	WREG kivonása f-ből	1	0101 11da ffff ffff	C,DC,Z,OV,N	
SUBWFB f,d,a	WREG kivonása f-ből áthozattal	1	0101 10da ffff ffff	C,DC,Z,OV,N	1,2
SWAPF f,d,a	f alsó és felső 4 bitjének felcserélése	1	0011 10da ffff ffff	nincs	4
TSTFSZ f,a	f tesztelése és átlépés, ha 0	1 (2 v.3)	0110 011a ffff ffff	nincs	1,2
XORWF f,d,a	WREG és az f kizáró-VAGY kapcsolata	1	0001 10da ffff ffff	Z,N	
Bit orientált fájlregiszter műveletek					
BCF f,b,a	f adott bitjének törlése	1	1001 bbba ffff ffff	nincs	1,2
BSF f,b,a	f adott bitjének 1-be állítása	1	1000 bbba ffff ffff	nincs	1,2
BTFSC f,b,a	f adott bitjének tesztelése és átlép, ha 0	1 (2 v.3)	1011 bbba ffff ffff	nincs	3,4
BTFSS f,b,a	f adott bitjének tesztelése és átlép, ha 1	1 (2 v.3)	1010 bbba ffff ffff	nincs	3,4
BTG f,d,a	f adott bitjének invertálása	1	0111 bbba ffff ffff	nincs	1,2

Megjegyzések:

1. Amikor a PORT értéke változik, és a művelet eredménye önmagába íródik (pl. MOVF PORTB,0,1 vagy MOVF PORTB,F,A) az az érték kerül beírásra, ami jelenleg a lábon mérhető. Pl. ha a tárolt adat a regiszterben 1, és a láb bemenetnek van konfigurálva, és ezt a lábat egy külső eszköz 0-ba húzza, akkor a regiszterbe 0 kerül visszaírásra.
2. Abban az esetben, ha a művelet a TMR0 regiszterre vonatkozik (d=1, vagy d=f), és az előosztó a TMR0-hoz van rendelve, akkor az előosztó törlődik.
3. Amikor a programszámláló (PC) változik az utasítás két ciklus hosszú lesz. A második ciklusban a NOP utasítás kerül végrehajtásra.
4. Némely utasítás 2-szó hosszúságú. A második szó ezekben az utasításokban NOP-ként hajtódik végre, kivéve ha az utasítás első szavának folytatása ez a 16 bit.

Mnemonik Operandus	Leírás	Ciklus	16 bites kód MSB LSB	Állított jelzőbitek	Megjegyzés
Vezérlő utasítások					
BC n	ugrás, ha az átvitelbit 1	1 (2)	1110 0010 nnnn nnnn	nincs	
BN n	ugrás, ha az előjelbit 1	1 (2)	1110 0110 nnnn nnnn	nincs	
BNC	ugrás, ha az átvitelbit 0	1 (2)	1110 0011 nnnn nnnn	nincs	
BNN n	ugrás, ha az előjelbit 0	1 (2)	1110 0111 nnnn nnnn	nincs	
BNOV n	ugrás, ha túlcsordulásbit 0	1 (2)	1110 0101nnnn nnnn	nincs	
BNZ n	ugrás, ha a zéróbit 0	2	1110 0001 nnnn nnnn	nincs	
BOV n	ugrás, ha túlcsordulásbit 1	1 (2)	1110 0100 nnnn nnnn	nincs	
BRA n	feltétel nélküli ugrás	1 (2)	1101 0nnn nnnn nnnn	nincs	
BZ n	ugrás, ha a zéróbit 1	1 (2)	1110 0000 nnnn nnnn	nincs	
CALL n,s	Szubrutin hívás 1. szó 2. szó	2	1110 110s kkkk kkkk 1111 kkkk kkkk kkkk	nincs	
CLRWDT -	Watchdog Timer törlése	1	0000 0000 0000 0100	$\overline{TO}$, $\overline{PD}$	
DAW -	WREG decimális korrekciója	1	0000 0000 0000 0111	C	
GOTO n	feltétel nélküli ugrás 1. szó 2. szó	2	1110 1111kkkk kkkk 1111 kkkk kkkk kkkk	nincs	4
NOP -	nincs kijelölt műveletvégzés	1	0000 0000 0000 0000	nincs	
NOP -	nincs kijelölt műveletvégzés	1	1111 xxxx xxxx xxxx	nincs	
POP -	kivétel a veremből	1	0000 0000 0000 0110	nincs	
PUSH -	beírás a verembe	1	0000 0000 0000 0101	nincs	
RCALL n	relatív szubrutin hívása	2	1101 1nnn nnnn nnnn	nincs	
RESET	Szoftveres reszet	1	0000 0000 1111 1111	mindegyik	
RETFIE s	visszatérés megszakítás engedélyezéssel	2	0000 0000 0001 000s	GIE/GIEH PEIE/GIEL	
RETLW k	visszatérés a WREG-ben egy konstanssal	2	0000 1100 kkkk kkkk	nincs	
RETURN s	visszatérés a szubrutinból	2	0000 0000 0001 001s	nincs	
SLEEP -	Szundi üzemmód	1	0000 0000 000 0011	$\overline{TO}$, $\overline{PD}$	

Megjegyzések:

1. Amikor a PORT értéke változik, és a művelet eredménye önmagába íródik (pl. MOVF PORTB,0,1 vagy MOVF PORTB,F,A) az az érték kerül beírásra, ami jelenleg a lábon mérhető. Pl. ha a tárolt adat a regiszterben 1, és a láb bemenetnek van konfigurálva, és ezt a lábat egy külső eszköz 0-ba húzza, akkor a regiszterbe 0 kerül visszaírásra.
2. Abban az esetben, ha a művelet a TMR0 regiszterre vonatkozik (d=1, vagy d=f), és az előosztó a TMR0-hoz van rendelve, akkor az előosztó törlődik.
3. Amikor a programszámláló (PC) változik az utasítás két ciklus hosszú lesz. A második ciklusban a NOP utasítás kerül végrehajtásra.
4. Némely utasítás 2-szó hosszúságú. A második szó ezekben az utasításokban NOP-ként hajtódik végre, kivéve ha az utasítás első szavának folytatása ez a 16 bit.

Mnemonik Operandus	Leírás	Ciklus	16 bites kód MSB LSB	Állított jelzőbitek	Megjegyzés
Konstans műveletek					
ADDLW	konstans hozzáadása a WREG-hez	1	0000 0000 kkkk kkkk	C,DC,Z,OV,N	
ANDLW	konstans illetve a WREG ÉS kapcsolata	1	0000 1011 kkkk kkkk	Z,N	
IORLW	konstans és a WREG VAGY kapcsolata	1	0000 1001 kkkk kkkk	Z,N	
LFSR	FSR feltöltése 2. szó egy konstanssal 1. szó	2	1110 1110 00ff kkkk 1111 0000 kkkk kkkk	nincs	
MOVLB	BSR feltöltése egy konstanssal <3:0>	1	0000 0001 0000 kkkk	nincs	
MOVLW	konstans betöltése a WREG-be	1	0000 1110 kkkk kkkk	nincs	
MULLW	konstans és a WREG összeszorzása	1	0000 1101 kkkk kkkk	nincs	
RETLW k	visszatérés a WREG-ben egy konstanssal	2	0000 1100 kkkk kkkk	nincs	
SUBLW	WREG kivonása a konstansból	1	0000 1000 kkkk kkkk	C,DC,Z,OV,N	
XORLW	WREG és a konstans kizáró-VAGY kapcsolata	1	0000 1010 kkkk kkkk	Z,N	
Adatmemória ↔ programmemória műveletek					
TBLRD*	tábla olvasás	2	0000 0000 0000 1000	nincs	
TBLRD*+	tábla olvasás utólagos növeléssel		0000 0000 0000 1001	nincs	
TBLRD*-	tábla olvasás utólagos csökkentéssel		0000 0000 0000 1010	nincs	
TBLRD+*	tábla olvasás előzetes növeléssel		0000 0000 0000 1011	nincs	
TBLWT*	tábla írás	2(5)	0000 0000 0000 1100	nincs	
TBLWT*+	tábla írás utólagos növeléssel		0000 0000 0000 1101	nincs	
TBLWT*-	tábla írás utólagos csökkentéssel		0000 0000 0000 1110	nincs	
TBLWT+*	tábla írás előzetes növeléssel		0000 0000 0000 1111	nincs	

Megjegyzések:

1. Amikor a PORT értéke változik, és a művelet eredménye önmagába íródik (pl. MOVF PORTB,0,1 vagy MOVF PORTB,F,A) az az érték kerül beírásra, ami jelenleg a lábon mérhető. Pl. ha a tárolt adat a regiszterben 1, és a láb bemenetnek van konfigurálva, és ezt a lábat egy külső eszköz 0-ba húzza, akkor a regiszterbe 0 kerül visszaírásra.
2. Abban az esetben, ha a művelet a TMR0 regiszterre vonatkozik (d=1, vagy d=f), és az előosztó a TMR0-hoz van rendelve, akkor az előosztó törlődik.
3. Amikor a programszámláló (PC) változik az utasítás két ciklus hosszú lesz. A második ciklusban a NOP utasítás kerül végrehajtásra.
4. Némely utasítás 2-szó hosszúságú. A második szó ezekben az utasításokban NOP-ként hajtódik végre, kivéve ha az utasítás első szavának folytatása ez a 16 bit.

[4]