Debreceni Egyetem
Informatikai Kar

Komplex iskolai rendszer és kommunikációs környezete

Témavezető:
dr. Almási Béla

Külső témavezető:
Jászay Gábor

Készítette:
Leidgeb Ákos
Mérnök informatikus

Debrecen
2010

1 TARTALOM

2 BEVEZETÉS

2.1 A kezdetektől napjainkig

„Önkéntelenül a csengő felé mentek, amely éppen a nyolcadik osztály előtt volt a falba beerősítve, s a pedellust nézték, ahogy kurta karját felnyújtotta, s megfogta a csengő lelógó vastag drótfogantyúját, s rángatni kezdte a kisded harangot.”

Móricz Zsigmond: Légy jó mindhalálig

Volt idő mikor még a fenti idézetben olvasott módon jelezték az iskolákban az óra elejét és végét. A kis harangot még ma is sok középiskolában, kollégiumban meg lehet találni a falra erősítve, mint a régi korokat idéző emléktárgyat. Habár semmi kétség nem merül fel a rángatással működésbe hozható harang hatékonysága felől, a mai intézményeinkben szinte kivétel nélkül az elektromechanikus és a dallamcsengők azok, amik ezt a feladatot ellátják. Egy átlagos középiskolai tanítási nap szerves részei az imént említett órák elejét és végét jelző csengetések, valamint a közérdekű információk behirdetései a termekbe telepített hangszórókon keresztül. Napjainkban az iskoláknak jelentős része rendelkezik valamilyen formában ezekkel a rendszerekkel. A legelterjedtebb verzió talán még ma is az elektromechanikus, portás által üzemeltetett csengő és a 100 voltos vonalon meghajtott hirdetőrendszer. Egyes helyeken már digitális időzítő segítségével vezérlik a csengők megszólaltatását. Néhány újabb intézményünkben a két különálló berendezés ötvözése is megvalósul azáltal, hogy a hirdetőrendszeren egy, vagy több dallamot játszanak be az órák kezdetének és végének jelzése céljából. Ilyen hangrendszerre csatlakoztatható programozható dallamcsengőket kifejezetten erre a célra már több, akár magyar cég is gyárt. Ezeknek a berendezéseknek az elektromechanikus-nyomogatós megoldással szemben hatalmas előnyük az automatizáltság és a programozhatóság. Ez az előny azonban olykor inkább hátrány, hiszen a különleges alkalmak során vagy ki kell kapcsolni (jobb esetben), vagy át kell programozni az eszközt, amihez rendszerint hozzáértő személyre van szükség. Általában ezen kis szépséghiba ellenére is nagy népszerűségnek örvend ez a megoldás.

2.2 Mi is az a komplex iskolai rendszer?

A volt középiskolámban is sokáig az imént említett programozható dallamcsengő volt az, ami az órák elejét és végét jelezte. A gimnázium az elmúlt 15 évben külső és belső felújításon is átesett, hogy a napjaink elvárásainak megfelelő korszerű oktatási intézményként működhessen. A kezdetektől ott dolgozó villamosmérnök diplomával is rendelkező informatika tanár Jászay Gábor fejében már a belső felújítások elején megfogalmazódott az a szándék, hogy a dallamcsengős rendszert egy korszerűbb és intelligensebb megoldás váltsa fel. A dolog indítéka a hangszórók állapotának termenkénti menedzselhetőségének kialakítása volt úgy, hogy ne kelljen minden hangszórótól egy vezetékpárt eljuttatni a titkárságig, mert az közel 50 darab hangszóró esetén már kezelhetetlenül sok huzal lenne. A belső felújítások során, (habár nem volt betervezve a közeljövőben ennek a kiépítése) előrelátó módon a villamos hálózat helyrehozásának keretén belül beköttette a majdani rendszer kommunikációs gerinceként szolgáló kábeleket a termekbe.

Az elképzelés a felújítási munkálatok befejeztével még sokáig csak terv szintjén létezett, hiszen az egyéb fontosabb tevékenységek háttérbe szorították ennek a nagyobb volumenű, ám nem létfontosságú szolgáltatásnak a kiépítését. Az egész nagyjából 3 éve kapott ismét lendületet, mikor a vezetés részéről felmerült az igény, hogy a termekben egységes órarendszer kerüljön beszerelésre első sorban azért, hogy az érettségi és a tanórák során mindenhol egységes legyen az időalap. A megoldás szinte adta magát, miszerint a már eltervezett rendszer részeként kerülne ez is megvalósításra, ezáltal mindegyik óra közös szinkronjelet kapna. A tervezés első szakaszában átgondolásra került, hogy még milyen szolgáltatásokat célszerű integrálni a rendszerbe. A fő szempontokat az elmúlt évek során gyűjtött tapasztalatok, hiányosságok és igények adták, ezáltal számos olyan ötlet felmerült, amibe nem invesztálnának feltétlen pénzt és energiát külön, de ha mód van egy rendszer részeként megvalósítani, akkor érdemes belevágni. Az alapgondolat szerint egy központi egység vezérelné a csengetéseket, valamint a termekben lévő hangszórók be és kikapcsolásáért is ez felelne. Minden terembe szükség van egy alegységre, ami kommunikál a központtal a már behúzott kábelen keresztül és ezek az alegységek kapcsolgatnák a hangszórókat.

Az új igények megjelenésével bővült a feladatkör is. Az új tervek alapján a rendszer felépítése úgy módosult, hogy a konfigurációjához ne kelljen szaktudás, ám azt mégis rugalmasan és sokoldalúan lehessen használni. A cél elérése érdekében a központ az ethernet hálózathoz illeszthetőnek lett tervezve, így azt bármelyik az intézményben lévő számítógépről el lehet érni.

A hangszóróvezérlő egységek a funkciójuk szempontjából két típust különböztetnek meg. Az egyik az eredeti ötlethez igazodó, csupán a hangszórók vezérlését lehetővé tevő modul, ami a központtal kommunikálva az igényeknek megfelelően képes az adott térrészen aktív, illetve passzív állapotba hozni a 100 voltos hangszórót. A másik többnyire a tantermekbe tervezett, az újabb igények megjelenésének köszönhetően már egyéb funkciókat is ellát. Az egyik legfontosabb ezek közül talán az időkijelzés négy darab hétszegmenses kijelző segítségével úgy, hogy az minden teremben szinkronban és a lehető legpontosabban történjen. Ez az egységtípus már inkább hasonlít egy digitális órára, hiszen a hétszegmenses kijelzők nagy mérete miatt az őket borító plexi mögött kényelmesen elfér a kijelzőt és hangszórót vezérlő modul. Az egység másik fő funkciója az egyes szolgáltatások lokális vezérlésének biztosítása távirányító segítségével. A modulon lévő infra vevő lehetőséget biztosít helyszíni szolgáltatások elérésére, ugyanis a megfelelő távkapcsolók jelét értelmezve az óra képes átváltani visszaszámláló üzemmódba (dolgozatok írásánál hasznos), valamint a jelet képes akár a központnak is továbbítani ahol további szolgáltatások vezérelhetőek. Ilyen szolgáltatás lenne például a központnál található AV-eszközök vezérlése, így a filmnézés, vagy a zenék lejátszása az épületen belül bárhol elérhetővé válhatna.

A rendszer sokoldalúságát mi sem bizonyítja jobban, hogy a tantermi és folyosói eszközök kommunikációs vonalára egyéb rendszerek is kapcsolódhatnak, így azok felügyelete és vezérlése is központosíthatóvá válik. Ilyen rendszer például az informatika termekbe telepített esőérzékelő, ami azért bír nagy jelentőséggel, mert a nyári időszakban fontos, hogy a viharok idején a tetőtéri ablakok be legyenek zárva, nehogy szétázzanak a számítógépek. A tetőtéri ablakok állapotáról adatokat gyűjt a feldolgozóegység és ha elered az eső és valamelyik ablak nyitva van, akkor jelez. Az esőérzékelő becsatlakoztatásával a másik szárnyban lévő portára lehet majd kijelzőt telepíteni, ami az ablakok állapotáról ad információt.

A tervek szerint a közérdekű hirdetések menedzselését is ez a rendszer venné át, oly módon, hogy a hirdetés, a hangszórók állapota, valamint a csengetések közti kapcsolat felügyelete az ő feladatává válna.

2.3 Itt tartunk most

Láthatjuk, hogy ez a rendszer már merőben túlmutat a csengetés vezérlésén és egy magasabb szinten valósítja meg az iskolaspecifikus szolgáltatások megteremtését, működtetését. Jelenlegi állapotában a fő csapások mint például a csengetés, vagy a hangszórók vezérlése már működnek. Hátra van még a hirdetésmenedzselés, az AV-eszközök vezérlése, valamint az egyéb rendszerek beintegrálása. Jómagam a projektbe a központ készítésénél csatlakoztam és tevékenységem ekkor fogadtattam el szakdolgozatom témájául.

Az én részem a megvalósításból első sorban az ethernet kommunikáció és a felhasználói oldal megvalósítása volt, de részt vettem az egyéb egységek hardveres építéseinél és a központ programozásában is segédkeztem. A dolgozat címe talán kissé általánosan hat, de láthatjuk, hogy ha ennél jobban akarnánk konkretizálni, akkor az már közel egy egész bekezdést felölelne. A következőkben a jelenlegi állapotában működő rendszer felépítését, működését és a rendszerben folyó kommunikációs szabványokat, előírt és általunk meghatározott protokollokat szeretném bemutatni.

3 A RENDSZER FELÉPÍTÉSE ÉS FUNKCIÓI

3.1 A rendszer felépítése

Ahhoz, hogy belevágjunk a részletek ismertetésébe fontosnak tartom, hogy bemutassuk a rendszert a maga teljességében is. Erre azért van szükség, hogy az egyes elemek, azok működése és a kommunikációs viszonyok bemutatása során el tudjuk helyezni azokat a funkcionalitásuk alapján. A 3-1. ábrán a tervezett rendszer sematikus elrendezését láthatjuk, melyen a legfontosabb elemeket ábrázoltam a teljesség igénye nélkül. A jelenlegi állapottól függetlenül az ábra alapján négy nagyobb részt különíthetünk el az eszközök szerepköre és elhelyezkedése alapján.

- Az első és legfontosabb a központ és a hozzá tartozó vezérlőegységek, melyek rendre: a tápegység, a DCF vevő, a dallamgenerátor, az AV-eszközök és az erősítők vezérlésére szolgáló modul, a 100V-os erősítők, valamint az AV-eszközök. A Központ maga az események ütemezéséért, a vezérlésért valamint a kommunikációk menedzseléséért felel. Felépítésével és a benne lévő alkatrészek közötti viszonnyal a későbbiekben még részletesen foglalkozunk.

  A meghajtást ellátó tápegység egy duplikált megoldáson alapszik, melynek a lényege, hogy két teljesen azonos toroid transzformátort kötünk párhuzamosan, melyek egyenirányított effektív feszültsége közel 24 volt. Abban az esetben, ha valamelyik tekercs meghibásodna, a másik még mindig képes ellátni az egységeket energiával. Az áramfelvételt figyelve észrevehető, ha valamelyik ág felmondja a szolgálatot. Ekkor a rendszer takarékos fogyasztásra áll át.

  A rendszer által nyújtott pontos-idő szolgáltatáshoz szükség van egy DCF vevőre ami a DCF77 nevű pontos időt sugárzó jeladó jelét fogja és konvertálja át a mikrovezérlő által is értelmezhető jelsorozattá. Magát az időt a központ viszonylag pontos belső órája adja, de azt a DCF által lesugárzott időinformációval igazítjuk azt az atomórához.

  A dallamgenerátor a különböző csengetések dallamait állítja elő, aminek a hangfrekvenciás jelét galvanikusan leválasztva a 100V-os erősítők segítségével juttatjuk el az épület minden részébe. Maga a dallamgenerátor jelenleg egy nem véglegesített modul. Az alapötlet az eddigi dallam többszólamúvá tétele azáltal, hogy mindegyik szólamért egy-egy mikrokontorller felelne még nem került megvalósításra. Jelenleg még mindig az alapdallam és a be- és kicsengetés előtt figyelmeztető hangpár az, ami megszólal. Felmerült olyan ötlet is, hogy egy MP3 modult teszünk az általunk készített egység helyére, így a dallamokat egyszerűen, akár az alkalomnak megfelelően lehetne változtatni. Maga a modul a központtal USART[1]-on keresztül kommunikál, valamint a tápfeszültséget is a központ felől kapja. Az előre beállított időpontok elérkezésekor a központ egy parancsot generál annak megfelelően, hogy melyik dallamot kívánja lejátszatni a dallamgenerátor által, majd ezt a parancsot a soros vonalon elküldi a modulnak. A dallamgenerátor mikrovezérlője értelmezi a parancsot és annak alapján elugrik a megfelelő programrészre ami a kívánt dallam lejátszásáért felel. A dallamot négyszögjelből alkotja meg, amit passzív tagokkal és egy műveleti erősítővel kicsit simít és felerősít. A felerősített hangfrekvenciás jel egy 1:1 tekercsarányú hangfrekvenciás transzformátor által galvanikusan leválasztásra kerül, majd eljut a 3 végerősítőhöz.

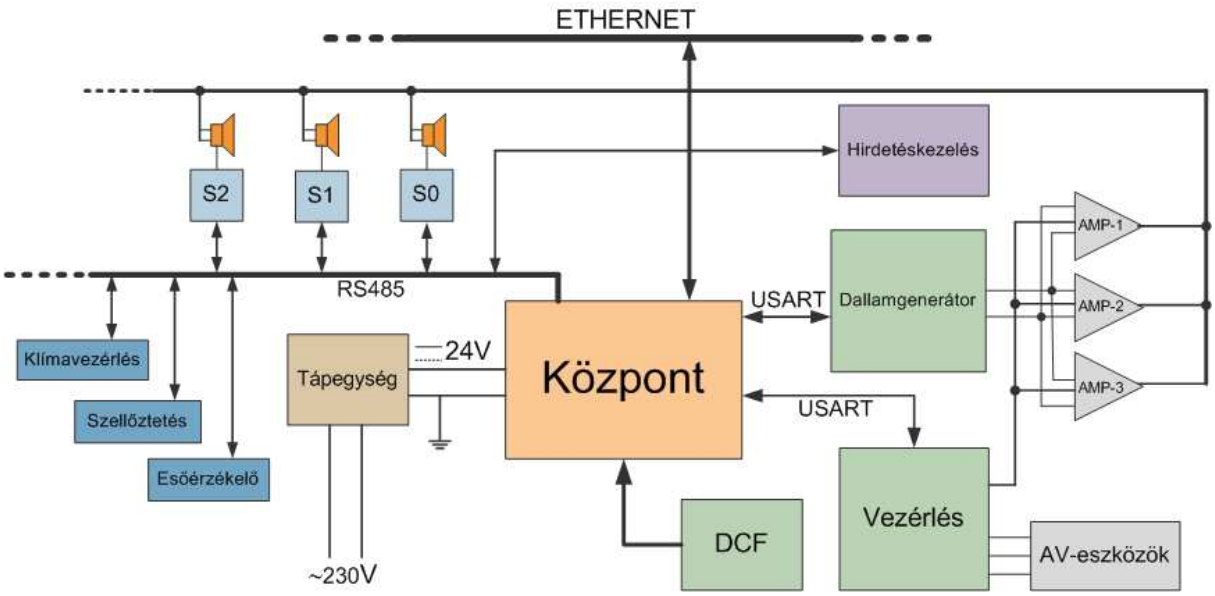
  Azért, hogy az iskola egész területén hallani lehessen a csengetéseket, a hirdetéseket, vagy az épp bejátszott hangokat, a három 100 Voltos erősítő felel. Ezek hajtják meg szintenként a telepített hangszórókat. Nyilván ezeknek csak akkor érdemes működniük, amikor használatban vannak, éppen ezért a ki- bekapcsolásukat is a rendszer végzi egy külön erre a célra fenntartott egység segítségével. A modul az

[1] Universal Synchronous Asynchronous Receiver Transmitter

ábrán a „Vezérlés” nevet kapta, hiszen ezen találhatók a külső eszközöket vezérlő elemek. Az erősítők vezérlése mellett a zenei és képi anyagok lejátszását biztosító AV-eszközök irányítását is ez menedzseli, ugyanis (ahogy arra a bevezetőben is utaltam) lehetőségünk van a termekbe kihelyezett egységeken keresztül távirányítóval vezérelni a központnál elhelyezett DVD, video, valamint CD lejátszókat.

- A második nagyobb egységnek a rendszer gerincét alkotó, a termekben és folyosókon elhelyezett állomások (a rajzon: S1, S2, S3, stb.) összessége tekinthető, melyekből ahogy azt már korábban említettem két típus létezik. Az egyik, ami talán a többséget is alkotja, és amely rendelkezik hétszegmenses kijelzővel általában a termekben és szobákban kapott helyet, míg a másik típust a folyosói állomások adják, ahol nincs szükség az idő megjelenítésére. Mindegyik állomás tartalmaz egy modult, ami az RS485-ös vonalon keresztül kommunikál a központtal, valamint biztosítja a helyi szolgáltatásokat. Az egységek elkészítésénél a fő szempont az igények kielégítése és a sokoldalúság volt, így aztán, ami felmerült - és megvalósíthatónak tűnt - az bele lett tervezve. A kijelzős kivitel esetén gyakorlatilag egy sötétített plexi előlappal ellátott fakeretről van szó (függelék, 15-3., 15-4. ábra), melyben 4 darab hétszegmenses kijelző, valamint a vezérlőpanel kapott helyet. Ezzel szemben az egyszerűsített változatok esetén csak vezérlőpanel található. Az egységek fő feladata a pontos idő kijelzése, a hangszórók ki- bekapcsolása, a távirányító jelének a továbbítása/feldolgozása.
- A harmadik és talán a legkevésbé kiforrott rész leginkább olyan elemekből tevődik össze, melyek önálló működésre is képesek lennének. Ezeknek a rendszerbe integrálása (ahogy azt a bevezetőben is olvashattuk) elsősorban a központosítás céljából történik és valószínűleg ez a rész kerül a legkésőbb megvalósításra. Ide elsősorban az informatika termekbe elhelyezett felügyeleti és vezérlési eszközök, valamint a hirdetéskezelő egység sorolható. Az informatika termek az egyik szárny tetőterében kaptak helyet. Itt a tetőtéri viszonyok miatt már korábban felmerültek különböző igények a hőmérséklet szabályozására, vagy a billenő ablakok nyitott állapotának jelzésére eső esetén. Az esőérzékelő rendszer, ami az utóbbit hivatott ellátni, már megvalósításra került és a tervek szerint az RS485-ös gerincvonalon kommunikálna a központtal, ami szintén ezen a vonalon keresztül küldené el az adatokat egy, a portán lévő kijelzőre. A termek szellőztetésének automatizálása és a klímavezérlés kivitelezése még nem történt meg, de a tervek szerint azok is ezen a vonalon kapcsolódnának a központi egységhez. A sorból talán a hirdetéskezelés lóg ki a legjobban, ami a titkárságon kapott helyet és a hirdetések megkezdése és lebonyolítása során felmerülő vezérlési, jelzési funkciókat látja el.
- A negyedik, és leginkább elszeparálódó egységet a felhasználó és rendszer közti kapcsolatot megteremtő PC oldali alkalmazások és a velük való kommunikációt megvalósító kapcsolat alkotja. A rendszer tervezése során fontos szempont volt a felhasználóbarát és könnyen személyre szabható kezelési felület. Éppen ezért merült fel az ötlet, hogy az ethernet hálózathoz illesztve a központot bármelyik iskolai számítógépről el lehessen érni a megfelelő szoftverrel. A jelenlegi állapotban négy

célszoftver áll rendelkezésre amiből az egyik teljes körű hozzáférést biztosít a rendszerhez adminisztrációs és szerviz célokból. A maradék három program a megfelelő személyek keze alatt teszi beállíthatóvá az adott napra vonatkozó csengetési rendet, valamint a hangszórók működési állapotát.



3-1. ábra, A rendszer sematikus blokkvázlata

A jelenlegi állapotban a központ a tápegységgel, a termekbe telepített órák és hangszórók, a DCF vevő, a dallamgenerátor és a PC oldali programok azok, amik már használatban vannak és voltaképpen ezek az elsődleges felhasználási területek képviselői. A dolgozat a továbbiakban ezen elemek tárgyalásán keresztül boncolgatja tovább a rendszert.

3.2 A rendszer jelenleg ellátott funkciói

A tervezés során a hangsúly a pontos idő szolgáltatásra, valamint a csengetés- és a hangszórók vezérlésére helyeződött. A csengetés alapkoncepciója az volt, hogy tartsuk meg a napi legfeljebb nyolc órás tanítást és mindegyik órához négy dallam tartozzon. Legyen egy becsengetést jelző (becsengetés előtt két perccel), egy becsengetés, egy kicsengetést jelző és egy kicsengetés. A rendszer összesen 10 csengetési rendet tud eltárolni, melyekben szabadon beállítható a be- és kicsengetések időpontja, valamint az órák száma. A tanév minden napjára külön-külön lehet beállítani (akár egész évre) a meglévő legfeljebb tíz csengetési rend valamelyikét. Az adott nap csengetési rendjét az igazgatóhelyettes állítja össze, vagy módosítja egy meglévő alapján, és lehetősége van azt kinyomtatni formázott alakban. A

nyomtatványon feltűntethető az esetleges csengetési rend megváltozásának oka és egyéb információk. Mindezt az egyik program menedzseli, ami a központtal kommunikálva kérdezi le, valamint állítja be a csengetési rendeket és az intézmény bármely hálózatba kötött számítógépén futtatható.

A másik fő irányvonal a hangszórók központi menedzselése is hasonlatos igények alapján lett megtervezve. A hangszórók állapotának a beállíthatósága azért fontos, mert vannak alkalmak (érettségi, megbeszélések, tanulmányi versenyek, stb.), amikor nem szabad mindenhol megszólalnia a csengetéseknek/hirdetéseknek. A másik ok pont a fordítottja, amikor csak egy bizonyos tanterembe, vagy tanterem-csoportba szeretnénk hallatni a hangokat. A hangszórók állapotai (ahogy a csengetési rendek) előre elkészíthetők és beállíthatók különböző időpontokra, azonban nulla óra nulla perckor minden visszaáll az alapértelmezett kofigurációba. Összesen 100 darab állapotot hozhatunk létre és 200 darab állapotváltozási bejegyzés állítható be a tanévre. A hangszórók állapotának megváltoztatása nem csak így érhető el. A hirdetések alkalmával megeshet, hogy nem kívánjuk az adott információt az egész intézménybe közzé tenni. Ekkor alkalmi változtatást eszközölünk ami a hirdetés idejére felülírja az akkorra beállított állapotot, majd a hirdetés végeztével visszaállítja azt. Az alkalmi állapotot egy külön program segítségével tudjuk beállítani. A továbbiakban vizsgáljuk meg ezen funkciókat és az őket ellátó eszközök működését.

4 A MIKROVEZÉRLŐK VILÁGA

Ahhoz, hogy a dolog hardveres felépítésében jobban elmélyedjünk, fontosnak tartom, hogy beszéljünk a működés alapjául szolgáló mikrovezérlőkről. Azokat az integrált áramkörből felépített rendszereket, amelyek programot képesek végrehajtani, mikroszámítógépeknek nevezzük. A mikroprocesszor típusától függetlenül a mikroszámítógépnek a következő főbb részekkel kell rendelkeznie:

- Központi egység (CPU)
- A programtár és az adattár (gyakorlatilag a memória), ami lehet közös vagy különálló.
- A be- és kiviteli egység (periféria, I/O egység) ami a külvilággal tartja a kapcsolatot.
- Az egységek közti információ- és adatáramlást biztosító vonalak.

A mikroszámítógépek és az integrálási technika fejlődése lehetővé tette, hogy egy teljes értékű mikroszámítógép minden részegysége egyetlen lapkára kerüljön: ezek a mikrokontrollerek, magyarul mikrovezérlők.

Mikrovezérlők ezrei segítenek minket hétköznapi életünk során. Amikor megnyomjuk a távirányító gombját, amikor kinyitunk egy zenélő képeslapot, vagy amikor ránézünk egy digitális hőmérőre, mindig szembetaláljuk magunkat eme parányi számítógép működésével. Manapság ezek annyira elterjedt eszközök lettek, hogy tulajdonképpen tudomást sem veszünk létezésükről. Nagy előnyük leginkább árukban és a sokoldalúságukban rejlik. Napjainkban számos cég foglalkozik mikrovezérlők fejlesztési célokból való gyártásával. A gyártás során alapvető szemléletmód a családelvűség, ami azt jelenti, hogy a különböző feladatokra használható mikrovezérlőket eltérő kialakításban gyártják. Ez annyit tesz, hogy a mikrovezérlők alapvető egységei (CPU, memória) mindegyik eszközben megtalálhatóak (esetleg eltérő paraméterekkel), de a különféle perifériák már családtól függően kapnak helyet a lapkán. Az ilyen, és hasonló mikrovezérlőkhöz bárki hozzájuthat viszonylag alacsony áron és szabadon felhasználhatja, valamint programozhatja őket. Az egyik legnagyobb forgalmazó, a Microchip Technology már 1985 óta gyártja a PIC (Peripheral Interface Controller) névre keresztelt mikrovezérlő-családjait a nagyközönség számára. Ilyen mikrovezérlőket használtunk mi is a rendszer elkészítése során. Mielőtt azonban részletesebben megvizsgálnánk annak felépítését, ismerkedjünk meg a PIC mikrovezérlők belső világával és működésével.

4.1 A PIC-ről általában

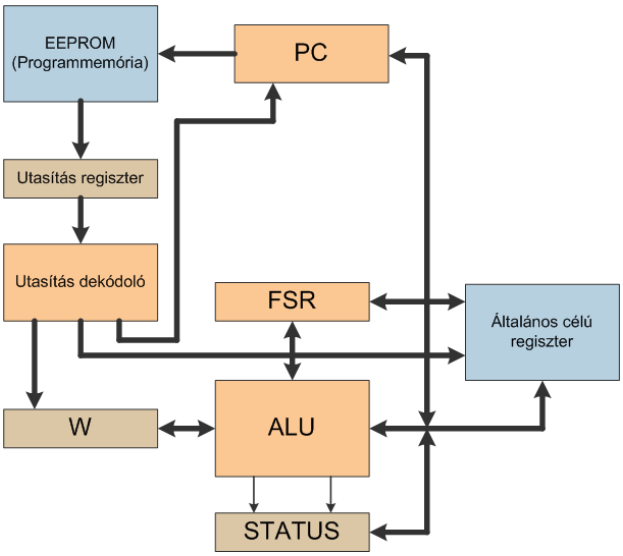
A PIC mikrovezérlők RISC jellegű, viszonylag kevés és egyszerű utasítást végrehajtó processzor-magra épülnek. A memória szervezése Harvard architektúrájú, tehát két külön (egy program és egy adat) tárolóra van felosztva, ezáltal a memóriáért való sorbaállás is minimalizálódik. A két különálló tároló eltérő bitszélességű adat- és programbusz használatát engedi meg, ezáltal lehetővé téve a gyorsabb működést, hiszen a megnövelt utasításhosszban elfér a műveleti kód és az operandus is. A következőkben (az egyszerűség kedvéért a legegyszerűbb) a 12 bites PIC mikrovezérlők általános bemutatásával fogjuk a teljesség igénye nélkül körbejárni ezeknek az eszközöknek a felépítését és működését.

4.2 Az utasítás végrehajtásának menete

Ahogy azt már említettem a mikrovezérlőkben a CPU-n, az adat- és programmemórián kívül még számos egyéb egység is helyet kapott az önálló működés és a célfeladatok ellátása érdekében. Ahhoz, hogy megértsük a működést, vizsgáljuk meg, hogy az egyes utasítások miként is hajtódnak végre.

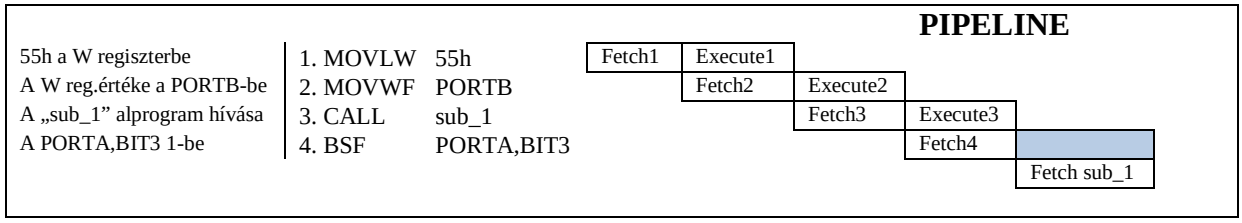
Az 12 bites utasításokból álló programot az EEPROM/Flash programtároló tartalmazza. Ezt a memóriát az ún. PC (Program Counter) címzi meg és az általa megcímzett helyről az aktuális utasításszó az utasításregiszterbe, majd onnan az utasításdekódolóba kerül. Ott megtörténik az utasítás értelmezése, majd az értelmezés alapján a végrehajtása. Ha az utasítás egy közvetlen ugrás, akkor a közvetlen cím beíródik a PC-be és a következő utasítás már erről a címről fog betöltődni az utasításregiszterbe.

A másik lehetőség az, hogy az utasítás logikai, vagy aritmetikai jellegű. Ekkor az egyik operandust betöltjük a W (work) regiszterbe, a másik operandust pedig a következő utasítás operandusmezője által tartalmazott cím alapján fogjuk meghatározni. Mikor ez az utasítás bekerül az utasításdekódolóba, a dekódoláskor az általános célú regisztertömböt fogja megcímezni az utasítás operandusa, majd onnan az adatbuszon keresztül a megcímzett fájlregiszter tartalma az ALU-ba kerül, és megtörténik a művelet végrehajtása. Az eredmény ilyenkor vagy a W-regiszterbe kerül, vagy pedig visszaíródik a megcímzett regiszterbe. Az utasítás végrehajtása után az utasítás típusától és az eredménytől függően az ún. STATUS regiszter értékei is beállítódnak, de ezzel még a későbbiekben részletesen foglalkozunk.



4-1. ábra, Az utasítás végrehajtásában résztvevő modulok

Az egyes utasítások végrehajtása több lépcsőben történik meg. Az, hogy ez mégis mennyi idő alatt zajlik le, azt (mint minden processzornál) az órajel frekvenciája fogja meghatározni. A PIC-ek órajelüket általában belső RC oszcillátorukból, vagy nagyobb pontosságot igénylő alkalmazásuk esetén külső kvarc-kristályból vagy rezonátorból nyerik. A mai PIC mikrovezérlők többsége néhány 4 - 80MHz-es órajelen működik. Egy utasítás végrehajtása a 12 bites mikrovezérlő esetén 4 órajelciklust igényel. Ez családonként és azon belül is utasításonként változó lehet, de a bemutatás kedvéért kövessük ezt a megközelítést. Az első órajel alatt történik az utasítás dekódolása, a második alatt, az operandus kiolvasása a fájlregiszterből, a harmadik során megtörténik a végrehajtás és végül a negyedik során visszaíródnak az értékek. Ha az utasítás típusa olyan, akkor a fájlregiszter-elérés természetesen elmaradhat. Ahhoz, hogy a folyamatokat gyorsítsák, a PIC átlapolt utasításciklust használ, amit más néven pipe-line-nak hívnak. Ennek az eljárásnak a lényege az, hogy míg az utasítás végrehajtása zajlik, addig a következő utasítás betöltése történik a memóriából. Abban az esetben ha az adott utasítás ugró utasítás, akkor a következőt az ugrás helyéről kell betölteni, és az átlapolás által beolvasott címet el kell dobni. A szituációt a 4-2. ábrán követhetjük nyomon, ahol a 4. utasítás egy subrutin-hívást tartalmaz.



4-2. ábra, A pipeline működése subrutinhívás esetén

4.3 Utasításkészlet

A PIC mikrovezérlők által végrehajtható utasítások funkciójuk szerint hat csoportra bonthatók, ezek rendre: logikai utasítások, aritmetikai- és shift utasítások, bitműveletek, adatmozgató utasítások, programvezérlő utasítások és rendszervezérlő utasítások. A logikai utasítások a 8 bites adatok bitenkénti logikai műveleteit végzik. Ez annyit tesz, hogy a klasszikus logikai műveleteket (AND, OR, XOR, stb.) a 8 bites adaton bitenként végzik el, így a maszkolástól kezdve a kitüntetett bitpozíciók módosításáig rengeteg dologra van lehetőségünk általuk. Az aritmetikai és shift utasítások a különböző regisztertartalmak

összeadását, kivonását, eggyel való növelését és csökkentését teszik lehetővé, valamint lehetőségünk van azok eggyel jobbra és balra forgatására, melynek során a „kicsurgó” bit a carry-be kerül. Bit szintű utasításból összesen négy került ebbe a családba: adott bit törlése, adott bit beállítása, utasítás átugrása, ha az adott bit értéke 0, utasítás átugrása, ha az adott bit értéke 1. Az adatmozgató utasítások a W regiszter és a fájlregiszterek közti adatok mozgatását teszik lehetővé. Lehetőségünk van arra is, hogy ne közvetlenül az utasításban található cím határozza meg az operandust, hanem indirekt módon egy speciális regiszter (FSR – File Select Register) tartalmaként megjelenő cím fog erre szolgálni, míg az utasításba ennek a speciális regiszternek fog bekerülni a címe. Ezt nevezzük indirekt (közvetett) címzési módnak. A programvezérlő, ugró és szubrutinkezelő utasítások (ahogy azt a nevük is sejteti), a program működését és az utasítások végrehajtásának sorrendjét vezérlő utasítások. Ezek közül a GOTO és a CALL-RETURN páros az, ami talán a leggyakrabban használt ilyen jellegű utasítás. A GOTO paranccsal a programon belüli ugrást tudjuk megvalósítani, egy a program szövegében előre definiált címkére. Ezzel gyakorlatilag a magasszintű nyelvekben megtalálható különböző ciklusok megfelelőit implementálhatjuk. Amikor bizonyos programrészeket többször is meg akarunk ismételni, akkor nem szoktuk azokat minden alkalommal amikor szükség van rájuk beleírni a programunkba, mert akkor igen hamar túlnőnénk a programmemória méretét, ráadásul a kódunk pillanatok alatt nagyon bonyolulttá válna. Helyette kisebb alprogramokat (subrutinokat) hozunk létre, melyekre a CALL paranccsal tudunk hivatkozni. Ekkor az aktuális cím bekerül egy verembe és a PC-be beíródik a subrutin kezdőcíme. Miután a subrutin lefutott, a végén található RETURN utasítás segítségével visszatérhetünk a hívás helyéhez a programba azáltal, hogy a PC-be a verem tetején lévő cím fog visszaíródni. A rendszervezérlő utasítások a processzor működési állapotát befolyásoló utasítások. Ilyenek például a CPU-t alacsony fogyasztású állapotba küldő SLEEP, vagy a watchdogot törlő CLRWDT utasítás.

4.4 Regiszterek

Ahogy már az előzőekben láthattuk, az utasítások végrehajtásában fontos szerepet játszanak az adatmemóriát alkotó különféle regiszterek és az általuk tárolt értékek. A regisztereknek két nagy csoportját különítjük el. Az első az általános célú fájl regiszterek, melyek a felhasználói program adatainak tárolására szolgálnak. Ezek a bekapcsolási Reset-folyamat alatt véletlen értéket vesznek fel. A másik nagy csoport a működtető fájlregiszterek,

más néven SFR (Special Function Registers), amelyek a CPU és a perifériák működését kezelik. Ide tartoznak a következő regiszterek, regisztercsoportok:

- A valós idejű óra/számláló regisztere(i) (TMRX, X=0,1,2..): Ez a regisztercsoport a mikrovezérlőbe beépített hardveres számlálók értékeit és az azokhoz tartozó beállításokat tartalmazzák.
- A már említett programszámláló (PC), ami a soron következő utasítás címét tartalmazza.
- Az állapotregiszter (Status Register), melynek bitértékei, mindig a mikrovezérlő aktuális állapotától függően fognak beállítódni.
- Az I/O regiszterek, melyek bitértékei megfelelnek a regiszterekhez rendelt portok lábértékeivel. Ezen regiszterek által lehet a külvilág és a program közti kapcsolatot megvalósítani.
- Fájlregiszter választó regiszter (FSR, File Select Register): A már említett indirekt címzést lehet megvalósítani általa.
- További speciális regiszterek szolgálnak az I/O portok konfigurálására és az előosztó kezelésére. Ilyenek például az ADCON, vagy TRIS regiszterek.

4.5 Megszakításkezelés

Sokszor, amikor időkritikus alkalmazásokhoz használjuk a mikrovezérlőnket, fontos, hogy egyes feladatokat el tudjon látni függetlenül attól, hogy éppen mit csinál, vagy a program melyik része hajtódik végre. Ennek érdekében fejlesztették ki a mikrovezérlők megszakításkezelését, melynek lényege, hogy amikor bekövetkezik egy megszakítás, akkor az, az eredetileg futó program végrehajtását leállítja, és helyette egy úgynevezett megszakítási alprogram (ISR – Interrupt Service Rutin) hajtódik végre, majd annak befejeződése után a processzor visszatér és folytatja a megszakított programot. A megszakítás egyik fő feltétele, hogy legyen olyan bemenete, melynek állapota megváltozásakor a CPU képes felfüggeszteni a működését, a megszakított program programszámlálójának az értékét elmenteni és a megszakítási program kezdőcímét betölteni. Miután lefut a megszakítási program, a programszámlálóba visszatöltődik a megszakított program elmentett értéke és folytatódik a program futása. Tulajdonképpen a megszakítás előre nem látható időben bekövetkező szubrutinhívás. A megszakítások egy speciális esete, mikor a megszakítást nem egy bemenethez, hanem az egyik számlálóhoz kötjük, így az, az adott számláló minden túlcsordulása esetén bekövetkezik. A PIC mikrovezérlők esetén egyszerre egy, vagy két megszakítás kiszolgálása valósulhat meg. Abban az esetben, ha a mikrovezérlő egy

megszakítást kezel, és egyszerre több megszakítás következik be, akkor a prioritást azok ISR-ben való sorrendje határozza meg. Ha a mikrovezérlő két megszakítást is képes kezelni, akkor rendszerint a két megszakítási szinthez prioritások vannak rendelve. Abban az esetben, ha bekövetkezik az alacsony prioritású szinten egy megszakítás, akkor annak a futását a magas prioritású szinten lévő megszakítás megszakíthatja, de ugyanez fordítva már nem igaz. A megfelelő működés érdekében ügyelnünk kell a megszakított főprogram és a megszakítási alprogram viszonyára. Vannak a főprogramban olyan részek, melyek időkritikusságuk miatt a helyes működés szempontjából nem tűrik el a megszakítást. Az ilyen programrészek előtt lehetőségünk van globálisan tiltani a megszakítások kezelését, majd az adott programrész végén ismét engedélyezhetjük őket.

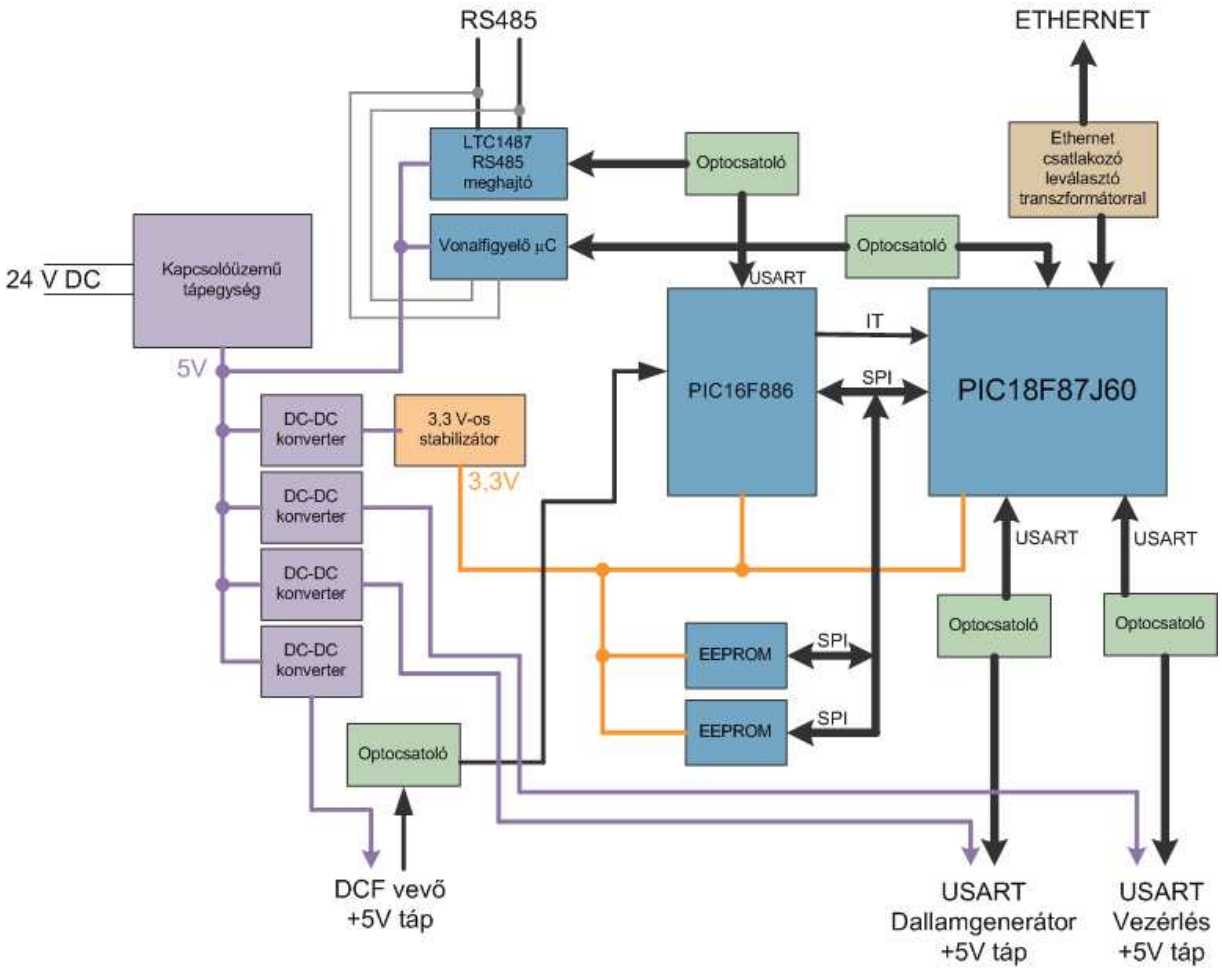
Az imént leírtak csak a legáltalánosabb dolgokat ragadták ki a mikrovezérlők működésének és felépítésének hátteréről. Ezeken a dolgokon kívül még számos finomhangolási és architektúrabeli különbség van akár az azonos márkán belül lévő családok között is. Azt pedig, hogy mennyire elterjedtek és népszerű az alkalmazásuk, mi sem bizonyítja jobban, mint az, hogy a világ számítógépnek nyilvánítható eszközeinek több mint a felét ők alkotják.

5 A KÖZPONT ÉS AZ ÁLTALA HASZNÁLT KOMMUNIKÁCIÓS SZABVÁNYOK

A komplex iskolai rendszer legfontosabb eleme a központi vezérlőmodul. Gyakorlatilag ez felel a rendszerben zajló eseményekért, a belső kommunikációs csatornák működtetéséért és a működéshez szükséges adatok tárolásáért. A következőkben ismerkedjünk meg a központ felépítésével, valamint a benne alkalmazott kommunikációs szabványokkal.

5.1 A központ felépítése

A 5-1. ábrán a központ sematikus blokkvázlata látható. Azt, hogy kapcsolási rajzot mutassak fölöslegesnek tartottam, hiszen a bogarászás során elvesznénk a részletekben, ráadásul az már nem a diplomamunka, hanem a dokumentáció kategóriába sorolná át a dolgozatot.



5-1. ábra, A központ sematikus blokkvázlata

A modul tartalmaz egy központi-, valamint egy vele szorosan együttműködő, kommunikációs mikrovezérlőt, melyek név szerint rendre a PIC18F87J60, valamint a PIC16F886. Gyakorlatilag az utóbbi végzi a termekben és folyosókon lévő egységekkel való kommunikáció lebonyolítását, valamint a DCF-vevőből érkező adatok feldolgozását. A csomópontok elhelyezkedése miatt olyan kommunikációs szabványt kellett használni, ami képes nagy távolságon belül nagy számú eszközzel való kapcsolattartásra anélkül, hogy a külső zavarok hatással lennének rá. A választás az üzemi környezetben is használt RS485-ös szabványra esett. Az RS485-nek szüksége van egy speciális meghajtó áramkörre ami biztosítja a szabványban meghatározott fizikai jellemzőket. A kommunikációs vonalon a kommunikáció során fellépő hardveres hibák detektálására, egy külön mikrovezérlőt alkalmazunk, ami a vonal közvetlen figyelésével tudja az esetleges problémákat jelezni a központi mikrokontrollernek. Az adatok tárolása 2 külső EEPROM-ban történik, melyek egyenként 1024Kbit kapacitással bírnak. A tervezés során az egyik fő szempont az volt, hogy a két központi mikrokontroller és az EEPROM-ok teljesen el legyenek különítve a külső egységektől, hogy a különféle tranziensek, esetleges hibák ne okozzanak zavart a belső működésben. Ahhoz, hogy ezt megvalósítsuk, a tápellátás és a kommunikációs vonalak teljes galvanikus leválasztására volt szükség. A 24 voltos tápból a kapcsolóüzemű tápegység 5 voltos stabil feszültséget állít elő. A galvanikus leválasztás DC-DC konverterek segítségével történik. A DC-DC konverter egy olyan aktív elektronikai elem, ami az egyenáram megfelelő szaggatásával, egy tekercsen keresztül képes leválasztott feszültséget biztosítani, ami bizonyos veszteséggel ismét felhasználható. A modulon 5 darab ilyen DC-DC konverter lett elhelyezve, de ebből kettő párhuzamosan van kötve a nagyobb teljesítmény elérése érdekében, ebből kifolyólag a rajzon csak 4 konvertert ábrázoltunk. Az áram megfelelő leválasztása után eljuttatjuk azt a felhasználási területekre. Két DC-DC konverter a két külső egység, a dallamgenerátor és a vezérlés energiaellátásáért felel. Egy harmadik konverter hajtja meg a DCF vevőt, a negyedik (ami voltaképpen 2 konverter párhuzamosan kötve) pedig a mikrovezérlők és az EEPROM-ok tápellátását biztosítják. A rajzról kitűnhet, hogy a soros vonal meghajtásáért felelő LTC vonalmeghajtó IC és az azt figyelő mikrovezérlő nincs galvanikusan leválasztva. Ennek egyszerűen az a magyarázata, hogy teljesen felesleges lenne, mert az RS485-ös vonal sincs leválasztva amit meghajt. Ha megfigyeljük az ábrát láthatjuk, hogy a mikrovezérlők és az EEPROM-ok nem 5 voltról, hanem 3,3 voltról üzemelnek, éppen ezért a leválasztás után még be kell iktatni egy megfelelő stabilizátor áramkört, ami biztosítja

az eszközök számára a megfelelő feszültséget. A kommunikációs kapcsolatok leválasztása optpcsatolókkal lett megvalósítva. Az optocsatoló ahogy azt a neve is mutatja, fénnyé alakítja az elektromos jelet, majd azt vissza elektromos jellé. Általában egy LED-et és egy fototranzisztort tartalmaz zárt tokban. A LED állapotától függően a fototranzisztor vezetőképessége is változik. A mi esetünkben miután a kommunikáció a legtöbb esetben kétirányú, két optocsatolót kell használni. Az egyszerűség kedvéért az ábrán a kétirányú csatornák leválasztását is egy optocsatolóval szimbolizáltam. Ha az ábrán végignézünk, láthatjuk, hogy a két fő mikrovezérlő, valamint az EEPROM-ok teljesen el vannak galvanikusan szeparálva a külvilágtól. Gyakorlatilag ezek az elemek azok, amik a rendszer belső magját alkotják. Alapvetően a vezérlési és a kommunikációs feladatok nagy részét a PIC18F87J60-as mikrovezérlő látja el, míg a másik a PIC16F886 az RS485-ös kommunikációért, valamint a DCF-vevőből érkező jel feldolgozásáért felel.

5.2 A PIC18F87J60 és a PIC16F886

A PIC18F87J60-as mikrokontroller egy 80 lábú TQFP tokozású mikrovezérlő. A választás első sorban azért esett erre a típusra, mert ez azon család tagja, amibe már hardveresen integrálták a közeghozzáférési és fizikai támogatást az IEEE 802.3 szabványhoz. Ez azért nagy szó, mert gyakorlatilag megvalósíthatóvá vált a beágyazott rendszerek ethernet hálózatba való csatlakoztatása anélkül, hogy az időzítési, és egyéb fizikai szinten történő kommunikációs feltételek megteremtését nekünk kellene biztosítanunk a szabványnak megfelelően. Elég egy leválasztással ellátott RJ45-ös anyacsatlakozót kötnünk a mikrokontorller megfelelő lábaira és ha helyesen írjuk meg a kommunikációhoz szükséges protokollt, akkor az ethernet hálózatban képesek vagyunk a más csomópontokkal való kommunikációra. Arról, hogy ez a mi rendszerünkben hogy valósult meg, a későbbiekben még lesz szó. A mikrovezérlő ezen felül még számos kommunikációs szabványhoz (SPI, I^2C, EUSART) nyújt hardveres támogatást. A programmemória mérete 128 kilobájt az adatmemóriáé pedig 3808 bájt. A mikrokontoller ezen felül tartalmaz még egy 8192 bájtos puffert az ethernet részére, amit megosztva használ az adat küldésére és fogadására. Egy érdekessége a családnak a többszintű megszakításkezelés, ami annyit jelent, hogy megkülönböztetünk alacsony és magas prioritású megszakításokat. Ha egy alacsony prioritású megszakítás kiszolgálás alatt van, és kiváltódik egy magas prioritású, akkor a magas prioritású megszakíthatja az alacsony prioritású kiszolgálását, ám ez fordítva nem történhet

meg. Ennek az előnye leginkább akkor jelentkezik mikor ethernet kommunikáció mellett más, időkritikus folyamatokat szeretnénk futtatni. Az ethernet megszakításai az alacsony prioritású megszakítási szinten helyezkednek el, mert az ethernet kommunikáció képes elviselni akár az 1-2 másodperces késleltetéseket is, ezáltal az időkritikus tevékenységek a magas prioritású megszakítási szinten minden esetben elsőbbséget élveznek.

A PIC16F886-os mikrovezérlő elsősorban az RS584-ös kommunikációért, valamint a DCF által küldött információ feldolgozásáért felel. A mikrokontroller 8192 szavas programmemóriával, 368 bájtos SRAM-mal, valamint egy 256 bájtos EEPROM-mal rendelkezik (az utóbbi kettő az adatmemóriát alkotja). Hardveresen támogatja az RS232, RS485 EUSART kommunikációs szabványokat, valamint modulon lévő elemek közti adatcserére kiélezett SPI és I^2C szabványokat.

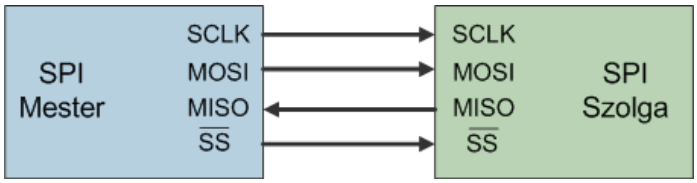
5.3 SPI

A két központi mikrovezérlő és az EEPROM-ok az SPI szabvány szerint kommunikálnak egymással. Az SPI (Serial Peripherial Interface) egy a Motorola által kifejlesztett full-duplex kommunikációs szabvány. Az eszközök mester-szolga viszonyban állnak egymással és egy SPI buszra több eszköz is csatlakozhat, de ezek közül csak egy lehet a mester. A mi esetünkben ezt a szerepet a PIC18F87J60-as mikrokontroller tölti be, a szolgák pedig az EEPROM-ok és a kommunikációs mikrovezérlő. Az SPI kommunikációt használó eszközök közötti busz 4 vezetékből áll (5-2. ábra), melyek a következő felsorolásban leírt funkciókat látják el.

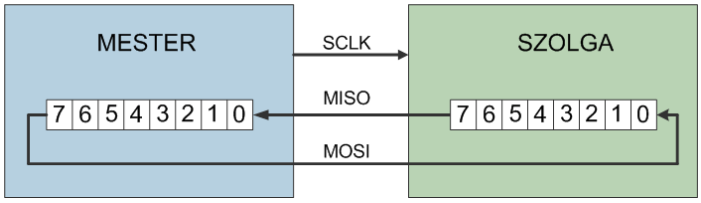
- SCLK – Serial Clock: Ez biztosítja a közös órajelet a szinkron kommunikációhoz. A master generálja és mindegyik slave egységhez el van vezetve. Az órajelre nézve a legfőbb kritérium az, hogy egyik slave egység órajelénél sem lehet nagyobb.
- MOSI – Master Output, Slave Input: A mestert a szolgákkal összekötő sín, melyen az adatirány a mestertől a szolgák felé mutat.
- MISO – Master Input, Slave Output: Gyakorlatilag ez a MOSI ellentéte irányítottság szempontjából. A slavektől a master felé küldött adatok haladnak ezen a vonalon.
- SS – Slave Select. Olykor Chip Selectnek is szokták hívni. Miután egyszerre csak egy eszközzel kommunikálhat a master, ezért kommunikáció előtt ezen a vonalon keresztül jelez az adott szolgának úgy, hogy magas feszültségszintről alacsonyra rántja

az adott slave select kimenetét egészen addig, amíg a kommunikáció tart. Minden szolgának csak egy SS bemenete, míg egy masternek több SS kimenete lehet. A mesterek esetén rendszerint nincs is kitüntetett kimeneti port, míg a szolgák esetén a slave select is egy konkrét lábra van kiosztva.

A kommunikáció az egységek közt bájtos szervezésű és a következő módon történik. Mindegyik eszköz rendelkezik egy nyolc bites shift-regiszterrel (5-3. ábra). Ezek a regiszterek a MOSI-MISO vonalakon keresztül összeköttetésben állnak egymással oly formán, hogy a master regiszterének legkisebb helyiértéke összeköttetésben áll a slavek regiszterének legnagyobb helyiértékével a MISO vonalon keresztül. A MOSI vonal a slavek legkisebb helyiértékét a master legnagyobb helyiértékével kötik össze. Amikor a mester adatot akar küldeni, akkor behelyezi azt ebbe a regiszterbe, majd a kommunikációs órajel által diktált ütembe bitenként jobbra tolja az adatot, így az átkerül a MOSI vonalon keresztül bitenként az adott slave shift-regiszterébe. Ezzel egyidőben a szolga shift-regisztere is megkezdi a bitek eltolását, ezáltal azok a MISO vonalon keresztül a master regiszterébe csorognak át. Gyakorlatilag 8 órajel alatt megtörténik a két regiszter tartalmának a cseréje. Ha a kommunikáció kétirányú, akkor a slave és a master egység is elhelyezi a regiszterbe a küldeni kívánt adatot, majd a küldés ütemében meg is kapják azt. Mikor csak az egyik kíván küldeni, akkor a beérkező adatokat figyelmen kívül hagyja, vagy ha azt a vevő oldal nem módosítja, akkor ellenőrzés céljából a következő küldésnél vissza tudja olvasni az előzőleg elküldött adatait.

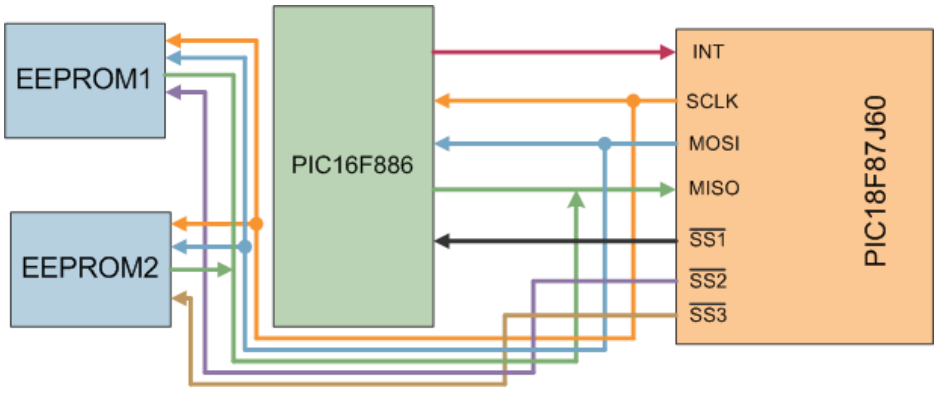


5-2. ábra, A mester és a szolga közt lévő vonalak



5-3. ábra, A shift-regiszterek kapcsolata

Az órajel szempontjából három lényeges dolog befolyásolhatja az adat kiléptetésének és kiolvasásának a módját. Az egyik az órajel frekvenciája. A másik az órajel fázisa, ami azt mutatja meg, hogy mi az alapállapot amitől az órajel generálásakor periodikus ütemben eltér. Ennek alapján lehet az alapállapot magasa feszültség szint, ekkor minden órajelciklusban alacsony szintre esik az érték, valamint ez történhet fordítva is. A harmadik jellemző az eltolás és a kiolvasás fázisa az órajel szempontjából. Ez azt mutatja meg, hogy az órajel felfutó és lefutó élénél melyik esemény következzen be az adott eszközben. Az eszközök összekapcsolása a mi esetünkben a 5-4. ábra szerint valósult meg. Az érdekesség a dologban az, hogy bár a szabvány szerint a master a mi esetünkben a PIC18F87J60-as mikrovezérlő, mégis a kommunikációs mikrokontrollerrel való adatcserét nem ő kezdeményezi. Erre azért van szükség, mert a kommunikációért felelős PIC16F886-os nem tud, csak bizonyos időnként adatcserét folytatni és ez nem történhet akármikor. Nyilván a 87J60-as master nem tudja előre meghatározni, hogy mikor történjen adatcsere, éppen ezért a 886-os mikrovezérlőnek kell neki szólni. Ezt egy megszakítással tudja tudatni a mesternek kinevezett mikrovezérlővel. Amikor a megszakítás bekövetkezik, lebonyolítják a kommunikációt és mindketten folytatják a folyamataikat. Ez azért érdekes, mert ennek tükrében megbomlik a mester-szolga viszony, hiszen a szolga kéri fel a mestert a kommunikációra és nem a mester a szolgát. Létezik egy másik elrendezése az eszközöknek, miszerint a mester és a szolgák egy gyűrűbe vannak felfűzve, tehát egy shift-regiszter hálózatot kell elképzelni, amik sorba vannak kötve egymással. Minden egységhez ugyanaz a slave select jel lesz elvezetve, ezáltal a kommunikációba minden egyes egység részt fog venni. Az adatelem amit a master elküld gyakorlatilag mindegyik szolgán keresztül halad, ezáltal a slave-ek számától függő órajelciklus alatt mindegyik szolgához megérkezik az adott információ.



5-4. ábra, A mikrovezérlők és az EEPROM-ok az SPI kommunikáció viszonylatában

5.4 EUSART

A PIC18F87J60-as mikrovezérlő tartalmaz 2 EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) modult. Az EUSART olyan I/O egység, ami képes sorosan szinkron, vagy aszinkron kommunikációra úgy, hogy annak megfelelő konfigurációjával számos kommunikációs szabványhoz (RS232, RS485) illeszthetővé válik. Mindkét modulhoz tartozik egy-egy Tx (adó) és Rx (vevő) láb, amin keresztül az adást és a vételt tudjuk megvalósítani. Az EUSART modulok minden olyan szükséges elemet és hardveres támogatást tartalmaznak, amik biztosítják azok programtól független működését. Ilyen elemek például a pufferek, shift regiszterek és órajel generátorok. A mi esetünkben full-duplex aszinkron kommunikáció zajlik a központi- és a két hozzá csatlakoztatható modul között. NRZ jelkódolást használunk és egy adategység egy start bitből, 8 adatbitből és egy stop bitből áll. A start bit egy a vonal alapállapotához képest ellentétes állapotú egy bitideig tartó jelzés, ami az adat kezdetét jelöli. Ezzel szemben a stop bit az alapállapottal azonos, egy bitideig tartó jelzés, ami az adatok végét határolják. Abban az esetben, ha két bájtot szeretnénk elküldeni, akkor a bájtokat egy stop és egy start bit határolja. Az egységek 9600 baud-os kommunikációra vannak konfigurálva. A kommunikáció során a küldendő bájtot a mikrovezérlő elhelyezi egy speciális regiszterbe, majd a megfelelő bitek beállításával az USART modulnak jelzi, hogy az adatok küldésre készek. Az USART a beállított baudrate alapján a regiszter megfelelő shiftelésével kipakolja a sínre az adatokat természetesen a start- és a stop bitekkel együtt. A vevő oldalon a start bit megjelenésével a puffer elkezd feltöltődni az adatokkal és ha minden bit megérkezett, az USART modul egy megszakítást vált ki, ezzel az adatok feldolgozásra kerülnek.

5.5 RS485

A termekben lévő egységek és a központ közötti kommunikáció half-duplex, aszinkron, soros elven történik. A nagy távolságok és az elektromosan zajos környezet által keltett zavarok miatt itt már nem elegendő a klasszikus imént tárgyalt soros átvitel. A zavarok által keltett hibákat még csak-csak ki lehetne szűrni valamilyen szinten a megfelelő kódolással, ám a vezeték csillapításából adódó torzulások már nem korrigálhatóak aktív eszköz nélkül. Az ilyen hálózatok működtetése céljából fejlesztették ki az RS485-ös szabványt. Az RS485 egy

Master-Slave viszonyokat feltételező kommunikációs szabvány, ahol az egységek egy A és B vonalból álló sínre vannak. A sín hossza legfeljebb 1200 méter lehet, amin 32 eszköz kommunikálhat egymással, de ez a vonal meghajtásának függvényében változhat. A kommunikáció sebessége 10 méteres távolság esetén megközelítheti a 35 Mb/s-ot is, míg 1200 méter esetén akár még 100 Kb/s-ot is el lehet érni. A kommunikáció megbízhatósága abban rejlik, hogy a 2 vezetékből álló sín nem aszimmetrikus, hanem szimmetrikus meghajtást kap. Ez annyit jelent, hogy a jel nem a föld és a vonalon mérhető feszültségszint különbsége lesz, hanem a két vonalon mérhető potenciálok különbsége. Ha a vonalra zavar ül, akkor az mindkét vezetéken meg fog jelenni, ezáltal a potenciálkülönbség nem fog megváltozni. A kommunikációban a 0 és 1 állapotokat a vonalak polaritásának felcserélésével tudják megvalósítani. Az így létrejövő potenciálkülönbség akár 15V is lehet, így a vezeték csillapítása ellenére is lehetőség válik a nagy távolságon való kommunikáció. Ahhoz, hogy ezt a mi rendszerünkben is megvalósítsuk, szükségünk volt egy olyan áramkörre, ami biztosítja az RS485-ös szabványban leírt fizikai jellemzőket. Ez az áramkör az LTC1487-es vonalmeghajtó IC lett, ami egy nyolc lábú DIP tokban kapott helyet, melynek sematikus felépítése az ábrán látható. A mikrovezérlő USART kimenete (Tx) a meghajtó IC DI lábára a bemenet (Rx) pedig a meghajtó IC RO lábára van kötve. Az IC RE (Receive Enable) és DE(Driver Enable) lábak az adatok fogadását és küldését engedélyező lábak. Miután az RE láb fordított logikával működik, ezért a DE és az RE összekötésével megvalósítható, hogy az adat küldésének és fogadásának engedélyezése egymást kizáróan egy jellel vezérelhető legyen, így a kommunikációban résztvevő mikrovezérlőnek csak egy lábát kell erre a célra lefoglalni. A hibaellenőrzés végett szokás külön vezérelni a két lábat. A mi rendszerünkben az RE mindig engedélyezve van, így adáskor látjuk, hogy mikor ment el a bájt. Az LTC1487-es meghajtó áramkör lehetővé teszi akár 256 egység részvételét a hálózatba, ám a mi esetünkben jelenleg 64 egység van betervezve.

5.6 IEEE 802.3

A PIC18F87J60-as mikrovezérlővel lehetőségünk van az IEEE 802.3 szabványnak megfelelő 10Base-T kommunikációra az ethernet hálózaton. Az előzőekkel ellentétben ez a szabvány sokkal összetettebb és bonyolultabb. Az IEEE 802.3 az ethernet közeghozzáférését (MAC) és fizikai jellemzőit írja le és az OSI modell fizikai rétegét és adatkapcsolati rétegének egy részét képzi. A fizikai réteg első sorban az átviteli közeg tulajdonságait és a jel közegre való

helyezésének módját írja le. A mi esetünkben az átviteli közeg csavart érpár és a maximális átviteli sebesség 10 Mb/s. A bitek közegre való helyezésénél a szabvány által előírt Manchester kódolást alkalmazzuk, aminek az a lényege, hogy a bitek értéke a bitidő felénél bekövetkezett jelváltás irányától függ. Lefutó él esetén a bit értéke „1”, míg felfutó él esetén a bit értéke „0”. A kódolás nagy előnye, hogy a folyamatos jelváltás biztosítja az egységek közti szinkronizációt. Ez az előny egyben hátrányt is jelent, hiszen dupla akkora jelváltási sebességet igényel. Az ethernet keret mérete a mi esetünkben 512 bit – 1518 bájt nagyságú lehet. Minden keret egy 7 bájtos előtaggal rendelkezik, melynek minden bájtja 10101010 mintájú, ezáltal a vevők rá tudnak szinkronizálni az adó által küldött keretre. Az előtagot egy 1 bájtos keret kezdet határoló követi, ezzel jelezvén, hogy honnan kezdődik a tényleges információ. A következő 6 bájton a küldő állomás címe, míg az azt követő 6 bájton a célállomás címe található. Ezt követi két bájt ami a rákövetkező adatmezőben lévő adatok hosszát adja meg. A következő egység maga a hasznos adat. Mivel egy keretnek legalább 64 bájt hosszúnak kell lenni, ezért abban az esetben, ha nincs küldendő adat, a keretet feltöltik, hogy meglegyen a kívánt hossz. A keretet végül a 4 bájtos ciklikus redundancia összeg (CRC) zárja amivel a keret hibamentességét lehet ellenőrizni. Az IEEE802.3 közeghozzáférési algoritmusa a CSMA/CD-re épül. A CSMA (Carrier Sense Multiple Access) lényege, hogy az adó figyeli a kommunikációs csatornát, hogy nincs-e használatban. Ha nincs, akkor megkezdheti az adatok küldését, ha használják, akkor vár addig, amíg fel nem szabadul. A CD (Collosion Detect) az ütközésfigyelést hivatott ellátni. Ha egy adó egy másik adóval együtt teszi a keretét a csatornába, akkor azok ütköznek egymással és egyik vevő részére sem lesz értelmezhető az adat. Ekkor minden egység amelyik adni próbált és érzékelte az ütközést, zavaró jeleket visz a közegre, ezáltal mindegyik eszköz befejezi az adatküldést és véletlenszerű ideig vár és csak ezután próbálja ismét megkezdeni az adást. A várakozási idő a sikertelen kísérletek számának függvényében változik.

A PIC-be integrált ethernet modul gyakorlatilag teljes hardveres támogatást ad az ethernet szinten való közeghozzáféréshez. A modult öt fő funkciót megvalósító egységre lehet felosztani:

- A fizikai adó-vevő modul, ami kódolja és dekódolja a csavart érpáron érkező és az arra kiküldendő adatokat.
- A MAC modul, ami megvalósítja az IEEE802.3 által leírt közeghozzáférési algoritmusokat és közegfüggetlen interfészmenedzsmentet biztosít a fizikai modulhoz.

- Található benne egy mindentől független 8 Kbájtos puffer, ami a kimenő és bejövő adatok tárolására szolgál.
- Egy az ethernet RAM-hoz való hozzáférést szabályzó egység ami a mikrokontroller, a DMA és az adó- és vevő blokkok felől érkező igényeket szolgálja ki.
- Egy regiszter interfész, ami megvalósítja a kapcsolatot a modul és a mikrovezérlő SFR-jei között.

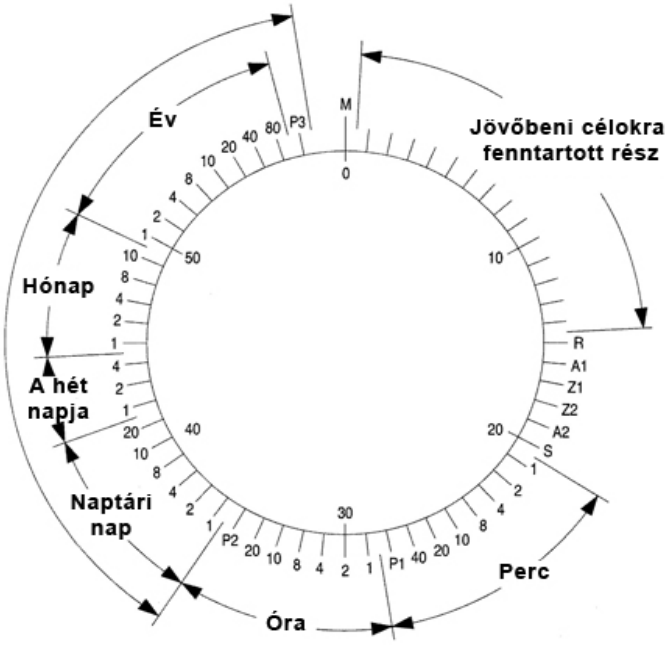
A hardveres támogatás végett szoftveresen csak a MAC réteg felett elhelyezkedő funkcionalitásokat kell implementálni attól függően, hogy milyen kommunikációt szeretnénk megvalósítani.

6 A DCF

Azért, hogy a rendszer mindig a pontos időhöz legyen szinkronizálva, egy DCF nevű szolgáltatás felel. A DCF egy európai lefedettségű szolgáltatás, ami földi sugárzásban hosszúhullámon minden percben elküldi egy atomóra pontos idejét. Ahhoz, hogy ehhez az információhoz hozzájussunk, csupán egy DCF vevőre volt szükség, amit a 886-os kommunikációs mikrovezérlőhöz csatlakoztattunk. A vevő által szolgáltatott időinformációt elküldjük a központi mikrokontorllernek, ami ennek függvényében pontosítja a belső óráját és a dátuminformációit.

A DCF vevő gyakorlatilag nem több egy 77,5 kHz-re hangolt ferritantennánál, és egy, a mikrovezérlő számára könnyen értelmezhető jelet generáló IC-nél és a Frankfurttól 25 km-re lévő mainflingeni DCF adó hosszúhullámon sugárzott jelét fogja. Az adó pontosan 1 perc alatt küldi el az atomórájának idejét, valamint az egyéb azzal kapcsolatos információkat (időzóna, időszámítás, stb.). A 6-1. ábra az adat egy perc leforgása alatt való elküldésének a szekvenciáit ábrázolja. Gyakorlatilag mindegyik másodperc egy bitidőnek felel meg. A nulladik másodperc a jelsorozat indító jele, előtte pedig az 59. másodperc a perc-szinkron, melynek segítségével meg tudják határozni a vevőegységek, hogy az értelmezhető információ honnan kezdődik. Az 1-14 másodperceket a tervezők meghagyták a jövőbeni esetleges igények számára. A 15 -20 másodpercek alatt kapjuk meg az időzónára, a téli-nyári időszámításra és az egyéb szervizkódokra vonatkozó információt. A 21-27 másodpercek alatt kapjuk meg a percidőre vonatkozó információt. Az adat BCD kódolású, ami azt jelenti, hogy

a tízes számrendszerben az egyes helyiértékeknek megfelelő számokat kódoljuk binárisan. Ezzel az eljárással 1 bájton 0-99-ig tárolhatunk értékeket. Azért választották ezt a kódolási módot, mert ha az időt hétszegmenses kijelzőn akarjuk megjeleníteni (márpedig sok esetben van erre szükség pl.: karórák), akkor az így kapott adatok alapján sokkal egyszerűbb azt vezérelni, mintha csak egyszerű bináris tárolást alkalmaznánk. Mivel a percinformáció csak 0-59-ig terjed, ezért nincs szükség, csak 7 bitre a tároláshoz. A 28. másodperc a percidőt reprezentáló 7 bit paritásbitje. A 29-34 másodpercekben az órainformációk kerülnek elküldésre.



6-1. ábra, A DCF információk csoportosítása másodperces bontásban

A kódolási eljárás ugyanaz mint a percek esetén, ám amint az ábrán is láthatjuk itt már csak 6 bitet használnak az adott óra elküldésére. A 35. másodperc itt is paritásbitként szolgál. A 36-41 másodpercek során a hónap napjának a számát, 42-44-ig a hét napját, 45-49-ig a hónap sorszámát, 50-57-ig pedig az adott év utolsó két számjegyének értékét küldik el. Az 58. másodpercben a dátuminformációk paritásbitje kapott helyet. Az adatok amplitúdó moduláció útján továbbítódnak. A jel 2 állapot között váltakozik úgy, hogy ez minden másodpercben megtörténik. Az alacsony állapotba tartózkodás ideje szabja meg, hogy logikai 0, vagy 1 lett-e elküldve. Ha alacsony állapotban 0,1 másodpercig tartózkodik, akkor az 0-t, ha 0,2 másodpercig, akkor az 1-et jelent. Az így elküldött jelet a vevő a mikrovezérlő által megkülönböztethető szintekre alakítja, így azt már fel tudjuk dolgozni. A jelet a központ dolgozza fel, amit a későbbiekben részletesen is tárgyalunk.

7 A TERMEKBE KIHELYEZETT EGYSÉGEK

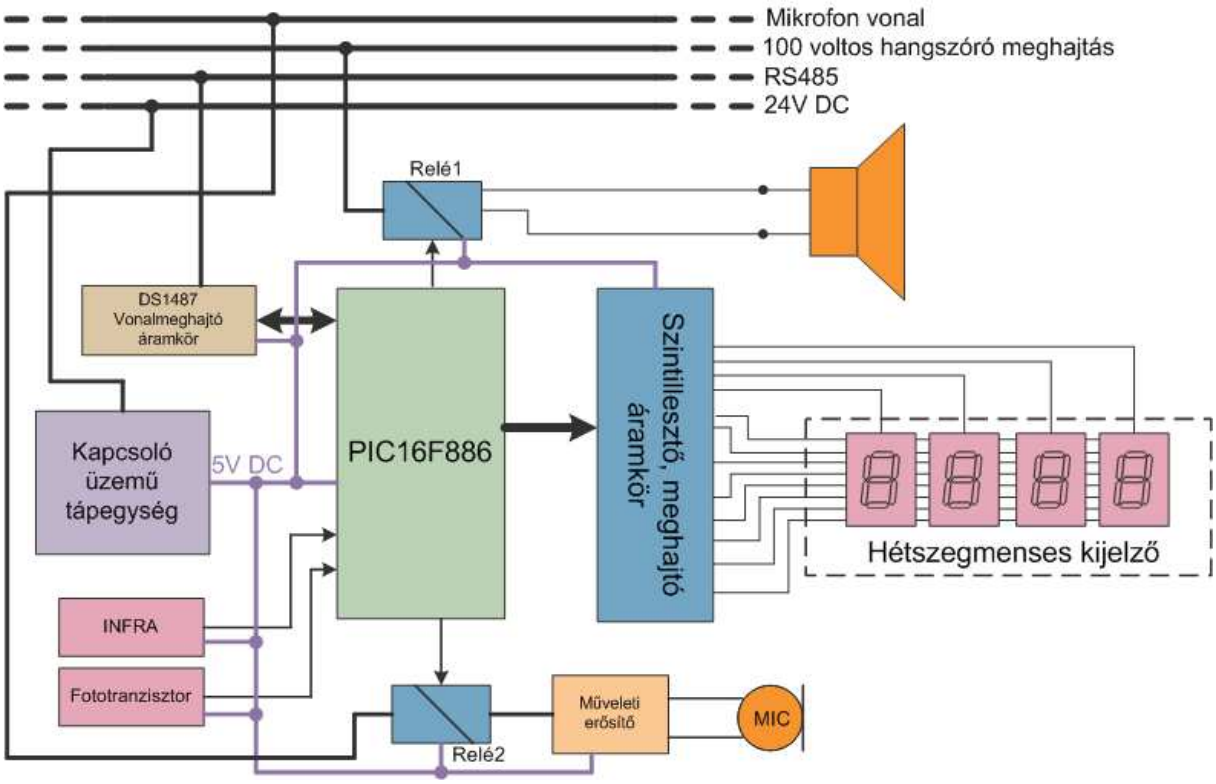
A hangszóróvezérlő egységeknek két típusát különböztetjük meg. Az egyik, ami leginkább a termekben lelhető fel az, ami rendelkezik hétszegmenses kijelzővel. A másik csupán a hangszórók működtetéséért felel. Most a hétszegmenses kijelzővel ellátott verzión keresztül fogom bemutatni ezeket az egységeket, mert az utóbbi gyakorlatilag a kijelzős konstrukció egyszerűsített változata. A modul vezérléséért a központban is megtalálható PIC16F886-os mikrovezérlő felel, ám itt DIP tokozásban van jelen. A mikorkontroller a következő feladatköröket látja el az egységekben:

- Kommunikációs tevékenységet folytat a központtal. Egy vonalmeghajtó áramkör közbeiktatásával az RS485-ös gerincvonalon keresztül tudja tartani a kapcsolatot a központi kommunikációs mikrovezérlővel. A kommunikáció során a központ felől érkező parancs függvényében állítja be és küldi el az egység állapotát.
- A hangszóró vezérlése gyakorlatilag egy relén keresztül történik amit a mikrokontroller vezérel, ezáltal a hangszórót képes teljesen leválasztani, vagy becsatlakoztatni a 100 voltos erősítők által meghajtott vonalra.
- Az egységben helyet kapott egy elektret mikrofon egy műveleti erősítővel, amin keresztül egy külön erre a célra fenntartott árnyékolt vezetéken keresztül jut el a jele a központhoz. Erre az ellenőrzések, hibakeresések végett van szükség. Gyakorlatilag bele lehet hallgatni az egység közvetlen környezetébe, ezáltal ellenőrizhetővé válik a hangszóró állapota. Miután erre az árnyékolt vonalra minden egység mikrofonja csatlakozik, ezért itt is a megfelelő utasítással lehet be illetve kikapcsolni az adott egység vonalra telepedését. A mikrovezérlő egy másik relén keresztül vezérli a mikrofon jelének a központba való eljuttatását.
- Az egyik leglátványosabb feladat a hétszegmenses kijelzők vezérlése. A digitek meghajtása végett a mikrovezérlő és a digitek közé egy meghajtó áramkört iktattunk be, mert a viszonylag nagy kijelző túlságosan nagy áramot vesz fel. A megjelenítés a multiplex kijelzés elvén került megvalósításra. Ennek az a lényege, hogy egyszerre mindig csak egy digitre küldünk információt és mindig csak egy rövid ideig, majd lépünk a következőre és így tovább. Ha elég nagy sebességgel váltunk a digitek között, akkor az a szemnek úgy tűnhet mintha azok folyamatosan mutatnák az adott információt. Ha megnézzük a rajzon a bekötést, láthatjuk hogy mindegyik digit egy 7+1 vezetékből álló sínre van felfűzve és mindegyikhez külön-külön még tartozik egy-egy vezeték. A multiplexelés úgy történik, hogy a digitek azonos szegmenseinek az anódja rá van kötve a 7+1 vezetékköteg megfelelő vezetékére és digitenként mindegyik katód arra a plusz egy vezetékre van kötve, ami mindegyik digithez külön-külön el van vezetve. A 7+1 széles buszon kiküldjük a megjeleníteni kívánt információt, az egyedi vezetékeken pedig kiválasztjuk, hogy melyik digiten szeretnénk azt megjeleníteni. Ha megfelelő sebességgel változtatjuk a digitre kitett információt

és a digitek állapotát, akkor a szemnek úgy tűnhet, hogy mindegyik digit folyamatosan működik és a saját információját jeleníti meg.

- A 886-oshoz csatlakozik még egy fototranzisztor és egy infra vevő is. A fototranzisztor a környezet fényét méri és ennek függvényében a mikrovezérlő a multiplex kijelzés sebességének szabályozásával változtatja a hétszegmenses kijelző fényerejét. Tehát éjszaka halványabban, nappal pedig erősebben világítanak a szegmensek. Ez takarékossági szempontból előnyös. Az infra vevő a távirányítók jelét képes a mikrovezérlő számára is értelmezhető jellé alakítani. Attól függően, hogy melyik gombot nyomtuk meg a távvezérlőn, a mikrovezérlő a kapott parancsot vagy feldolgozza, vagy pedig a központnak továbbítja.

A modulok a tápellátásukat a 24 voltos tápvonalról nyerik, amiből egy kapcsoló üzemű tápegység segítségével állítják elő a rajtuk lévő eszközök által igényelt stabil 5 voltot.



7-1. ábra, A termekbe kihelyezett egységek sematikus blokkvázlata

8 ADATSZERKEZETEK ÉS TÁROLÁSUK

8.1 A központon lévő EEPROM-ok

A rendszer működéséhez szükséges adatokat a központi modulon lévő, egyenként 128 kilobájt kapacitású EEPROM-okban tároljuk. A memóriákat SPI-n keresztül tudjuk elérni, 3 bájton tudjuk megcímezni és az írására mind bájt, mind pedig lap szinten lehetőségünk van. Egy lap 256 bájt méretű, tehát összesen 512 darab van belőle. Laponkénti írás esetén ügyelnünk kell arra, hogy egy írási ciklusban ne szaladjunk túl az adott lap határán, mert akkor nem következő lap elejét kezdi írni, hanem az aktuális lap kezdőcímétől fogja folytatni az írást, ezzel felülírva az ott lévő adatokat. Ezt a problémt szoftveresen kell menedzselni a program futása során. A különböző műveletek végrehajtásának menete mindig egy 1 bájtos parancs kiadásával kezdődik, majd ezt követi a 3 bájtos cím. A 3 bájt több mint elég a címek számára, éppen ezért az EEPROM a 24 bit utolsó 7 bitjét gyakorlatilag figyelmen kívül hagyja. Ezeket a biteket „don't care" biteknek nevezzük. Azért, hogy a 2 EEPROM egy összefüggő memóriaterületként viselkedjen, egy olyan eljárás felel, ami a címbájtok első „don't care" bitje alapján (ami a mi esetünkben a 18. lesz) fogja eldönteni, hogy melyik EEPROM-ot fogja kiválasztani az SPI kommunikációhoz. Eleinte az első don't care bit nulla lesz, ekkor az adott címinformáció alapján megcímezzük a kiválasztott EEPROM-ot. Abban a pillanatban, hogy nagyobb címet adunk meg, mint amekkora az EEPROM mérete, az első don't care címbit 1-be billen és ezt észlelvén az SPI kommunikáció során a másik EEPROM kerül kiválasztásra, ami ismét csak a számára lényeges címinformációt fogja feldolgozni. Fontos, hogy ez a megoldás csak abban az esetben működik, ha az EEPROM-ok mérete, és kommunikációs protokollja megegyezik. A memória írása esetén az adatok a címinformáció után következnek, olvasás során pedig a parancs és a cím elküldése után folyamatosan kapjuk őket a memóriától.

8.2 Adatszerkezetek

Az EEPROM-ok három fajta információ tárolását végzik a rendszerben:

- csengetési információk
- a hangszórók állapotára vonatkozó információk
- a rendszerre vonatkozó eseményinformációk

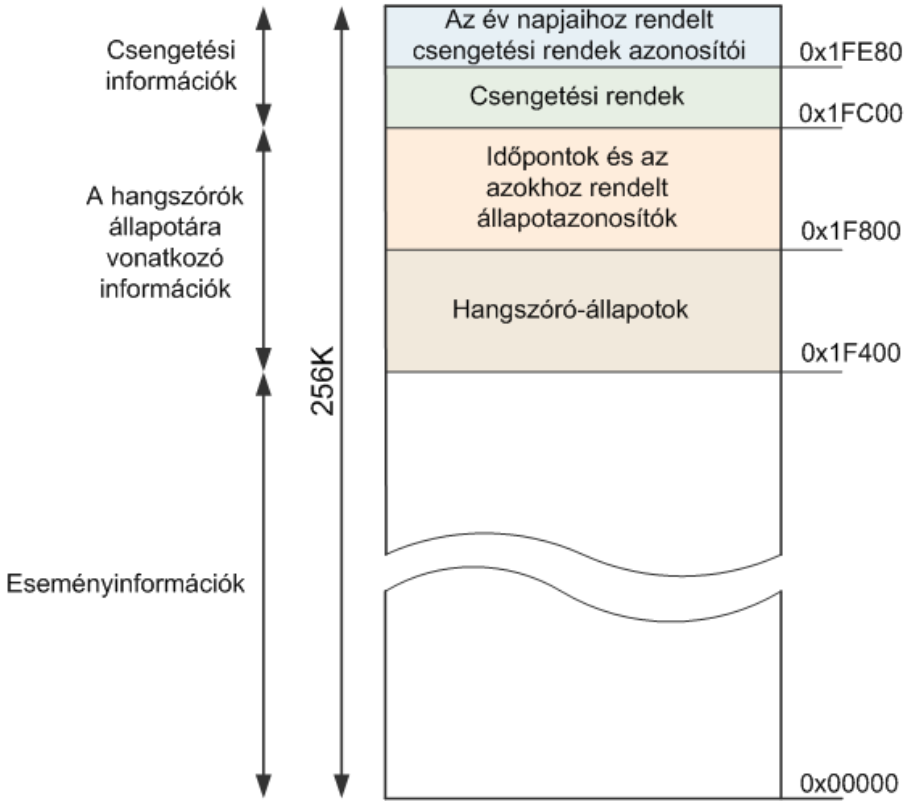
A csengetési információk a 2 memória által alkotott közös memóriaterület utolsó 1 kilobájtjában az 0x1FC00 címtől kezdődően foglalnak helyet. A terület első 640 bájtja 10 darab, egyenként 64 bájt nagyságú csengetési rend tárolására nyújt lehetőséget. Egy tanítási nap legfeljebb 8 tanóra lehet, és mindegyik tanítási órához 4 időpont tartozik. Egy időpont órából és percből áll, amit BCD kódolásban tárolunk a DCF által szolgáltatott időformátum végett. Ez azt jelenti, hogy ha egy időpont 2 bájt, akkor egy tanítási órához 8 bájt tartozik. Nem feltétlenül igaz, hogy minden nap mind a 8 órában kell csengetni, de olyan nem lehet, hogy a tanítási nap közepén ki kell hagyni a csengetést. Ha az adott csengetési rend nem rendel mind a 8 órához csengetést, akkor azoknak az időpontoknak a helyét, ahol nem kell csengetni 0xFE értékkel töltjük fel. Ezáltal gyakorlatilag biztosítjuk, hogy mindegyik csengetési rend ugyanakkora helyet foglal el függetlenül attól, hogy hány órán kell csengetni és ezzel kiszámíthatóvá válik az adott csengetési rend kezdőcíme. A csengetési rendek folytonosan helyezkednek el az 0x1FC00 címtől kezdve egészen a 0x1FE80-as címig. Abban az esetben, ha ez a 640 bájtos terület nincs teljesen kihasználva (tehát nincs 10 darab csengetési rend), akkor a fennmaradó területet 0xFF értékkel töltjük fel. A 0x1FE80 cím feletti maradék terület az aktuális év napjainak van fenntartva, hogy hozzájuk tudjuk rendelni az aktuális csengetési rendet. A tervezés során minden hónaphoz 31 napot rendeltünk függetlenül attól, hogy ténylegesen hány napot tartalmaz. Azt, hogy az adott napon melyik csengetési rendet akarjuk használni úgy állítjuk be, hogy az 1. egyenlet alapján található címre beírjuk az adott csengetési rend sorszámát. Abban az esetben, ha nulla értéket írunk arra a bájtra, akkor az adott napon nem lesz csengetés, egyébként 1-10-ig akármelyik létező csengetési rend használható.

1. egyenlet

$$célcím = 0x1FE80 + (aktuális\_hónap\_sorszáma \times 31) + a\_hónap\_napjának\_sorszáma$$

A hangszórókra vonatkozó állapotinformációkat a csengetési információk alatti két kilobájton tároljuk. 0x1F401 címtől kezdve legfeljebb 127 darab a hangszórók által felvehető állapotvariációt tárolhatunk. Egy állapot 8 bájtot (64 bitet) jelent, ahol mindegyik bit egy-egy hangszórót reprezentál. A tartomány első bájtja a 0x1F400-as címen azt mutatja meg, hogy hány darab állapot van jelenleg a rendszerbe. A következő 0x1F800-as címen kezdődő kilobájton pedig az egyes állapotok időponthoz való rendelése található meg időrendben. Ha a hónap, nap, óra, perc és állapot adatokhoz egy-egy bájtot rendelünk, akkor egy bejegyzés összesen 5 bájtot jelent. Ilyen öt bájtos bejegyzésekből legfeljebb 200 darabot tudunk tárolni. Minden általunk megadott időponthoz meg tudjuk adni, hogy a letárolt állapotok közül hanyadik tartozzon.

A 0x1F400 alatti terület a rendszerben bekövetkezett események tárolására van fenntartva. Egy év minden napjára több mint 600 bájt adatmennyiség jut. Jelenleg még nem tart ott a fejlesztés, hogy a rendszer eseményei folyamatos rögzítés alatt legyenek, éppen ezért még a tárolási elvről sincs elképzelés, de valószínűleg ez is folytonosan történne.



8-1. ábra, Az EEPROM-ok által biztosított memóriaterület felosztása

9 A KÖZPONTON LÉVŐ PIC16F886-OS MIKROVEZÉRLŐ FOLYMATAI

9.1 A mikrovezérlőben futó folyamatok

A PIC16F886 kommunikációs mikrovezérlőnek jelenleg három fő funkciót kell ellátnia:

- Le kell bonyolítania termekbe és folyosókra telepített egységekkel való kommunikációs tevékenységeket az RS485-ös vonalon keresztül.
- Be kell gyűjtenie és fel kell dolgoznia a DCF által szolgáltatott időinformációt.
- Kommunikációs tevékenységet kell folytatnia a központi PIC18F87j60-as mikrovezérlővel.

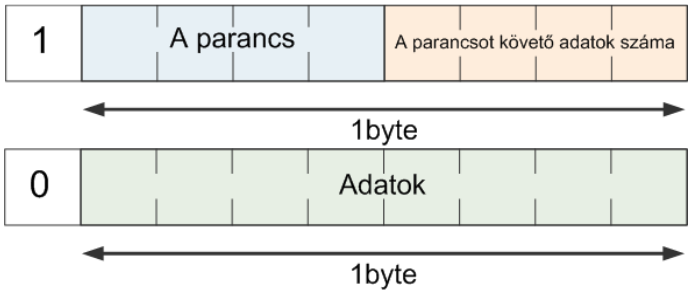
A mikrovezérlő a folyamatokat azok jellegétől függően két szinten futtatja. Alapvetően a 3 fő funkció a főprogram végtelenített ciklusában hajtódik végre, ám ezeknek vannak olyan eseményei amik időkritikusságuk miatt nem várhatnak addig, amíg a végrehajtás során rájuk nem kerül a sor. Ezeket a feladatokat megszakításokkal kezeljük annak függvényében, hogy mikor kell őket végrehajtani. A megszakítások lekezelésének sorrendje azoknak a megszakításvektorban elhelyezett sorrendtől függ. A főprogram alapján először a 485-ös vonalon keresztül történő kommunikációs rutinok hajtódnak végre. Az adatok küldése és feldolgozása a főprogramban, míg a fogadás az adat érkezésénél kiváltódó megszakítás lekezelése folyamán történik. A DCF-ből érkező impulzus(sorozat) kiértékelése egy a mikrovezérlő időzítőjéhez kötött 200µs időközönként bekövetkező megszakítás alkalmával hajtódik végre. Ha az egy perces DCF ciklusban minden adat megérkezett, akkor egy jelzőbit segítségével tudathatjuk az időinformáció fogadásának a sikerességét a főprogramban található feldolgozó eljárással. Az eljárás a letárolt adatot több szempontból is megvizsgálja és ha helyesnek ítéli, akkor a központnak továbbítható formátumúvá alakítja (pl. az első 14 bitet, vagy a paritásbiteket elhagyja). Abban az esetben ha ez nem teljesül, akkor a kapott értéket eldobja. Gyakran előfordult, hogy a paritásbitek nem bizonyultak eléggé megbízható ellenőrzési lehetőségnek, mert a páros számú hibákat nem lehetett velük kimutatni, ezért ahhoz, hogy elfogadjunk egy időadatot, ahhoz 3 egybehangzó jó vételre van szükség. Amikor lezajlott a 485-ös kommunikáció és a DCF lekezelése, egy megszakítással felkéri a 886-os a

központi mikrovezérlőt, hogy kezdeményezzen vele SPI kommunikációt. Ezen kommunikáció folyamán cserél adatot a két mikrokontroller.

9.2 Az RS485-ös vonalon történő kommunikáció protokollja

A kommunikációs mikrovezérlő és a terembe kihelyezett egységek közti kommunikáció protokollja a slave-master viszonyokon alapul. Az egész lényege az, hogy a kommunikációt mint mester mindig a 886-os központi mikrokontroller kezdeményezi és ennek függvényében reagálnak a slave egységek. A kommunikációs mikrovezérlő oldaláról az adategységek az EUSART által is támogatott 9 bites szervezésben kerülnek a 485-ös vonalra a slave egységek számára. Két fajta adattípust különböztetünk meg:

- Ha parancsot akarunk küldeni, akkor a 9. bitet egybe állítjuk. A maradék egy bájt 2 részre oszlik, ahogy a 9-1. ábra is mutatja. Az első 4 bitje a bájtnak maga a parancs kódja, a második 4 bit pedig azon adatcsomagok számát mutatja meg amik a parancsot követik. Ennek alapján legfeljebb 16 fajta parancsot tudunk kiadni az egységek számára amiket maximum 15 bájt követhet. Abban az esetben, ha csak egy egységgel szeretnénk kommunikálni akkor közölni kell a sínen lévő állomásokkal, hogy a parancs és az esetlegesen hozzá tartozó adatok kinek szóltak. Ebben az esetben a parancs után következő csomag a címinformációt tartalmazza. Azt, hogy a parancsot fogja-e cím követni, minden egység el tudja dönteni a parancs alapján.
- A másik csomagtípus az, ami az egységek számára feldolgozható adatokat valamint esetlegesen címet tartalmaz. Ebben az esetben a 9. bit értéke nulla, a maradék egy bájton pedig vagy cím, vagy pedig az adott parancshoz tartozó adatok találhatók.



9-1. ábra, A parancs- és az adatcsomagok felépítése

Mindegyik az RS485-ös vonalra csatlakozó slave egységnek létezik egy egyedi azonosítója. Abban az esetben, ha a központon lévő kommunikációs mikrovezérlő csak egy egységgel szeretne kommunikálni, akkor a parancs kiadása után elküldi annak a címét. Olyan eset nem lehetséges, hogy egy parancs vonatkozhat egyszerre több egységre és akár egyre is.

Minden parancsnál előre meg van határozva, hogy fogja-e követni cím, vagy sem. Abban az esetben, ha az adott parancshoz nem tartozik cím, akkor a parancsot és az utána következő esetleges adathalmazt mindegyik egységnek fel kell dolgoznia. Ha a parancs jellege megköveteli a címet, akkor a végrehajtás előtt az egységek megvárják, hogy kire vonatkozik az utasítás. Nyilvánvaló, hogy a szolgáknak tudniuk kell, hogy melyik parancshoz várjanak címet és melyiket hajtsák végre feltétel nélkül. Az, hogy milyen parancsokkal fog üzemelni a rendszer, még nincs teljesen letisztázva, de azokat amik már véglegesítve vannak a 9-2. ábra tartalmazza.

Parancs	A parancsot követő adat
Hangszóró bekapcsolása	címinformáció
Hangszóró kikapcsolása	címinformáció
Mikrofon bekapcsolása	címinformáció
Mikrofon kikapcsolása	címinformáció
Kijelzés	szegmensadat, amit a hétszegmenses kijelzőkön kell megjeleníteni
Igénybejelentés	nincs adat
Az összes hangszóró állapotának megváltoztatása	8 bájtos adatcsomag, amibe mindegyik bit egy hangszórót reprezentál
Egyéb beállítás	címinformáció + vezérlőbitek, melyek az adott egység egyes funkcióit tudják beállítani

9-2. ábra, A parancsok és a hozzájuk tartozó adat

Ugyan a fali egységek egyedi statikus címmel rendelkeznek, azok regisztrálása a rendszerbe dinamikusan történik. A kommunikációs mikrokontrollernek létezik egy igény bejelentésre felszólító parancsa. Abban az esetben, ha valaki új tagként kapcsolódott a rendszerbe, megvárja ezt a parancsot és utána közli a központtal, hogy innentől kezdve ő is részt kíván venni a kommunikációba. Ez az elképzelés nagyszerűen működik akkor, ha egyszerre mindig csak egy egység akar bejelentkezni a hálózatba. De mi történik akkor, ha éppen kettő, vagy akár az összes egység be akar jelentkezni az adott pillanatban[2]? Hogyan fogja tudni a központ, hogy ki akarta benyújtani a felvételi kérelmét, ha egyszerre többen is ráülnek a kommunikációs csatornára? Erre egy nagyon szellemes, magasabb szinten talán nem is realizálható megoldás lett foganatosítva. A dolog lényege, hogy a felszólító parancs kiadása után a központ kommunikációs mikrokontrollere a slave egységekkel együtt kikapcsolja az usartot, ezáltal a Tx és az Rx lábak egyszerű I/O portokként fognak viselkedni. A központ a Tx lábat 1-be rántja, majd elengedi és mint bemenetet elkezdi figyelni. Ha egy egység várt csak az igény bejelentésének lehetőségére, akkor bitenként elkezdi kipakolni a

[2] Akkor fordulhat elő, hogy mindegyik egység be akar jelentkezni, amikor például a rendszer újraindul.

vonalra a címét reprezentáló bájtot, és mindig figyeli a vonalat, ha az adott bit amit ki kell pakolnia 0. A központi 886-os mikrovezérlő a Tx lábán begyűjti a 8 bitet és felveszi az új címet a listára. Abban az esetben, ha egyszerre több egység próbálkozik a bejelentkezéssel, akkor mindegyik elkezdi bitenként a vonalra pakolni a címét. Abban a pillanatban amikor a két vagy több egység által kitett bit nem egyezik meg azok az egységek mind leállnak akik 0 értékű bitet akartak volna kipakolni, hiszen a vonal figyelése közben érzékelték, hogy valaki más is be akar jelentkezni. Ez a folyamat a 8 bit kipakolása végéig tart és végül a bejelentkezésre várók között a legnagyobb című egység lesz az aki a 8. bitet is elküldi. Ezzel az eljárással biztosítva van, hogy mindegyik bejelentkezési körbe a bejelentkezésre várók közül a legnagyobb című egység iratkozhat fel. Az igénybejelentésre felszólító parancs a főciklusban található, tehát néhány másodperc alatt akár mind a 64 állomás regisztrációja megtörténhet.

10 A PIC18F87J60-AS MIKROVEZÉRLŐ FOLYAMATAI

Mint azt a korábbiakban már többször is említettem, a rendszer magját a központon elhelyezett PIC18F87J60-as mikrovezérlő és a benne futó folyamatok alkotják. A Microchip az ilyen, ethertnet modullal ellátott mikrovezérlőit első sorban olyan alkalmazások fejlesztésére és működtetésére alkotta meg, amik igénylik az etherneten és azon keresztül akár a világhálón való kommunikációt. Ilyen alkalmazási terület lehet például a biztonságtechnika vagy az épületautomatizálás. Miután a mikrovezérlő az IEEE 802.3 szabványban leírtakhoz hardveres támogatást nyújt, nekünk már „csak” az általunk használni kívánt OSI-rétegig kell szoftveresen implementálnunk a protokoll vermet. Ez így első hallásra egyszerűnek tűnik, de ha utána nézünk, láthatjuk hogy csupán egy szint megírása mennyi nehézséget jelentene. Ezt a problémát valószínűleg a Microchipnél is felismerték és nem sokkal az első ilyen mikrovezérlők megjelenése után ingyenesen elérhetővé tették a saját fejlesztésű protokoll vermüket a fejlesztők számára. Ez egy általánosan megírt protocoll stack, ami több mikrovezérlő családot is támogat. Tulajdonképpen ez egy „.c” és „.h” forrásállományokból álló struktúra, ami az alapvető protokollok implementálásán túl számos egyéb funkciót is megvalósít. Ilyen például az applikációs rétegben futó szolgáltatások, az egyéb soros kommunikációt megvalósító modulok, vagy az erre a célra fejlesztett demo-kitek támogatása.

Gyakorlatilag a fejlesztés során mi is ezt az általuk kiadott forrást vettük alapul és egészítettük ki, vagy éppen töröltük a számunkra felesleges funkciókat.

A mikrovezérlő által jelenleg ellátott tevékenységeket a következők szerint csoportosíthatjuk:

- Funkcionális tevékenységek
  - Az idő mérése
  - A belső óra szinkronizálása a DCF-től érkező jelsorozat alapján
  - Az idő és az aktuális csengetési rend függvényében a csengetés jelzése
  - Az aktuális hangszóróállapot beállítása az idő függvényében
- Kommunikációs tevékenységek
  - Kommunikáció a 886-os mikrovezérlővel
  - Kommunikáció az EEPROM-okkal
  - Kommunikáció az USART-on keresztül az egyéb eszközökkel
  - Kommunikáció az ETHERNET-en

A továbbiakban vizsgáljuk meg a teljesség igénye nélkül, hogy milyen fontosabb folyamatok zajlanak le a központi mikrovezérlőben a rendszer működése során.

10.1 Funkcionális tevékenységek

Nyilvánvaló, hogy az imént felsoroltak között sok időkritikus és véletlenszerűen bekövetkező esemény található. A rendszer működése szempontjából minden folyamatnak meg van a maga prioritása és azt, hogy ezek zavartalanul működhessenek egymás mellett, a megszakítási szinteken való megfelelő elhelyezésükkel és jelző flagekkel valósítottuk meg. A Microchip ezen mikrovezérlője, ahogy azt már korábban is említettem két megszakítási szinttel, egy magas- és egy alacsony prioritásúval rendelkezik. Az alacsony prioritású megszakítási szinten a protokoll stack által használt megszakítások, valamint a timer_1[3] időzítő által generált megszakítás, míg a magas prioritású szinten a kommunikációs mikrovezérlő által generált megszakíás található.

Azért, hogy a pontos idő szolgáltatás megfelelően működjön a rendszerbe, a központnak elő kell állítania egy megfelelően pontos időalapot, amit aztán majd a DCF-hez fog szinkronizálni. Ez úgy került megvalósításra, hogy a timer_1 időzítőtől 10 millisecundumonként kérünk egy megszakítást. A megszakítás során egy változó értékét inkrementáljuk egészen addig, amíg el nem éri az értéke a 100-at. Amikor ez megtörténi a

[3] A mikrovezérlő belső hardveres számlálója, amely minden túlcsorduláskor képes megszakítást generálni

másodperceket tartalmazó változó értékét növeljük eggyel és nullázzuk a 10 millisec.-es időket mérő változó értékét. A másodpercekből származtatjuk a perceket, azokból pedig az órákat és így tovább. Amikor az adatmemóriában frissül a kommunikációs mikrovezérlő által elküldött DCF információ, akkor annak alapján ellenőrizzük és pontatlanság esetén korrigáljuk az időt tartalmazó változók értékét. Az így előállított időinformáció alapján történik az időkijelzés vezérlése, valamint a csengetések- és a hangszórók állapotának a menedzselése. Az időkijelzéshez a főprogram előállítja a megfelelő szegmensinformációt a pontos idő alapján, majd a 886-os mikrovezérlővel történő kommunikáció során elküldeti az állomásoknak.

A csengetés vezérlése egy a főprogramban végrehajtódó eljárással történik, ami csak percváltás esetén fut le. Az eljárás először lekérdezi az EEPROM-ból az adott naphoz tartozó csengetési rendet. Ezek után elkezdi végignézni az időinformációkat és számon tartja azt is, hogy az adott időhöz melyik csengetést kell lejátszania. Ezt nagyon egyszerű megtennie. hiszen a 4 dallam [4] egymás után ismétlődik az időinformáció alapján. Abban az esetben, ha egyezést talál a belső óra és valamelyik bejegyzés alapján, az USART Tx regiszterébe elhelyezi a lejátszandó dallam sorszámát. Az USART modul hardveresen kiküldi az információt a dallamgenerátornak, ahol kiválasztván a dallamot, lejátssza a 100 voltos erősítőkön keresztül.

A hangszórók állapotának beállítására két lehetőség van. Az egyik egy ideiglenes állapot megteremtése, egy a PC oldali programok felől érkező parancs és állapotinformáció segítségével. Ez a megoldás a hirdetések alkalmával hasznos, amikor esetlegesen az aktuális állapottól eltérő állapotot szeretnénk produkálni (mondjuk csak egy terembe akarunk beszólni). A hirdetés megkezdése előtt az etherneten át elküldjük a központnak az ideiglenes állapot megteremtésére való felszólítást és az állapotot reprezentáló 64 bitet. A hirdetés megkezdésekor a központi mikrovezérlő leellenőrzi, hogy nem érkezett–e ideiglenes állapotkérelem. Ha érkezett, akkor a hirdetések idejéig a kommunikációs mikrovezérlőn keresztül beállítja az adott állapotot a hangszórókon. Ha nincs ilyen jellegű igény, akkor nem történik semmi. A másik az EEPROM-ban tárolt információ alapján történik. A főprogramban minden percváltáskor lefut egy eljárás amiben az EEPROM-ban növekvő sorrendben tárolt időinformációk és az aktuális idő kerül összehasonlításra. Az aktuális idő alapján megvizsgálja, hogy melyik az az időpont ami már elmúlt és melyik az ami még nem

[4] A becsengetés előtt 2 perccel megszólaló, a becsengetést jelző, a kicsengetés előtt 2 perccel megszólaló, a kicsengetést jelző

következett be. Minden egyes a memóriába tárolt időpont átlépése esetén lekéri az időponthoz tartozó hangszóró-állapotot. Minden állapotváltozás esetén az új állapotot a 886-os mikrovezérlőn keresztül kiküldi a hangszórókat vezérlő állomásoknak. Abban az esetben, ha 00:00 óráig nem történik meg az alapállapotba [5] való visszaállás, a központ erről automatikusan rendelkezik.

10.2 Kommunikációs tevékenységek

A kommunikációs tevékenységek egyik fő jellemzője, hogy nem lehet előre meghatározni, hogy mikor következnek be. Éppen ezért egy olyan megoldáshoz kell folyamodni amivel úgy tudjuk kezelni őket, hogy se egymást se a főprogramot ne zavarják a szabályos működésben. A mikrovezérlő által jelenleg ellátott tevékenységek csoportosítása alapján érdemes megvizsgálni, hogy az egyes kommunikációs tevékenységek ténylegesen hogyan is szerveződnek. Az ethernet felől a TCP szinten legfeljebb 4 felhasználó részéről érkezhetnek adatok a mikrovezérlő saját memóriájába vagy pedig valamelyik EEPROM-ba, vagy igények ugyanezen adattárolókban található adatok kinyerésére. A Microchip által implementált protocoll stack tartalmaz egy multiuseres kiszolgálást megvalósító TCP servert, ami egy eljárás formájában a főprogramban fut. Maga az eljárás a protocoll stack alacsonyabb szinten működő eljárásaira építve biztosítja a különböző kliensek kiszolgálását. Jóformán itt már csak fel kell építeni a megfelelő socketen keresztül a kapcsolatot és le kell bonyolítani a kliensekkel való kommunikációt. A server IP címe statikus úton lett beállítva, ami a mi esetünkbe azt jelenti, hogy a program forráskódjába lett megadva. A kezdeményezésmindig a kliensek kezében van olyformán, hogy elküldik a megfelelő parancs és címinformációt, valamint ha írásról van szó az adatokat. A PC-k felől a következő igények érkezhetnek a rendszer irányába:

- Adatok olvasása az EEPROM-ból + cím
- Adatok írása az EEPROM-ba + cím + adat
- Adatok olvasása a mikrovezérlő belső memóriájából + cím
- Adatok írása a mikrovezérlő belső memóriájába + cím
- Adatok blokkos írása az EEPROM-ba + cím + adat

[5] Alapállapotnak az számít, amikor mindegyik hangszóró be van kapcsolva.

A blokkos írás során a server mindig küld egy válaszüzenetet a kapott csomag után (ez a többi írási mód esetén elmarad). Ennek segítségével, ha a kliensek irányából egy nagyobb memóriaterületet, vagy eltérő címen elhelyezkedőket akarunk egyszerre írni, akkor ez mint egy szabályzó hat a kliensben pörgő ciklusra, hogy ne küldjön addig új csomagot, amíg az előzőt fel nem dolgozta a server. A többi parancs és hozzá tartozó adat magától értetődik. Ha a mikrovezérlő adatmemóriáját szeretnénk írni vagy olvasni, akkor nincs gond, hiszen azt közvetlenül meg tudjuk tenni, viszont az EEPROM-ok esetén már más a helyzet.

A mikrovezérlő az EEPROM-ok és a 886-os mikrovezérlő elérését SPI buszon keresztül biztosítja a folyamatok és az etherneten keresztül igényt küldő kliensek számára. Az SPI tárgyalásánál láthattuk, hogy egyszerre csak egy egységgel tud a master kommunikálni annak függvényében, hogy melyiket választotta ki. Látható, hogy szükség van egy irányelvre amivel meg lehet szervezni a különböző kommunikációs kérelmek lekezelését. Az irányelv alapja a kommunikációs mikrovezérlő lett, miszerint az csak akkor tud adatot cserélni a központi egységgel, amikor befejezte a tantermi és folyosói állomásokkal való kommunikációt, valamint a DCF-információ feldolgozását. Azért, hogy ez biztosítva legyen a 886-os számára, az SPI kommunikáció az általa kiváltott megszakítással fog elkezdődni. Amikor a megszakítás bekövetkezik, a központi mikrovezérlő tudja, hogy eljött az alkalom, hogy lebonyolítsa az adatcserét a 886-ossal. Masterként kiválasztja a kommunikációs mikrokontrollert és kicserélik egymás közt a vezérlési- és státuszadatokat. A kommunikáció végeztével a központ megvizsgálja, hogy valamelyik folyamata, vagy egy az ethernetről érkező igény nem kíván-e az EEPROM-okkal adatot cserélni. A vizsgálat a következő sorrend alapján hajtódik végre:

1. Akar-e a központ olvasni az EEPROM-ból?
2. Akar-e a központ írni az EEPROM-ba?
3. Akar-e valamelyik kliens olvasni az EEPROM-ból?
4. Akar-e valamelyik kliens írni az EEPROM-ba?

Ha valamelyik jelzőbit be van állítva, akkor az adott igénynek megtörténik a kommunikációs kiszolgálása. Ahhoz, hogy ez így működjön, szükség van egy átmeneti tárolóra, ahova a folyamatok, vagy a kliensek lepakolhatják az igényeiket és a hozzájuk tartozó adatokat (pl . címinformáció, beírandó adat), addig amíg a kiszolgálásra várakoznak. Ez a puffer egy 512 bájt nagyságú memóriaterület a mikrovezérlő adatmemóriájában. Az

olvasási igény esetén a kiolvasott adat is ide kerül és innen kerül majd a folyamatok által feldolgozásra, vagy az etherneten keresztül a kliens programokhoz.

Az USART-on való kommunikációt egyenlőre csak a csengetésért felelős eljárás használja, ráadásul az is csak adatküldésre. Miután az USART hardveresen támogatott, ezért csak annyi az eljárás feladata, hogy az USART Tx regiszterébe helyezi az elküldeni kívánt adatot. Az adat küldése ezek után minden folyamattól függetlenül az USART modul által fog megtörténni.

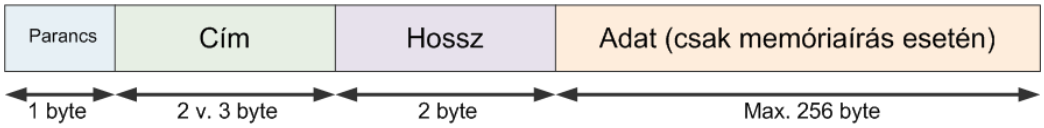
11 FELHASZNÁLÓI ALKALMAZÁSOK

A rendszerhez tartozik négy, annak menedzselésére szolgáló PC oldali kliensprogram, melyből 3 a rendszer különböző szolgáltatásainak beállítására, egy pedig az adminisztrációs és karbantartási műveletek elvégzésére készült. Mindegyik alkalmazás a Microsoft .NET keretrendszerre C# programozási nyelven íródott. Az alkalmazások gyakorlatilag képesek hozzáférni a központi mikrovezérlő adatmemóriájához és az EEPROM-okhoz képesek hozzáférni az etherneten keresztül. Ez a szabadság azonban veszélyekkel járhat, ha például a memóriák írása esetén nem vagyunk elég körültekintőek és felülírunk egy olyan memóriatartalmat ami a helyes működés szempontjából értékes adatot tárolt. Az ilyen esetek elkerülésére mindegyik alkalmazás a maga módján van felkészítve amit az adott alkalmazás részletes tárgyalásakor be is mutatok.

Alapvetően mindegyik kliens azonos protokollt használ a kommunikációra. A kapcsolat kiépítésében egy a .Net System.Net.Socket osztályából származtatott TimeOutSocket osztály működik közre, aminek egyik nagy erénye, hogy a példányosítás során meg lehet adni egy időkorlátot millisecundumban. Abban az esetben, ha a kapcsolat kiépítése annak megkezdésétől az időkorlát által meghatározott idő alatt nem történik meg, akkor a kapcsolódási szándék egy kivétel dobásával megszűnik. Ezáltal, ha a server nem tud fogadni klienst, vagy egyszerűen nem érhető el, nem fog kifagyni az alkalmazás. A megoldás hátránya, hogy arról nem kapunk információt, hogy a server foglaltsága miatt nem tudunk csatlakozni, vagy ténylegesen elérhetetlenné vált a központ. A TCP kommunikáció során a kliens alkalmazásoktól a server felé küldött csomagok adatrészének a felépítése az ábrán

látható módon szerveződik. Az első bájt mindig egy parancs, ami lehet írási, vagy olvasási kérelem. Attól függően, az EEPROM-ra vagy a belső memóriára irányul-e a kérelem a parancs után az előbbi esetén a parancsot egy 3 bájtos, az utóbbi esetén pedig egy 2 bájtos cím követi. A címinformáció után két bájton elküldjük, hogy mennyi adatot kívánunk kiolvasni, illetve beírni. Olvasás esetén legfeljebb 512 bájtot kérhetünk egyszerre, mert a mikrovezérlőben lévő átmeneti puffer mérete is ekkora. Az EEPROM írása maximálisan 256 bájtos adagokban történhet, de ez is csak akkor, ha laphatártól kezdjük az írást. Ha nem így történik akkor egy adagban legfeljebb csak az adott címtől a következő laphatárig írhatunk. A laphatárok a memóriában a 256 egész számú többszörösei értékeinek megfelelő címeken találhatóak. Olvasás esetén a parancs, cím és darabszám elküldése után a server visszaküldi a kívánt adatot. Írás esetén az adatok elküldése után nem érkezik megerősítés a fogadás sikerességéről. Ez azonban nem elég abban az esetben amikor egyszerre több címre akarunk írni, mert a következő adag küldése esetén még nem tudjuk, hogy feldolgozta-e már az igényt a központ. Erre az esetre létezik egy speciális írási parancs, ami minden feldolgozott csomag után visszaküld egy 1 bájtos megerősítést. Ez a parancs csak az EEPROM-ok írásánál használható.

A következőkben vizsgáljuk meg az egyes alkalmazások funkcióit és működésük lényegét. Az alkalmazások képernyőmaszkjait megtalálhatjuk a dolgozat végén lévő függelékben.



11-1. ábra, A kliensek által küldött csomagok szerkezete

11.1 Az adminisztrációs alkalmazás

Az adminisztrációs program gyakorlatilag a rendszer felügyeletére és akarbantartási műveletek elvégzésére készült. Ez az egyetlen olyan alkalmazás amelyen keresztül akár az EEPROM-ok, akár a belső memória bármelyik részéhez hozzá tudunk férni. A program indításakor automatikusan megpróbál csatlakozni a szerverhez és ha ez nem sikerül neki, akkor hibaüzenettel kilép. A szoftver elsődleges funkciója, hogy az általunk megadott

címekről az általunk megadott adatmennyiséget, az általunk kiválasztott memóriába tudjuk írni, vagy éppenséggel abból olvasni. Természetesen a server által megszabott korlátok itt is jelen vannak, tehát 512 bájtnál több adatot egyszerre nem olvashatunk és legfeljebb 256 bájtot írhatunk attól függően, hogy az EEPROM laphatárán kezdjük-e, vagy sem. A program írásakor a belső memória kezelése esetén meghagytam ezeket a korlátokat, mert jellemzően egy-két bájt figyelésén, vagy beállításán túl (pl. időváltozók) nem megyünk tovább, tehát nem volt indokolt a nagyobb adathalmazok kezelését biztosítani. Az EEPROM-ok esetén már más volt a helyzet. Itt az írási és olvasási korlát egy körben 512 bájt. Az olvasási mechanizmus tehát nem változott, ám írás esetén a 256 bájtnál több adat legalább 2 memórialapot jelent, ráadásul ha nem laphatáron kezdjük, akkor kapásból három részre tagolódik, hiszen legfeljebb csak laphatárig írhatunk egy körben. Nyilván nem várhatjuk el a felhasználótól, hogy mindig azt számolgassa, hogy mikor következik laphatár és, hogy mennyi adatot küldhet el. Ezt a dolgot szoftveresen kezeljük le úgy, hogy a maximum 512 bájt méretű adathalmazt a megfelelő darabolásban küldjük el, annak függvényében, hogy hol kezdjük az írást. A programban helyet kapott egy olyan funkció is, amellyel a csengetési és hangszóró információkat tartalmazó 3 kilobájtot kilobájtonként ki tudjuk olvasni.

Lehetőségünk van az egyes memóriaterületek tartalmának a fájlba mentésére, valamint a visszaírására. Erre mentési célokból van szükség és itt az adatmennyiség felső korlátja a 2 bájton elküldhető legnagyobb érték. Az adatok darabolását és több ciklusban való küldését/fogadását itt is a kliens szervezi meg a cím és a méret függvényében.

Az alkalmazásban található egy csengetési rend inicializáló modul. Alapvetően ha nincs semmi kivételes alkalom, akkor a következő elv szerint szerveződnek a tanítási napok: hétfőtől szerdáig, valamint pénteken 8 db 45 perces tanítási óra van megtartva, viszont csütörtökön a 7. óra az utolsó és hétvégén nincs csengetés. Abban, hogy ne nekünk kelljen egyesével minden napra előre beállítani a csengetési rendeket, a csengetési rend inicializálásának a lehetősége nyújt segítséget. Az inicializálandó tanév megadása után a program legenerálja a csengetési rendekről szóló információt tartalmazó 1 kilobájtot úgy, hogy az tartalmazni fogja a két csengetési rendet, valamint az általunk beállított tanévre vonatkozó beosztásukat. Természetesen ezzel az adott memóriatartalom felülíródik.

11.2 A csengetési rendeket szervező alkalmazás

A csengetési rendek egyszerű menedzselését egy külön program teszi lehetővé a számunkra. Az alkalmazás lényege az, hogy a háttérben történő műveletek, ismerete nélkül tudjuk a rendszer csengetésre vonatkozó paramétereit beállítani, tehát egy laikus felhasználó is képes a használatára. Az alkalmazás elindítása és a szerverhez való sikeres csatlakozás után letöltődik az EEPROM csengetési információt tartalmazó 1 kilobájt méretű adathalmaza és ennek alapján a programban létrejön két adatstruktúra. Az egyik a csengetési rendeket, a másik pedig az év napjaira vonatkozó beosztást tartalmazza. Minden módosító művelet során megtörténik az adott struktúra frissítése és generálunk egy olyan bájthalmazt, amelyet elküldve a központnak, ott is végrehajtódik az adott változtatás. A program által a következő műveleteket tudjuk elvégezni:

- Új csengetési rend felvétele abban az esetben, ha még nem létezik, vagy a csengetési rendek száma nem érte el a tízet.
- Csengetési rend hozzárendelése az adott naphoz.
- Csengetési rend törlése az adott napról.

A program főablaka egy táblázatot, valamint egy naptárat tartalmaz. Az alkalmazás indulásakor a táblázat az adott napra vonatkozó csengetési rendet mutatja a megfelelő háttérszínnel, ugyanis minden, a rendszerben lévő csengetési rendhez tartozik egy szín. Erre azért van szükség, hogy a naptárban a napok színezésével áttekinthetőbbé váljon az adott hónap, hogy mikor melyik csengetési rend alapján történik a csengetés. A naptárban a napokra kattintva megjelenik a táblázatban az adott napra vonatkozó csengetési rend. Ha a kijelölt dátumra új csengetési rendet akarunk beállítani, akkor lehetőségünk van erre a „Csengetési rend beállítása a kijelölt naphoz” opcióval. Ekkor a csengetési rendeket mutató táblázat alatt lehetőségünk van a meglévő rendek közti böngészésre és a csengetési rend beállítására, vagy szimplán a csengetés törlésére az adott napra vonatkozóan. Abban az esetben, ha a rendszer nem tartalmazza az általunk használni kívánt csengetési rendet, lehetőségünk van új csengetési rend hozzáadására. Ezt a „Csengetési rend hozzáadása” opcióval megnyitott ablakban tudjuk megtenni. Ha valami esemény miatt egy speciális csengetési rend van érvényben (rövidített órákat kell tartani egy megemlékezés miatt), akkor azt előre ki szokták tenni az iskolai hirdetőtáblákra. Annak érdekében, hogy az adott napra vonatkozó csengetési rendet ne kelljen begépelni a program tartalmaz egy „Csengetési rend nyomtatása” funkciót amivel a táblázatban lévő csengetési rendet tudjuk kinyomtatni

megformázott alakban. A nyomtatás előtt lehetőségünk van a csengetési rend okára vonatkozó tájékoztató jellegű információk megadására. Ezek az információk a nyomtatás során csatolódnak a dokumentumhoz. Minden módosítási műveletről, azok idejéről és az őket alkalmazó felhasználóról eseménybejegyzés készül az iskolai serveren található, ezen alkalmazáshoz tartozó eseménynaplóba. Erre azért van szükség, hogy rendellenes működés esetén vissza lehessen keresni, hogy mikor és milyen beállításokat hajtottunk végre a rendszeren.

11.3 A hangszórók állapotát menedzselő alkalmazások

A termekben és folyosókon lévő hangszórók általunk meghatározott időpontokban bekövetkező állapotváltozásainak menedzselését is egy erre a célra fejlesztett alkalmazás segítségével tudjuk gyorsan és egyszerűen elvégezni. Erre azért van szükség, mert lehetnek olyan alkalmak (érettségi, értekezlet) amikor nem szerencsés, ha a csengetés, vagy a hirdetés mindenütt megszólal. Ezzel a programmal lehetőségünk van különböző hangszóróállapotok adott időpontokhoz való beállítására. Az alkalmazás kommunikációs működése megegyezik a csengetési rendeket menedzselő programnál tárgyalt mechanizmussal, annyi különbséggel, hogy itt az adatstruktúrák a hangszóró információkat tartalmazó 2 kilobájt méretű EEPROM-ban tárolt adathalmazból konstruálódnak. A program főablakában az iskola felülnézeti sematikus képe látható szintenkénti bontásban. A rajzon minden hangszórót egy kis négyzet jelképez, melynek kitöltési színe az adott hangszóró állapotától függően lehet zöld (aktív), vagy piros (inaktív). Az ábrától balra található egy naptár, amin azon napok háttérszíne módosul, ahol van állapotváltásra vonatkozó időpontbejegyzés. Egy ilyen napra kattintva megjelennek a naphoz tartozó időpontok, melyek közül bármelyikre kattintva megnézhetjük az ábrán, hogy milyen hangszóróállapotok tartoznak hozzájuk. Az időpontlista alatt lehetőségünk van új időpont felvételére az általunk megadott napra, de ehhez csak olyan állapot rendelhető, ami már létezik a rendszerben. Abban az esetben, ha még nem létezik számunkra megfelelő állapotvariáció, lehetőségünk van új állapotot létrehozni, vagy akár létező állapotot törölni, ha már nincs rá szükség. Az állapot létrehozása során lehetőségünk van egyenként (az adott négyzetre kattintva) de akár szintenként, vagy globálisan megváltoztatni a hangszórók aktív-passzív tulajdonságát. A bejegyzések törlésére is van lehetőség, az adott bejegyzés kijelölése után a „A kijelölt bejegyzés törlése” gombra kattintva. A program használata során ebben az esetben is történik eseménynaplózás.

A negyedik alkalmazás az utóbbinak egy egyszerűsített verziója. Ebben a programban nem történik meg a hangszórókra vonatkozó állapotinformáció lekérdezése. Nincs lehetőség több állapot kezelésére, se a napokhoz rendelésére. Ez az alkalmazás egész egyszerűen arra jó, hogy egy ideiglenes állapotot állítsunk be a rendszeren. A főablak ebben az esetben csak az iskola sematikus felülnézeti ábráját és a hangszórókat szimbolizáló, két állapotú négyzeteket tartalmazza. Ha beállítunk egy állapotot és elküldjük a központnak, akkor ebben az esetben az állapot a központi mikrovezérlő belső memóriájának egy erre fenntartott területére íródik, valamint bebillent egy jelzőflaget. A hirdetés alkalmával a jelzőflag állapotától függően fogja eldönteni a mikrovezérlő, hogy az eredetileg beállított, vagy pedig az újonnan leküldött ideiglenes állapot alapján szólaljanak-e meg a hangszórók. Erre az alkalmazásra azért volt szükség, mert a hirdetések alkalmával olykor felül kell bírálni az akkorra beállított hangszóróállapotokat. Ilyen eset lehet, ha például csak egy terembe szeretnénk beszólni.

12 Összefoglalás

A rendszer fejlesztése során nagyon sok mindent megtanultam, de mindenek előtt azt, hogy igaz az, amit a kezdő síelőknek mondani szoktak: *„Egy őszinte mozdulat a lejtő irányába és menni fog!”* Érdekes volt látni azt, hogy a szabványok, a protokollok, a folyamatkezelés, a memóriakezelés és még sok egyéb dolog, amikkel eddig legfeljebb csak elméleti szinten vagy vizsgán találkozhattam, hogyan válnak teljesen gyakorlati építőelemekké. A fejlesztés során természetesen gyakran ütköztünk nehézségekbe, de igazából ezek az akadályok voltak azok, amiken keresztül megismerhettem és megérthettem a mikorvezérlők és az egyes kommunikációs szabványok működését a kellő mélységekig. A rendszer jelenleg még nem tölti be az összes tervezett feladatát, de folyamatosan fejlesztjük és javítjuk az esetlegesen felmerülő hibákat. Néha ugyan előfordul, hogy elmarad egy becsengetés mert egy rosszul megszervezett ciklusban elszabadul a pointer, de erre azt mondom, hogy ezek az apró dolgok teszik a fejlesztők munkáját és a diákok napjait színessé.

13 KÖSZÖNETNYILVÁNÍTÁS

Szeretnék köszönetet mondani Jászay Gábor külső témavezetőmnek és volt informatika tanáromnak, aki végig segített és támogatott a diplomamunka elkészítése során. Öröm volt vele dolgozni és mindig hálás szívvel fogok visszatekinteni erre a néhány hónapra. Továbbá szeretném megköszönni a Lévay József Református Gimnáziumnak, hogy lehetőséget biztosított számomra a munka elvégzésére. Meg szeretném köszönni dr. Almási Béla témavezetőmnek, hogy elfogadta és végigvihettem nála ezt a szakdolgozati témát. Végül, de nem utolsó sorban köszönöm Istennek, hogy ennek az elvégzését is megadta a számomra.

„De akik az Úrban bíznak, erejük megújul, szárnyra kelnek, mint a sasok, futnak, és nem lankadnak meg, járnak, és nem fáradnak el.

Ézs.: 40:31

14 IRODALOMJEGYZÉK

14.1 Hivatkozások

- SPI

  http://www.freescale.com/files/microcontrollers/doc/ref_manual/S12SPIV3.pdf
  http://www.maxim-ic.com/app-notes/index.mvp/id/3947

- RS485

  http://www.bb-elec.com/bb-elec/literature/tech/485appnote.pdf

- EEPROM

  http://ww1.microchip.com/downloads/en/devicedoc/21836a.pdf

- RS485-vonalmeghajtó áramkör

  http://1100f.free.fr/LTC1487.pdf

- PIC16F886-os mikrokontroller

  http://ww1.microchip.com/downloads/en/DeviceDoc/41291F.pdf

- PIC18F87J60-as mikrokontroller

  http://ww1.microchip.com/downloads/en/DeviceDoc/39762e.pdf

- DCF

  http://www.ptb.de/en/org/4/44/442/dcf77_1_e.htm

14.2 Felhasznált irodalom

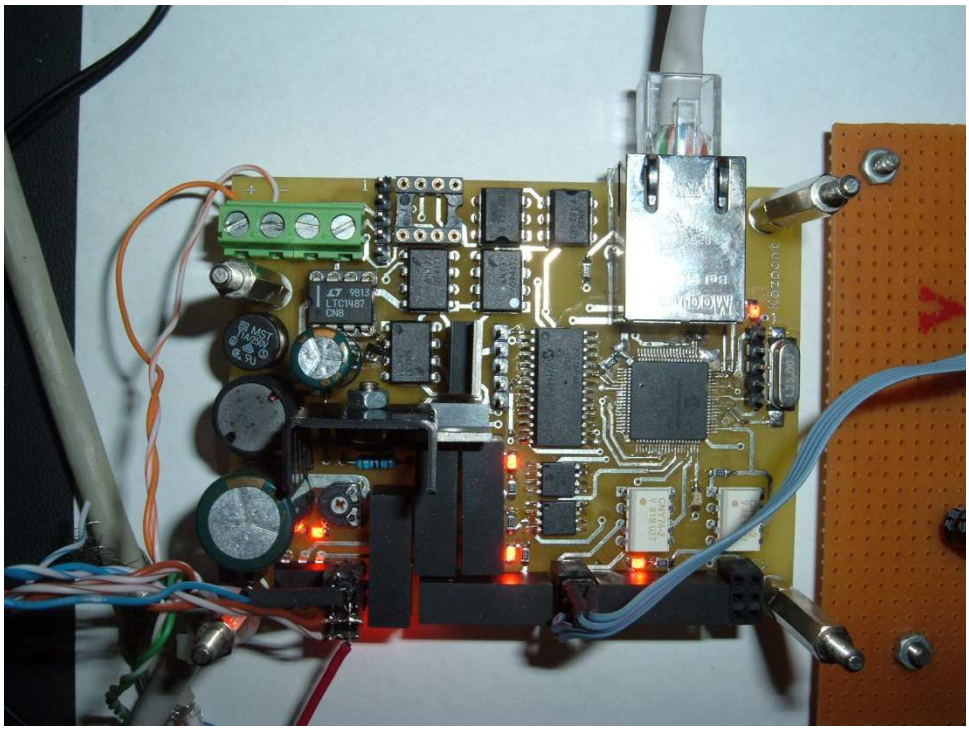
dr. Kónya László: PIC mikrovezérlők alkalmazástechnikája, 2000 ChipCad Elektronika Kft.

Andrew S. Tanenbaum: Számítógép hálózatok, 2004 PANEM

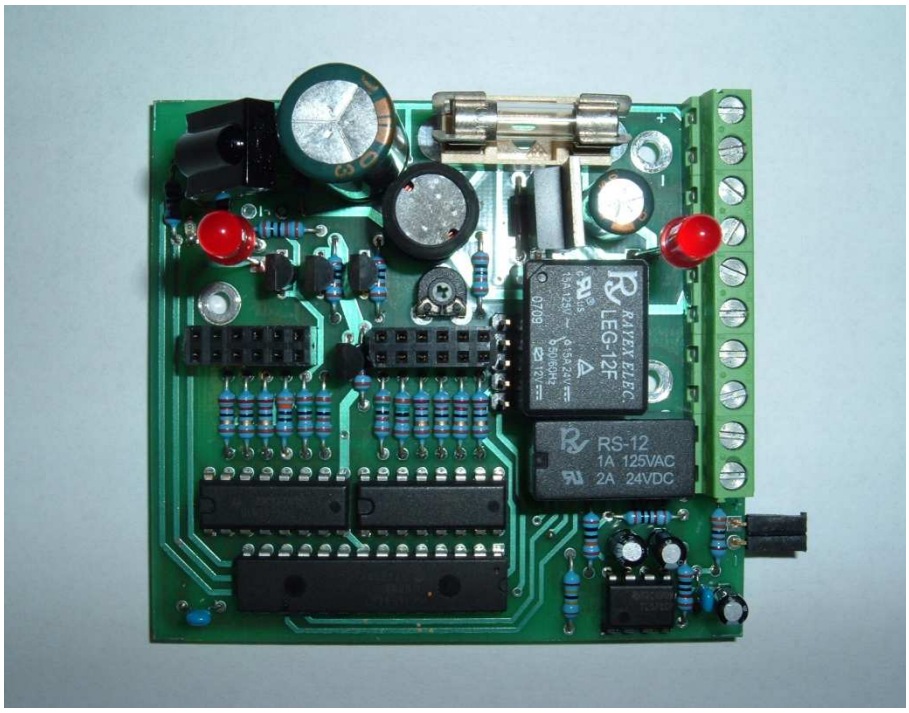
Kónya László – Kopják József: PIC mikrovezérlők alkalmazástechnikája – PIC Programozás C nyelven, 2009 ChipChad Elektronika Kft.

15 FÜGGELÉK

15.1 A rendszer egységeiről készült fotók



15-1. ábra, A központi modul

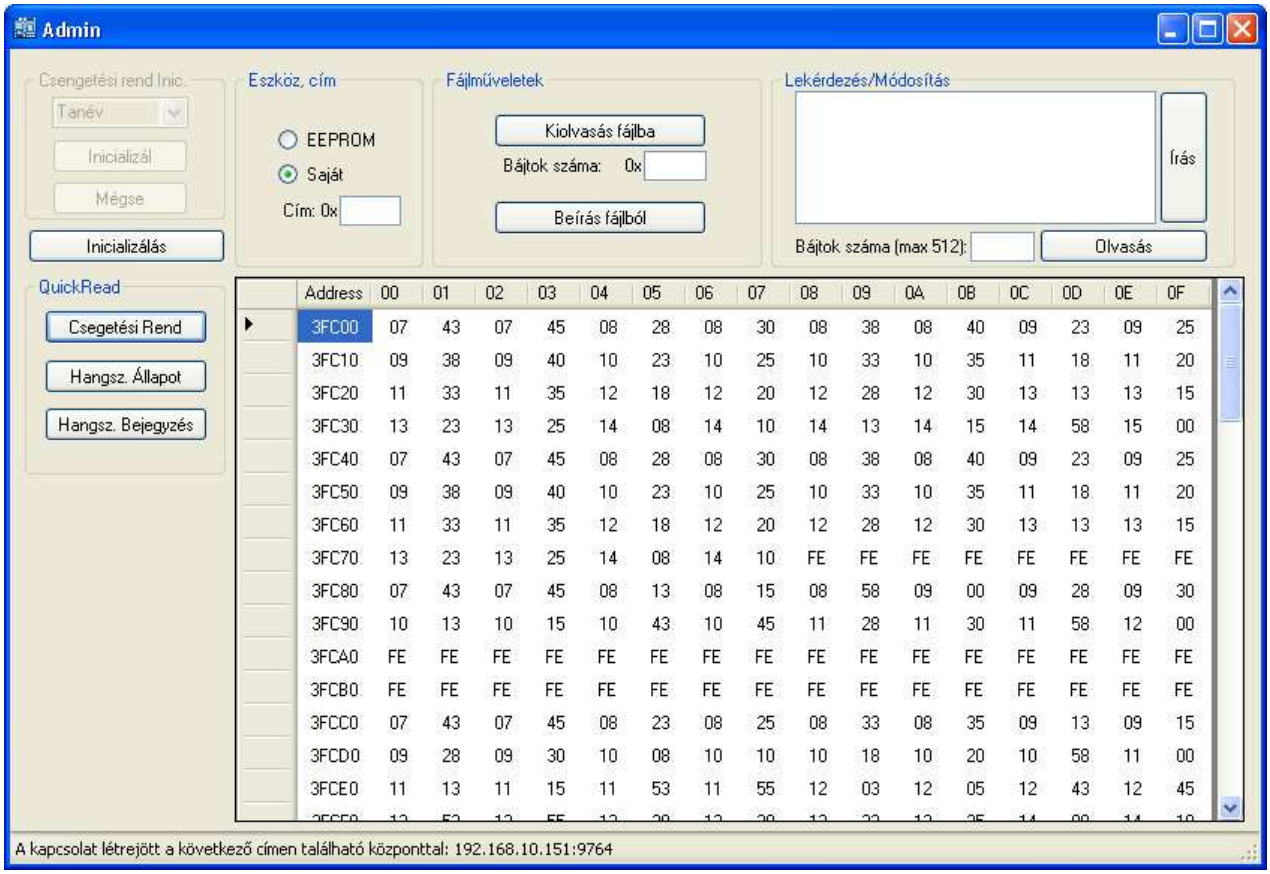


15-2. ábra, A kiejlzővel ellátott állomások központi modulja

15-3. ábra, Egy tantermi, kijelzővel ellátott egység szétszedett állapotban

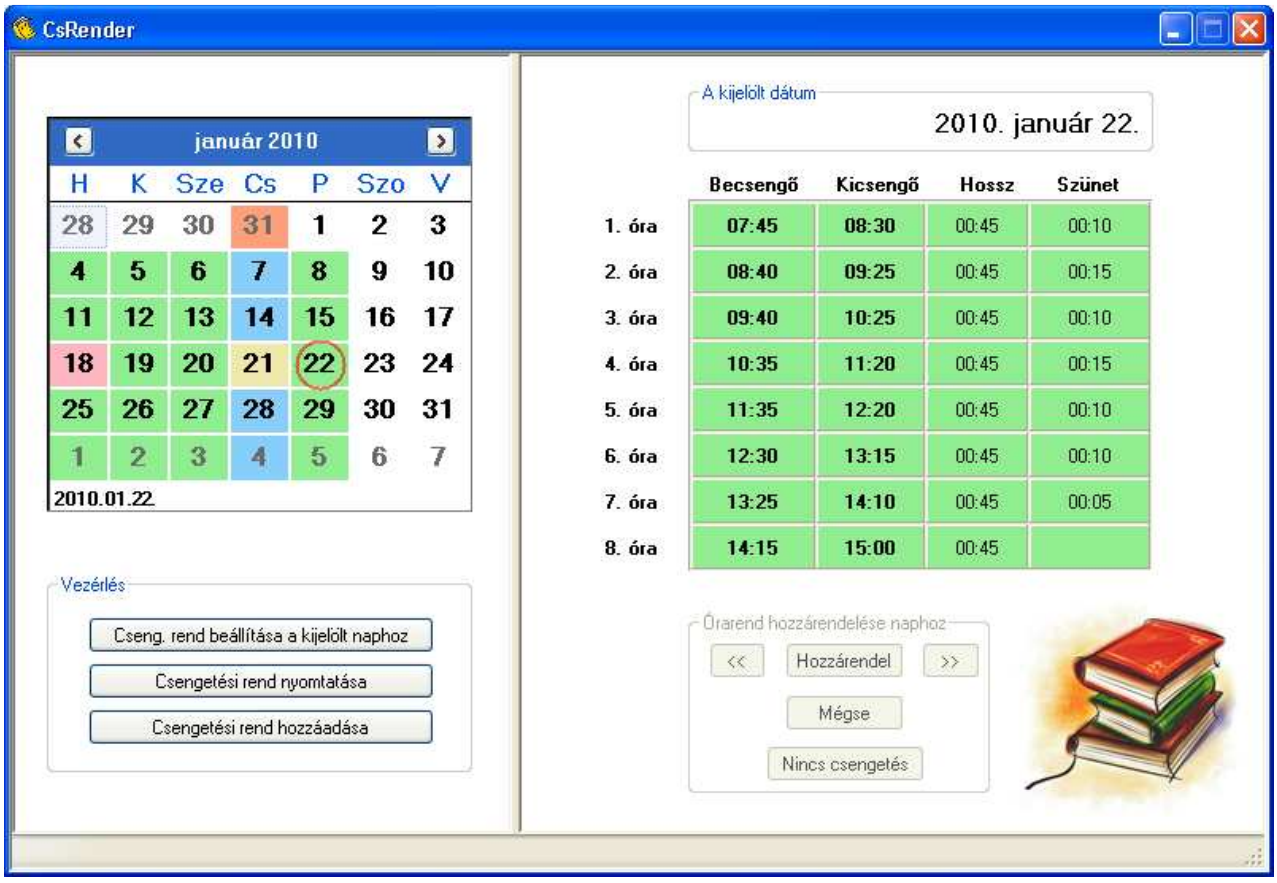
15-4. ábra, Egy tanterembe telepített egység

15.2 A PC-oldali kliensekről készült képernyőmaszkok

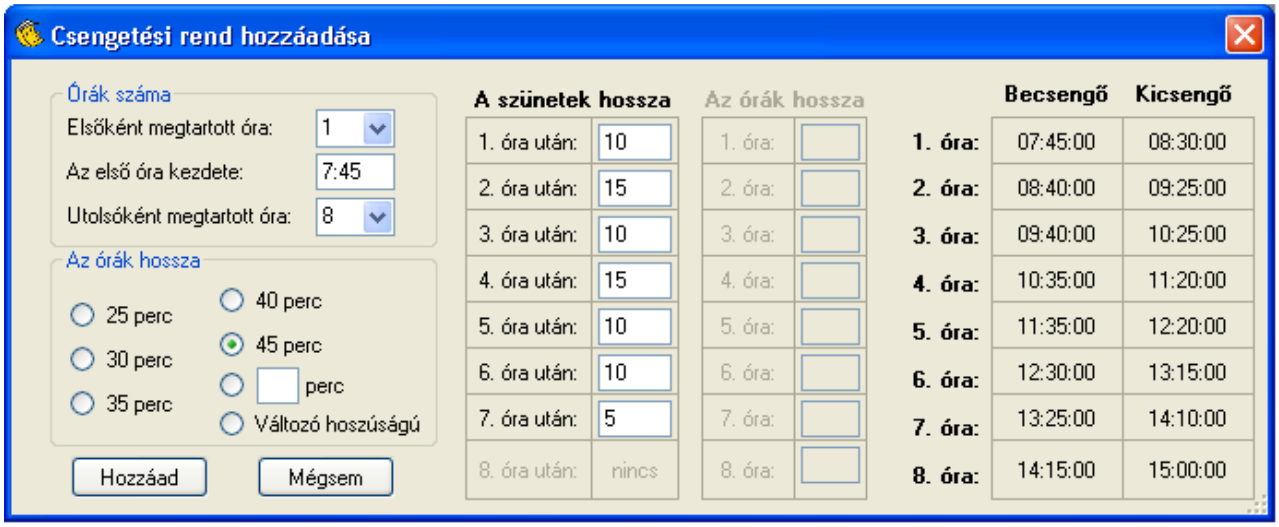


Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
3FC00	07	43	07	45	08	28	08	30	08	38	08	40	09	23	09	25
3FC10	09	38	09	40	10	23	10	25	10	33	10	35	11	18	11	20
3FC20	11	33	11	35	12	18	12	20	12	28	12	30	13	13	13	15
3FC30	13	23	13	25	14	08	14	10	14	13	14	15	14	58	15	00
3FC40	07	43	07	45	08	28	08	30	08	38	08	40	09	23	09	25
3FC50	09	38	09	40	10	23	10	25	10	33	10	35	11	18	11	20
3FC60	11	33	11	35	12	18	12	20	12	28	12	30	13	13	13	15
3FC70	13	23	13	25	14	08	14	10	FE	FE	FE	FE	FE	FE	FE	FE
3FC80	07	43	07	45	08	13	08	15	08	58	09	00	09	28	09	30
3FC90	10	13	10	15	10	43	10	45	11	28	11	30	11	58	12	00
3FCA0	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE
3FCB0	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE
3FCC0	07	43	07	45	08	23	08	25	08	33	08	35	09	13	09	15
3FCD0	09	28	09	30	10	08	10	10	10	18	10	20	10	58	11	00
3FCE0	11	13	11	15	11	53	11	55	12	03	12	05	12	43	12	45

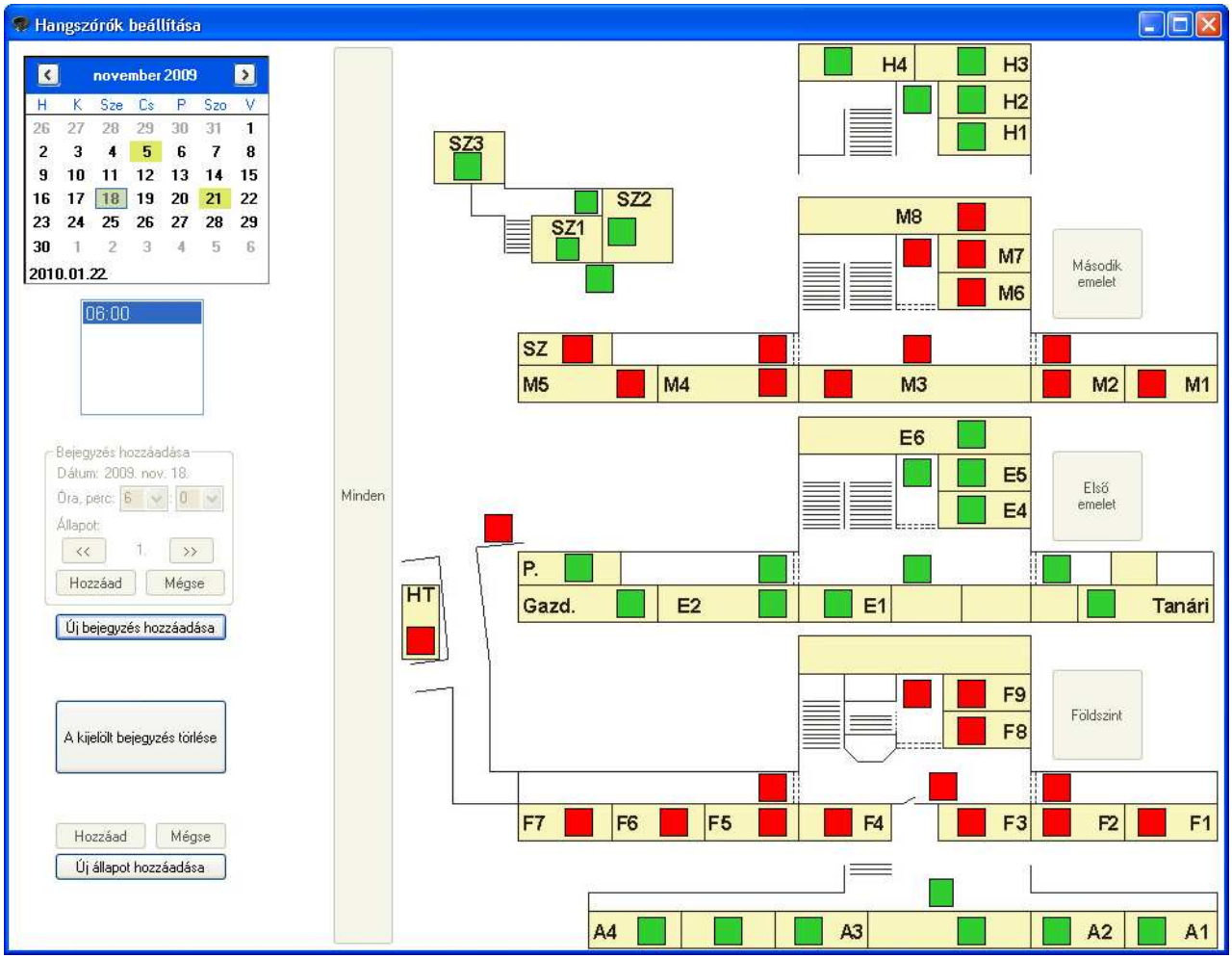
15-5. ábra, Az adminisztrációs alkalmazás



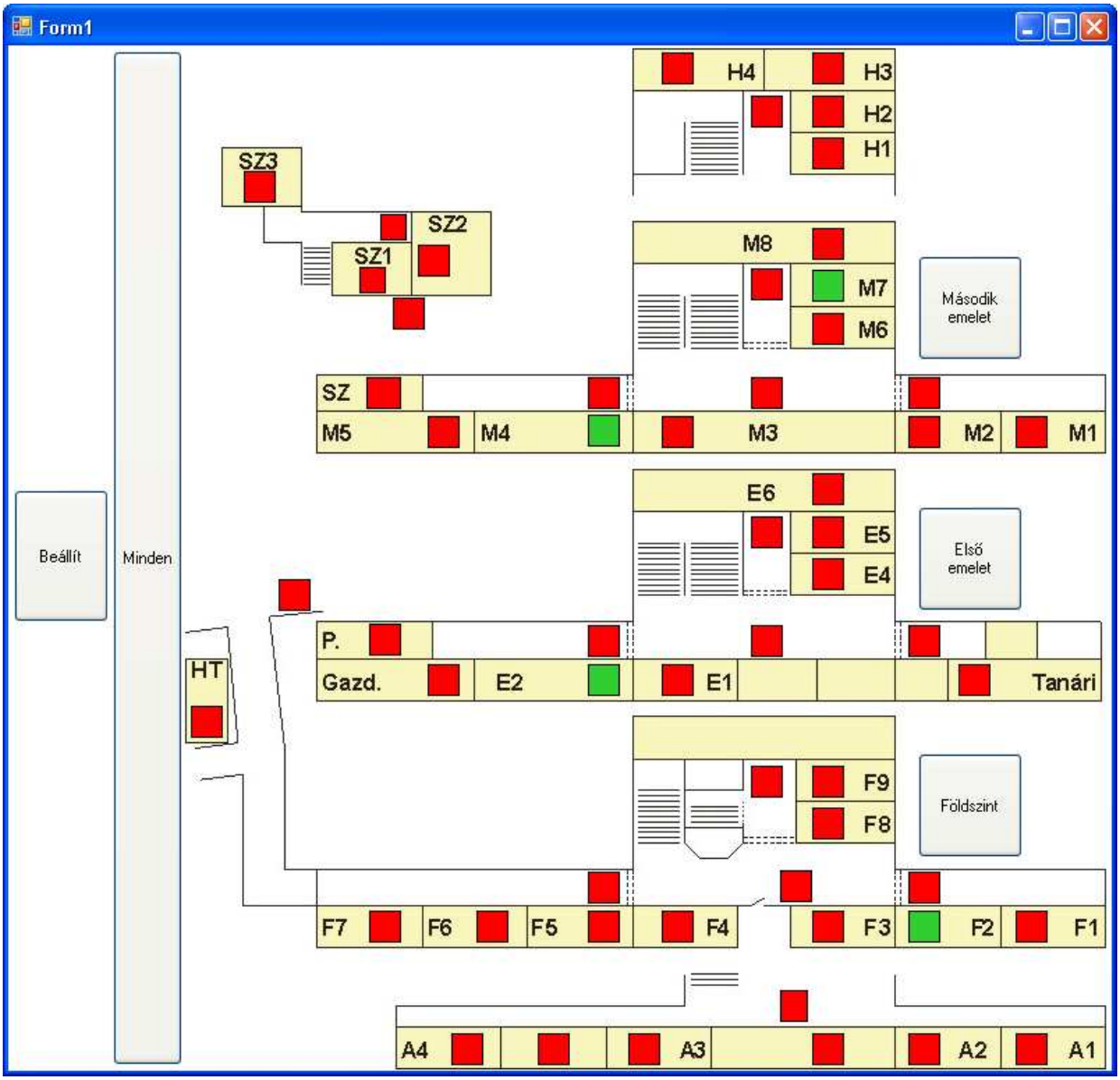
15-6. ábra, A csengetési rendek menedzselését ellátó alkalmazás



15-7. ábra, A csengetési rend hozzáadását lehetővé tevő ablak



15-8. ábra, A hangszórók állapotának menedzselésére szolgáló alkalmazás



15-9. ábra, A hangszórók állapotának alkalmi megváltoztatására szolgáló alkalmazás