

1. Tartalomjegyzék

1.	Tartalomjegyzék.....	1
2.	Bevezetés.....	3
3.	Az elektronikus hangszerek és a MIDI története	6
4.	A MIDI üzenetek.....	9
4.1.	A MIDI üzenetek fizikai átvitele.....	9
4.2.	A MIDI üzenetek logikai átvitele.....	11
4.2.1.	Üzem mód jellegű üzenetek (Channel Mode Messages)	12
4.2.2.	Hang jellegű csatornaüzenetek (Channel Voice Messages).....	13
4.2.3.	Egyszerű rendszerüzenetek (System Common Messages)	18
4.2.4.	Valós idejű rendszerüzenetek (System Real Time Messages)	19
4.2.5.	Exkluzív rendszerüzenetek (System Exclusive Messages).....	20
5.	A készülék leírása és használata.....	21
6.	A készülék hardvere	26
6.1.	Az LCD kijelző	29
6.2.	A PIC18F4550 mikrovezérlő ismertetése	30
7.	A Program felépítése	33
7.1.	A fejlesztői környezet.....	33
7.2.	A program feladatának meghatározása:	35
7.3.	A főprogram	41
7.4.	A MIDI üzenetek táblázatai a program memóriában	43
7.5.	A program összeállítások táblázata.....	50
7.6.	A billentyűzetkezelés megvalósítása.....	51
7.7.	A potenciométer (A/D) eljárások	57
7.8.	Az üzenet nevének kiírása.....	61
7.9.	Az Usart adás és vétel megvalósítása.....	63
7.10.	A FIFO tároló megvalósítása	69
7.11.	Az LCD kijelző	72
7.12.	A stringkezelő eljárások	73
7.13.	A makrók.....	75
8.	Irodalomjegyzék.....	78
9.	Ábrajegyzék	79
10.	A CD melléklet tartalma	80

Forráskód melléklet

Main.asm	81
Programs.asm	86
Keyb.asm.....	89
ADConv.asm	96
PrgWrite.asm.....	100
Usart.asm.....	103
Fifo.asm.....	106
Lcd.asm	107
String.asm.....	111
Macro.inc.....	113

2. Bevezetés

Diplomamunkám tárgya egy olyan hardver és szoftverfejlesztés, melynek eredményeként egy zenei célokra alkalmazható MIDI üzeneteket előállító keverő készül.

A hagyományos hangfrekvenciás jelekkel dolgozó keverőpultok általában sok bemenettel, és néhány kimenettel rendelkeznek. Ezek működési módjuk szerint lehetnek analógok, vagy digitálisak. A digitális keverőknél, ha a bemenetre analóg jelforrást kötünk, (mikrofon, elektromos gitár stb.) gondoskodni kell az analóg-digitális átalakításról is. Ha a bemenetre eleve digitális, hanginformációt tartalmazó eszközt csatlakoztatunk, akkor erre természetesen nincs szükség. Ez esetben elmarad az analóg-digitális átalakítás során történő mintavételezésből és kvantálásból adódó minőség veszteség. Ilyen eszköz lehet pl. egy CD vagy DVD lejátszó, számítógép hangkártya vagy szintetizátor digitális kimenete. A hangkártyák némelyike egész nagy tudású szintetizátort is tartalmaz, ezért ha ezentúl szintetizátort említek, beleértem ezeket a hangkártyákat is. A zenei billentyűzettel is ellátott szintetizátor logikailag két külön eszköznek tekinthető, ugyanis a szintetizátor mellett egy vezérlőbillentyűzetet is tartalmaz. Léteznek rack szekrénybe szerelhető, billentyűzettel nem rendelkező szintetizátorok is.

Hagyományos módon egy többcsatornás zenemű felvétele úgy történik, hogy sávonként külön-külön rögzítik az egyes hangszerek hangját. Ez történhet egyszerre, amikor a zenekar eljátsza a zeneművet, és minden hangszerhez odateszünk egy-két mikrofont. Annyi csatornás felvevőeszköz (magnetofon) kell természetesen hozzá, ahány mikrofont elhelyeztünk. A másik véglet, amikor minden szólamot külön-külön rögzítünk. Így könnyen létrehozható akár az egyszemélyes zenekar is. A felvett szólamokat visszajátszva és a keverőpulton a megfelelő hangerőarányokat beállítva megkapjuk a kész zeneművet. A szintetizátorok szólamainak felvételére viszont van másik módszer is: Nem a hangszer által előállított hangfrekvenciás jelet rögzítjük, hanem csak a billentyűleütéseket, amiket az adott szólam lejátszásakor elkövettünk. Erre szolgálnak az úgynevezett szekvencerek (dalszerkesztők). Ezeket ugyanúgy használhatjuk, mintha többsávos magnetofont használnánk. Az éppen felvenni kívánt szólamhoz tartozó csatornát felvételre kapcsoljuk, míg a többit lejátszásra. A dalszerkesztő ilyenkor nem csak az éppen felvétel alatt levő szólamot küldi tovább a szintetizátornak, hanem a korábban felvetteket is.

Ennek megvalósítására az alábbi megoldások léteznek:

- Célhardverrel (kis dobozka, kijelzővel, néhány nyomógommbal).
- Szintetizátorba beépített dalszerkesztővel.
- Személyi számítógépen futó dalszerkesztő programmal.

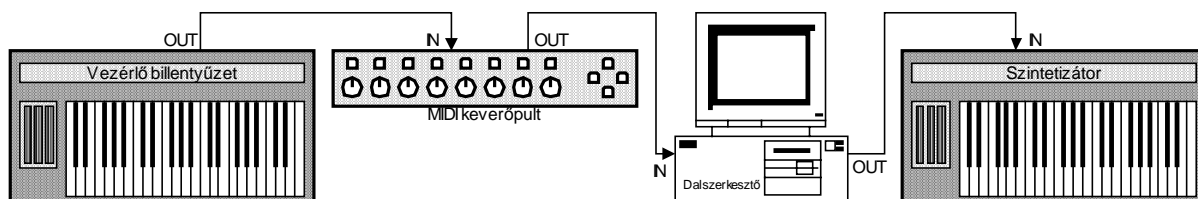
A felsorolt megoldásoknak több előnyük is van. Az egyik, hogy nem foglalunk el szólamonként 1-2 sávot a magnetofonon (esetleg magnetofonra sincs szükségünk). A másik, hogy a dalszerkesztőben át is alakíthatjuk a felvett szólamot. (Ha pl. egy hangot félreütöttünk, kijavíthatjuk anélkül, hogy az egész szólamot újból be kellene játszanunk.) Az így felvett szólamok ugyanúgy külön sávba kerülnek, mintha csak a sokcsatornás magnetofonnal vettük volna fel őket. Visszajátszáskor a szólamok hangerejét a dalszerkesztő programban található keverőpulttal tehetjük meg. Ez a keverőpult a szintetizátorba beépített keverőnek küldi az üzenetet, hogy az adott sávot milyen hangerővel játsza le. Ezek az úgynevezett MIDI üzenetek.

Felvetődött az igény, hogy ezeket a műveleteket ne a számítógép egerével kelljen végrehajtani, hanem valódi kezelőszervekkel. (Egérből ráadásul csak egy van, ezért ha egyszerre több sávon is szeretnénk hangerőt állítani, azt csak külön-külön tehetnénk meg.) Az is felmerült, hogy a hangerőn kívül mást is ily módon tudjunk állítani.

Hogyan lehet ezt megoldani? Adódik, hogy a kezelőszerveknek potenciométert használjunk, ami önmagában egy analóg eszköz. A potenciométerekből feszültségosztókat alakítunk ki, ahol a kimeneti feszültség a csúszka helyzetével arányos. Mivel a MIDI üzenetek digitális információk, ezért az így kapott feszültségeket át kell alakítani digitális értékekké. Ezt hívjuk analóg/digitális átalakításnak. Az így kapott számot felhasználva kell még egy olyan berendezés, ami ebből a számból magát az üzenetet előállítja. Ez lesz maga a szoftver. A szoftver futtatásához valamilyen mikroprocesszor szükséges. Hogy ne kelljen bonyolult hardvert építeni (mikroprocesszorral, memóriával, meg minden ehhez szükséges perifériával), keressünk olyan eszközt, amiben mindez együtt megtalálható. Ezt hívják mikrokontrollernek, magyarul mikrovezérlőnek. Ezek között olyat is lehet találni, amiben az összes szükséges periféria megtalálható a feladathoz.

Milyen perifériákra is van szükség? Kell az analóg digitális átalakító a potenciométerekkel előállított feszültség átalakításához. Mivel többféle üzenetet is szeretnénk küldeni, ezért kellenek digitális bemenetek, amikre az üzenetek kiválasztásához szükséges nyomógombokat köthetjük. A beállítások nyomon követéséhez visszajelzés szükséges. Erre a feladatra kiválóan alkalmas egy LCD kijelző, aminek a vezérléséhez viszont digitális be/kimenetek szükségesek. Végül az előállított üzeneteket el is kell küldeni. Mivel a MIDI

üzenetek aszinkron soros jelek, ezért kívánatos, hogy a mikrovezérlőnk soros porttal is rendelkezzen. (Lehetne ugyan bitenként, szoftveresen is soros adatot küldeni, de ez aránytalanul terhelné a processzort a többi feladat rovására). A soros porttal rendelkező mikrovezérlő lehetőséget biztosít arra, hogy a soros bemenetet is felhasználva a berendezéseinket láncba köthessük. Ily módon a készülékünket beiktathatjuk a zongorabillentyűzetet tartalmazó MIDI vezérlőbillentyűzet és a dalszerkesztő programot futtató számítógép közé.



1. ábra – A MIDI keverő beillesztése a rendszerbe

Ha szintetizátornak a dalszerkesztő gépben levő hangkártyát használjuk, akkor a dalszerkesztő és a szintetizátor közötti átviteli út csak logikailag létezik.

Elvárásunk az is, hogy készülékünk kikapcsolás után megtartsa a beállított állapotot, ehhez olyan adatmemória szükséges, mely tápfeszültség nélkül sem felejt el a tartalmát.

Ezeknek a kívánalmaknak megfelel a Microchip PIC18F4550 típusú mikrovezérlője. Ez a mikrovezérlő az előbb felsoroltakon kívül még számos perifériát tartalmaz. Ezek közül az USB port kihasználása szerepel a jövőbeni bővítési tervek között. Áramkörileg ez már most elő van készítve. Lehetőség van ugyanis a mikrovezérlőt egy úgynevezett bootloader programmal feltölteni. A bootloader úgy működik, hogy induláskor megvizsgálja az egyik bemenetre kötött nyomógomb állapotát. Ha nincs lenyomva a gomb, akkor a benne található normál programra ugrik, ha viszont lenyomott állapotban van, akkor arra a programra ugrik tovább, amivel lehetőségünk van az USB porton keresztül a mikrovezérlőben található programot kicserélni. Ehhez természetesen a PC oldalán is futtatni kell egy úgynevezett feltöltő programot, aminek segítségével áttölthetjük a módosított programot a mikrovezérlőbe. Mivel a készülék szolgáltatásainak bővítése ily módon hardveres beavatkozást nem igényel, a továbbfejlesztés lényegesen leegyszerűsödik, akár a felhasználó is végrehajthatja a program kicserélését.

3. Az elektronikus hangszerek és a MIDI története

Az 1900-as évek elején a zenei világban is elkezdődött az elektronikai forradalom. Eleinte két külön irányzat volt megfigyelhető az elektronikus hangszerek készítésében. Az egyik irányzat az elektromos orgonáké, a másik a szintetizátoroké volt. Az elektromos orgonákban általában oszcillátorokkal előállított négyszög vagy háromszögjelekből szűrők segítségével állították elő a különböző hangszíneket, míg a szintetizátorokban egy hangszín előállításához több oszcillátort is felhasználtak. Ezzel a módszerrel szinte végtelen sokféle hangzást lehet előállítani, viszont egy hangzáshoz nagyon sok összehangolt oszcillátor szükséges. Többszólamú zene létrehozásához mindezt ráadásul meg is kell többszörözni. A rengeteg oszcillátor összekapcsolása sem egyszerű feladat, ezért a kezelése nagyon bonyolult volt. A hőskorban ezért a szintetizátorok általában csak egyszólamúak voltak.

Az első elektronikus hangszert 1906-ban készítették. 1920-ban több kísérleti elektronikus hangszer mellett 2 szintetizátor is megjelent, a Theremin és az Ondes Martenot. A nagyközönség ebben az időben főleg a filmzenékben ismerhette meg ezeknek a hangszereknek a hangjait.

Érdekességgként megemlíthetjük a Hammond cég elektromechanikus orgonáját, amely olyan módon állította elő a rezgéseket, hogy ferritből készült különböző méretű és fogszámú fogaskerekeket forgattak álló tekercsek között. Ezek a fogazattól és a fordulatszámától függően feszültséget indukáltak a tekercsekben. Egy-egy hangszín előállításához több fogaskereket is használtak. Az indukált feszültséget az akkori technika lehetőségei szerint elektroncsöves erősítők és szűrők segítségével dolgozták fel. Ebből elképzelhetjük a hangszer súlyát és méreteit. Ebben a hangszerben valósították meg először a zenészek által annyira kedvelt Leslie-effektust is. Ezt oly módon hozták létre, hogy a hangszórókat változtatható sebességgel körbeforgatták. Ilyenkor nemcsak a hangerő, hanem a hangszín is erősen változott.

Az 1930-1950-es évekig egyre jobban fejlődtek az elektronikus hangszerek, bár a közönség ekkor is inkább csak a mozikban és a világkiállításokon találkozhatott a hangjukkal, mivel csak az egyetemeken és a hangstúdiókban léteztek ezek a berendezések. A legtöbb kísérleti eszközt a Bell Laboratóriumban és az RCA-nál fejlesztették ki.

Az 1947-ben feltalált tranzisztor újabb lökést adott az elektromos orgonák és szintetizátorok fejlesztésében.

1968-ban egy Walter Carlos nevű fiatalember elkészített egy albumot „Switched on Bach” címmel, amin Bach műveket adott elő egy Moog szintetizátoron. (Robert Moog a szintetizátort készítő mérnöknek a neve). Az emberekben a Moog név úgy összeforrt a szintetizátorral, hogy egy ideig szinonímaként használták. Innentől kezdve a szintetizátor betört a szórakoztatóiparba is.

Az 1970-es évek elején már több rockzenekar is elkezdte használni ezeket a hangszereket.

Az integrált áramkörök elterjedése mindinkább elérhetővé, és egyre nagyobb tudásúvá tette ezeket. A zenészek egyre több hangzást használtak, a színpadokon el kezdtek szaporodni a hangszerek. Jellemző kép a 70-80-as években, hogy a billentyűs teljesen körbeveszi magát ilyen hangszerekkel.

Ekkoriban vetődött fel, hogy próbáljuk meg összekötni egymással ezeket a szintetizátorokat, ahol az egyiket kinevezzük vezérlőszintetizátornak, a többit pedig ennek segítségével szolgáltassuk meg. Az ötlet nem volt újdonság, hiszen Robert Moog már korábban is próbált hasonló dolgot csinálni. Ő még analóg módon, a hangmagasságnak megfelelő feszültséggel oldotta ezt meg. Ennek két nagy hátránya is volt. Az egyik, hogy a feszültségértékeket nagyon pontosan kellett tartani (ne legyen a hang hamis), a másik hogy a többszólamú (polifon) hangszerekhez annyi kábel szükséges, ahány hangot egyszerre meg szeretnénk szolgáltítani. (Pl. 24 polifónia fokú hangszerhez 24 különböző feszültségértéket kell igen pontosan átvinni.)

A 80-as években a szintetizátorok már általában digitális vezérlést tartalmaztak, ezért vetődött fel az ötlet, hogy oldjuk meg ezeket a vezérlési feladatokat is digitálisan. A digitális átvitelnek megvan az az előnye, hogy csak 2 különböző értéket (a 0-t és az 1-et) kell tudnunk megkülönböztetni egymástól. Ez elég nagy biztonsággal megvalósítható, szemben az analóg vezérléssel, ahol a feszültség értékének kis mértékű megváltozása már zavaró hangmagasság változást fog okozni. A digitális vezérlésnek megvan még az a másik előnye is, hogy nem csak egyféle információt továbbíthatunk egy érpáron, hanem szinte tetszőlegesen sokfélét.

1981 júniusában az amerikai NAMM (National Association of Music Merchants) kiállításon három vezető hangszergyártó cég kulcsembere kezdeményezte egy olyan szabvány létrehozását, amelynek célja, hogy ezeket a hangszereket össze lehessen kötni egymással, függetlenül attól, hogy melyik gyártó készítette őket. 1981 novemberére kidolgoztak egy szintetizátor vezérlési szabványt, amelyet USI (Universal Synthesier Interface) névre keresztelnek el. Ez már hasonlított a mai MIDI szabványhoz, de az átviteli sebessége még 19200 bit/szekundum volt, a csatlakozói pedig 3,5"-os jack dugók voltak. A szabvány alapján készített berendezéseket 1982 januárjában a gyakorlatban is bemutatták a NAMM kiállításon.

Ez a szabvány alapvetően kettős koncepcióval készült:

- Az egyik szintetizátorral meg lehessen szólaltatni a másik szintetizátor hangkeltő részét.
- Az úgynevezett szekvencerrel (dalszerkesztő) rögzíteni tudjuk, ill. visszajátszhatjuk a korábban felvett szerzeményeket. Ennek lényege, hogy magnetofon nélkül fel tudjuk venni a zenedarabokat, amit azután a hangszer és az USI segítségével vissza tudunk játszani.

1982 júniusában ismét a NAMM-on találkoztak a gyártók, és az ekkorra már tökéletesített szabványt fogadták el. Az átviteli sebességet 31250 bit/szekundumra emelték meg, a csatlakozókat pedig az 5 pólusú tuchel (hivatalosan DIN) dugóban definiálták. A szabvány nevét is megváltoztatták MIDI-re (Musical Instruments Digital Interface).

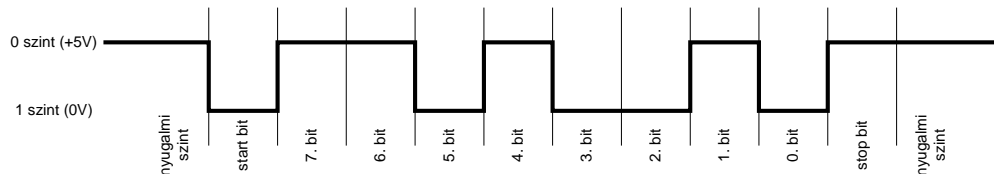
Az 1983 januári NAMM kiállításra több gyártó is elhozta a saját MIDI-s eszközét, ahol össze is kötötték őket. Bár az összeköttetés sikerült, rögtön kiderültek a hiányosságok is. Mivel a hajlítókerék (a szintetizátorokon a „nyávogtatás” kezelőszerve) átvitelét nem definiálták, mindegyik gyártó a saját házi szabványát alkalmazta erre, ami természetesen különbözött a másik gyártóétól. 1983 augusztusra azután az ilyen zavarok elkerülése érdekében pontosították a MIDI specifikációt.

Azóta a MIDI szabvány alapvetően nem változott, csupán kiegészítésekkel bővült.

4. A MIDI üzenetek

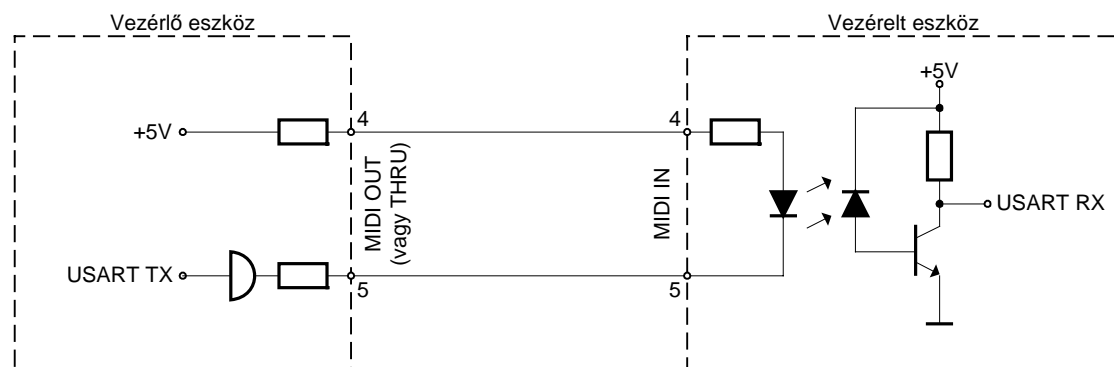
4.1. A MIDI üzenetek fizikai átvitele

A MIDI üzenetek fizikai átvitele aszinkron soros módon történik. Átviteli sebessége 31250 bit/szekundum, 1 startbitet, 1 stopbitet tartalmaz, paritás bit nincs. Negatív logikájú, tehát a logikai 0-nak +5V, a logikai 1-nek 0V feszültség felel meg.



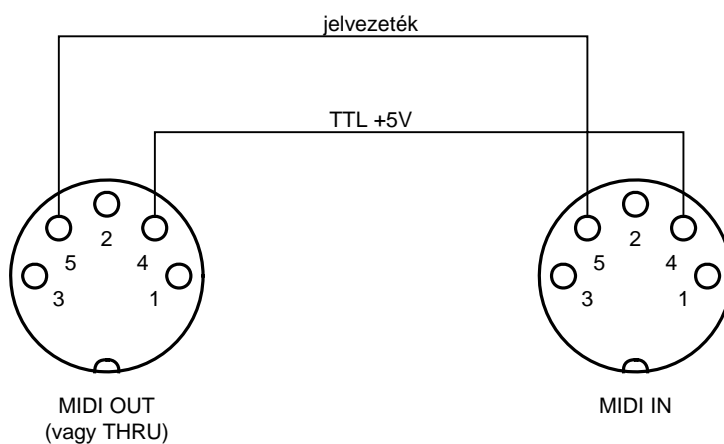
2. ábra - Az aszinkron soros adatátvitel jelalakja

Az összeköttetést galvanikus leválasztással valósítják meg. Minden eszköz bemenete egy optocsatolót tartalmaz. Ezt a módszert a földhurkok elkerülése miatt találták ki.



3. ábra - A MIDI kommunikáció tipikus áramköri megvalósítása

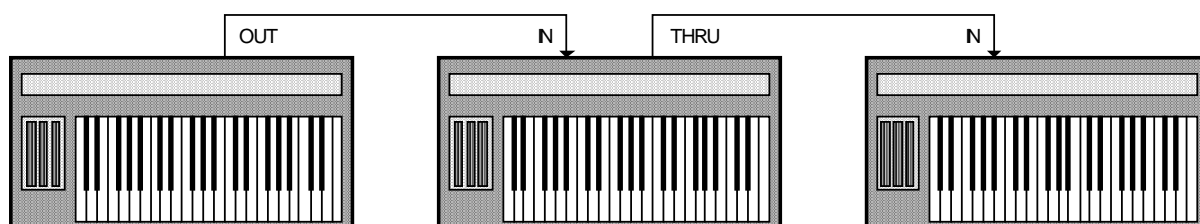
A csatlakozáshoz 5 pólusú tuchel (DIN) csatlakozókat alkalmaznak. Az adót és a vevőt az ábrán látható módon kell összekötni.



4. ábra - A MIDI kábel bekötése

Három eltérő funkciójú csatlakozó található az eszközökön:

- MIDI IN (bemenet)
- MIDI OUT (kimenet)
- MIDI THRU (továbbított jel). A MIDI bemenetre érkezett jelet logikai feldolgozás nélkül továbbítja a MIDI THRU kimenetre. Ez lehetővé teszi, hogy a vezérlő billentyűzetről vagy dalszerkesztőből érkező jelek több szintetizátorra is eljussanak, az ábrán látható módon láncba kötve az eszközöket.



5. ábra - A MIDI Thru alkalmazása

4.2. A MIDI üzenetek logikai átvitele

Eddig a fizikai átvittel foglalkoztunk. Nézzük meg, hogy mit tartalmazhatnak az így átvitt bájtok.

Egy üzenet legalább 1 bájt, legfeljebb 3 bájt hosszúságú (kivéve az exkluzív üzeneteket, mert azok bármekkora hosszúságúak lehetnek). Az üzenet építőkövei a státusz és az adatbájtok. Az üzenet minden esetben egy státuszbájttal kezdődik, utána pedig egy, vagy több adatbájt következik. A státuszbájt határozza meg, hogy az üzenet adatbájtjai milyen információt hordoznak. A státuszbájt legfelső helyiértékű bitje mindig 1, az adatbájtoké pedig mindig 0. Ebből következik, hogy 7 bit marad az információ kódolására. Ha az egymás után következő üzenetekben a státuszbájt értéke nem változik meg, nem szükséges újra elküldeni. A vevőnek kötelessége az utolsó státuszbájtot eltárolni, és ha a következő üzenet elejéről hiányzik, akkor az eltárolt értékkel kell helyettesíteni. Ezt a módszert státusztartásnak hívjuk, célja hogy a MIDI vezetéken kevesebb adatot kelljen továbbítani, ezáltal a késleltetési idők is csökkennek. Késleltetési időnek hívjuk azt az időt, ami ahhoz szükséges, hogy egy üzenet eljusson az adótól a vevőig.

Az átviteli sebességből következik, hogy egy 3 bájtos üzenet átvitele kb. 1 milliszekundum ideig tart. Ha az adóoldalon levő billentyűzeten lenyomunk pl. egy 3 hangos akkordot, akkor ez 3 hang megszólaltatása (note on) üzenetet fog okozni. A 3 db 3 bájtos üzenet átvitele 9 bájt átvitelét igényelné. Ha kihasználjuk a státusztartás lehetőségét, akkor 7 átvitt bájt is elegendő az akkord megszólaltatásához. Ebből következik, hogy a késleltetés az eredeti 3 milliszekundumról 2,3 milliszekundumra csökken.

Alapvetően 5 féle üzenettípust különböztetünk meg:

1. Üzem mód jellegű üzenetek (Channel Mode Messages)
2. Hang jellegű csatornaüzenetek (Channel Voice Messages)
3. Egyszerű rendszerüzenetek (System Common Messages)
4. Valós idejű rendszerüzenetek (System Real Time Messages)
5. Exkluzív rendszerüzenetek (System Exclusive Messages)

Az első két típus a csatornakoncepció alapján csak bizonyos eszközöknek szól. Az üzenet státuszbaírtja minden ilyen csatornaüzenet esetén tartalmaz egy 4 bites mezőt, ami meghatározza, hogy melyik csatornára vonatkozik az üzenet. A 4 bites mezővel 16 csatornát tudunk kiválasztani. A vevő oldalon levő hangszer csak akkor fog reagálni egy csatorna üzenetre, ha a 4 bites mező által meghatározott csatorna száma megegyezik a hangszerben beállított csatorna számával.

4.2.1. Üzem mód jellegű üzenetek (Channel Mode Messages)

A MIDI-s hangszerek 4 üzemmódban értelmezhetik a bejövő hang-jellegű csatorna üzeneteket.

1. OMNI ON, POLY üzemmódban a hangszer nem figyeli a bejövő üzenet csatornakódját, hanem minden üzenetre reagál. Ez akkor használható, ha csak egy vevő van a rendszerben, és nem akarunk bajlódni a csatornaszámokkal. Ezenkívül polifónikus üzemmódba kerül, azaz egy időben több hangot képes megszólaltatni. Hogy ez a több hang mennyi, az a hangszer polifónia fokszámától függ.
2. OMNI ON, MONO módban szintén minden csatornaüzenetet vesz, de egy időben csak egy hangot szólaltathat meg. Ha újabb hangot szólaltat meg, az előző hangot elnémítja.
3. OMNI OFF, POLY módban a hangszer polifónikusan működik, és azon a csatornán, amelyet számára kijelöltünk. Multitimbrálisnak nevezzük azokat a hangszereket, amelyek képesek egyszerre több különböző csatornán működni, esetleg mind a 16-on. Ez olyan, mintha több különböző hangszerünk lenne. (Pl. az egyes csatornán zongora hangszer, a kettesen gitár, a tízesen dob, stb.) A mostani hangszerek általában képesek erre.
4. OMNI OFF, MONO módban csatornaérzékeny a hangszer, de egy csatornán csak egy hang megszólaltatására képes. Ezt a beállítást tipikusan a gitár-MIDI átalakítóval használják, hiszen a gitár 6 húrral rendelkezik, de egy húr egy időben csak egy hangon képes megszólalni. A hat húrhoz viszont akár hat féle különböző hangszert is rendelhetünk.

Ezeket az üzemmódokat a hangszer kezelőszerveivel lehet beállítani (ha a hangszer ismeri ezeket az üzemmódokat), de hang-jellegű csatorna üzenetek között is megtalálható az üzemmód beállításra vonatkozó üzenet.

4.2.2. Hang jellegű csatornaüzenetek (Channel Voice Messages)

Ezek az üzenetek hordozzák a zenei előadás történéseit. Egy státuszbytával kezdődnek, és egy vagy 2 adatbytát tartalmaznak. Az üzenetek kódjait hexadecimális formában szokásos megadni.

Note On - hang megszólalása (9n bb vv)

Ez az üzenet a zenei hang megszólaltatására szolgál. Az adó akkor fogja küldeni, ha a billentyűzetén lenyomtak egy billentyűt. Az 'n' értéke tartalmazza a csatorna számát, ami 0..F közötti értéket vehet fel. Mivel az emberek jobban szeretik a számozást 1..16-ig, mint 0..15-ig, a hangszereken 1-16 csatornákat lehet beállítani. Az 'n' 0 értéke felel meg az 1. csatornának, az 1 érték a 2. csatornának egészen az F-ig, ami a 16-os csatornát jelenti.

A 'bb' értéke jelenti, hogy melyik billentyű lett lenyomva. Mivel az adatbyt értéke decimálisan 0-127 között lehetséges, ez 128 különböző hangmagasságot jelent. A normál 440Hz-es 'A' hang kódja: 45h (decimális 69). Mivel oktávonként 12 hang található, ez több mint 10 oktávnyi hangterjedelemhez elegendő. A hagyományos zongorán 88 billentyű van, így ez elég.

A 'vv' értéke jelenti a hang leütésének sebességét. A legnagyobb érték itt is a decimális 127, ami a maximális hangerőt jelenti. Ha a vezérlő billentyűzet nem érzékeny a leütés erősségére, (azaz sebességére) akkor általában középértéket (decimális 64) küld, vagy a hangszeren beállítható fix értéket.

Pl.

90h 45h 7Fh:

az 1. csatornán szólaljon meg a normál 'A' (440Hz) hang maximális hangerővel

96h 3Ch 60h:

a 7. csatornán szólaljon meg a normál 'C' hang

Note Off - hang kikapcsolása (8n bb vv)

A billentyű felengedését jelenti, az addig kitartott állapotban levő hang lecsengési fázisba kerül. Az 'n', 'bb', 'vv' értéke (mint a Note On esetén) a csatornát, a billentyű számát, ill. a felengedés sebességét jelenti. A gyakorlatban nem sok hangszer érzékeli a billentyű felengedésének sebességét. Egyes hangszerek a billentyű felengedését is Note On üzenettel oldják meg, ahol a hangerőhöz ('vv') 0 értéket rendelnek. Mivel zavart nem okoz, ez is bekerült a MIDI specifikációba.

Pl.

90h 45h 40h:

Az 1. csatornán hallgasson el a normál A hang

96h 3Ch 40h:

A 7. csatornán hallgasson el a normál C hang

Pitch Bend – hajlítás (En aa ff)

A billentyűzet bal oldalán levő hajlítókerékkel (vagy karral) lehet a hajlítást („nyávogtatást”) vezérelni. A hajlítás történhet lefelé, vagy felfelé. Mivel 'aa' és 'ff' is 7 bites, ezért a hajlítás értéke 14 bites. Az 'aa' tartalmazza a kisebbik helyiértékű, 'ff' pedig a nagyobbik helyiértéket. Ez 16384 különböző értéket adhat. A 0 érték a maximális lefelé hajlítás, 8192 a hajlítás nélküli alapállapot, a 16383 pedig a maximális felfelé hajlítás. A gyakorlatban ritkán használják ki a 14 bites felbontást, általában megelégszenek 8-10 bittel is.

Pl.

E0h 00h 00h:

az 1. csatornán legnagyobb hajlítás lefelé

E5h 7Fh 7Fh:

a 6. csatornán legnagyobb hajlítás felfelé

E7h 00h 40h:

a 8. csatornán hajlítás középállásban (alapérték, nincs hajlítás)

Poliphonic Pressure (An bb pp)

Azok a billentyűzetek tudnak ilyen üzenetet küldeni, amelyek érzékelné tudják a már lenyomott billentyűre ható nyomásértéket. Ezzel lehet pl. egy lenyomott akkordnál a vibrató mélységét, sebességét vagy hangmagasságát megváltoztatni. Hogy mit kezd a vevő ezzel az üzenettel, az a vevő beállításától függ. Az 'n' értéke itt is a csatorna, 'bb' a billentyű száma, 'pp' pedig a nyomás értéke.

Channel Pressure (Dn pp)

Szintén nyomáserősség közlésére szolgál (mint az előző), de nem az egyes billentyűkre, hanem csak összességében a teljes billentyűzetre vonatkozó nyomás. A Poliphonic Pressure képességekkel rendelkező billentyűzet igen drága, ez olcsóbban megvalósítható attól. Az 'n' értéke itt is a csatornát, 'pp' pedig a nyomás erősségét jelenti.

Program Change (Cn pp)

Ezzel lehet a vevővel közölni, hogy melyik hangszeren szólaljon meg. Az 'n' érték itt is a csatornát, 'pp' pedig a hangszer számát jelenti. Ezzel 128 hangszer közül választhatunk. A General MIDI ajánlás meg is határozza, hogy milyen értékhez milyen hangszer tartozzon. A hangszereken 1-től 128-ig szokták megszámozni a hangszereket, ennek a 'pp' = 0..127 érték feleltethető meg. A hangszereket 16 csoportban, csoportonként 8 féle hangszerben határozták meg. Ily módon az 1-től 8-ig terjedő zongora csoportban az 1. hangszer az Acoustic Grand Piano, a 8. pedig a Clavi névre hallgat. A második csoportot 9. hangszerként a Celesta nyitja és a Dulcimer zárja. Az utolsó csoportban (ami különféle effekteket tartalmaz) 121. a Guitar Fret Noise, 128. pedig a Gunshot.

Manapság nagyon sok hangszer túllépte a 128 hangszerválasztási határt. Ennek feloldására 1990-ben született egy egységes ajánlás, amely a Bank Select nevet kapta (lásd. Control Change üzenetek).

Control Change (Bn cc vv)

Ez a legnépesebb üzenettípus, amely igen különböző célokra szolgál. Az eddigiek közé be nem sorolható üzenetek ide vannak betéve (pl. pedálok, moduláció, csatorna hangerő, csatorna panoráma, stb.)

Az 'n' érték itt is a csatorna száma, 'cc' a kontroller azonosítója, 'vv' pedig az értéke.

A következő kontroller azonosítókat definiálták:

- 0: Bank Select (hangszerbank kiválasztása, minden ilyen bank maximum 128 hangszert tartalmaz, amelyekből a Control Change üzenettel választhatjuk ki a megfelelőt)
- 1: Modulation (modulációs kar/kerék értéke)
- 2: Breath Controller (MIDI-s fúvós hangszernél a csőbe fújt légnyomás értékét jelenti, általában hangerőt szabályoznak vele)
- 3: nem definiált
- 4: Foot Controller (lábpedál állítja ezt a paramétert)

- 5: Portamento Time (ha azt szeretnénk, hogy a két egymás után megszólaló hang ne különülten szólaljon meg, hanem az egyikből hajlítással menjen a másikba, akkor ezzel szabályozhatjuk a hajlítás idejét)
- 6: Data Entry (A 98-101 RPM és NRPM által kiválasztott regiszterbe itt írhatjuk be az adatot, használatára később kitérünk)
- 7: Main Volume (ez határozza meg a csatorna hangerejét)
- 8: Balance (ha egy hangszer hangja két hangmintából tevődik össze, a két hangminta hangerőarányát ezzel lehet beállítani. 00h esetén csak az 1. hangmintán fog megszólalni, 40h esetén mindkét minta egyforma hangerővel, 7Fh esetén csak a 2. minta szól)
- 9: nem definiált
- 10 (0Ah): Pan (a csatorna bal/jobbs oldali hangerő arányát szabályozza)
- 11 (0Bh): Expression (a Main Volume beállításához hasonló, hangerőt állít be)
- 12 (0Ch): Effect Control 1
- 13 (0Dh): Effect Control 2
- 14-15 (0Eh-0Fh): nem definiált
- 16-19 (10h-13h): General Purpose 1-4
- 20-31 (14h-1Fh): nem definiált
- 32-63 (20h-3Fh): az előző 32 kontroller kiterjesztése 14 bites értékre (kisebb helyiértéke)
- 64 (40h): Damper Pedal
- 65 (41h): Portamento
- 66 (42h): Sostenuto
- 67 (43h): Soft Pedal
- 68 (44h): Legato Footswitch
- 69 (45h): Hold 2
- 70 (46h): Sound Controller 1 (default: Timbre Variation)
- 71 (47h): Sound Controller 2 (default: Timbre/Harmonic Content)
- 72 (48h): Sound Controller 3 (default: Release Time)
- 73 (49h): Sound Controller 4 (default: Attack Time)
- 74-79 (4Ah-4Fh): Sound Controller 5-10
- 80-83 (50h-53h): General Purpose 5-8
- 84 (54h): Portamento Controll
- 85-90 (55h-5Ah): nem definiált
- 91 (5Bh): Effects 1 Depth (External Effects Depth)

- 92 (5Ch): Effects 2 Depth (Tremolo Depth)
- 93 (5Dh): Effects 3 Depth (Chorus Depth)
- 94 (5Eh): Effects 4 Depth (Celeste (Detune) Depth)
- 95 (5Fh): Effects 5 Depth (Phaser Depth)
- 96 (60h): Data Increment
- 97 (61h): Data Decrement
- 98 (62h): NRPM - Non Registered Parameter Number (kisebb helyiérték)
- 99 (63h): NRPM - Non Registered Parameter Number (nagyobb helyiérték)
- 100 (64h): RPM - Registered Parameter Number (kisebb helyiérték)
- 101 (65h): RPM - Registered Parameter Number (nagyobb helyiérték)
- 102-119 (66h-77h): nem definiált
- 120 (78h): All Sound Off
- 121 (79h): Reset All Controllers
- 122 (7Ah): Local Control On/Off
- 123 (7Bh): All Notes Off
- 124 (7Ch): Omni Off
- 125 (7Dh): Omni On
- 126 (7Eh): Mono On
- 127 (7Fh): Mono Off

Pl.

B0h, 07h, 7Fh:

Az 1. MIDI csatorna hangereje = 127 (maximum)

BFh, 0Ah, 40h:

A 16. MIDI csatorna panorámája középállásban

A felsorolt kontrollerek kevésnek bizonyulhatnak, ha mindegyik hangszergyártó cég a saját speciális kontrollereit használni szeretné. Erre megoldás az RPM és az NRPM paraméterek. Ez oly módon használható, hogy az RPM (100-101) vagy az NRPM (98-99) helyre beírjuk, hogy melyik kontrollert szeretnénk megcímezni. A 7 vagy 14 bites adatot pedig a Data Entry (6, 38) helyre írjuk. A címző regiszterek 14 bitesek, így 16384 db RPM és ugyanannyi NRPM kontrollert lehet megcímezni.

A következő három RPM üzenet van definiálva:

- 0 (00h): Pitch Bend Sensitivity (hajlítókerék érzékenysége)
- 1 (01h): Fine Tuning (finom hangolás)
- 2 (02h): Coarse Tuning (durva hangolás)

Példaként nézzük meg a Soundblaster hangkártyák Filter Cutoff (szűrő vágási frekvencia) kontrollerét. Ezt a kontrollert az NRPM 16277 címen lehet elérni. Az adat felső helyiértékére mindig 40h, alacsony helyiértékére pedig egy 7 bites érték adandó. Írjuk fel a 16277 decimális számot bináris formában. Az így kapott számot rögtön 2db 7 bites részre bontjuk (1111111 0010101). A 7 bites részeket hexadecimálisra alakítjuk (7Fh, 15h). Az NRPM nagyobbik helyiértékére (63h) beírjuk a 7Fh értéket, a kisebb helyiértékére (62h) pedig a 15h értéket. A Data Entry magas helyiértékére (06h) beírjuk a 40h értéket, az alacsony helyiértékére (26h) pedig a szűrési frekvenciát meghatározó értéket. Ha pl. a 8. MIDI csatornán 85 (55h) értéket szeretnénk adni a szűrőnek, akkor a státusztartást is kihasználva a következő bájt sorozatot kell küldeni:

B7h, 63h, 7Fh, 62h 15h, 06h, 40h, 26h, 55h

4.2.3. Egyszerű rendszerüzenetek (System Common Messages)

Ezek az üzenetek a rendszer egészének szólnak, ezért itt nincs szükség csatorna megadására.

MIDI Time Code Quarter-Frame (F1h)

MIDI idő kód negyed keret.

Song Position Pointer (SPP) (F2, ll, hh)

Szinkronizált lejátszás során a dal pozícióját lehet megszámozni vele. A pozíció 14 bites érték, alsó helyiértéke 'll', a felső 'hh'. Értéke 1/16-od hangonként eggyel növekszik.

Song Select (F3h)

Nem definiált (F4h, F5h)

Tune Request (F6h)

End Of Exclusive (F7h)

Exkluzív üzenet vége jelzés.

4.2.4. Valós idejű rendszerüzenetek (System Real Time Messages)

Ezek is a rendszer egészének szólnak, ezért itt sincs szükség csatorna megadására.

Timing Clock (F8h)

Ennek segítségével lehet az ütem szinkronizálását megoldani. Az adónak negyedhangonként 24-szer kell ezt az üzenetet elküldenie. Az egy perc alatt lejátszott negyedhangok száma jelenti a dal tempóját. Ennek mértékegysége a BPM (Beats Per minute).

Nem definiált (F9h)

Start (FAh)

Szinkronizált működés esetén az adó (általában a dalszerkesztő) küldi ezt az üzenetet.

Continue (FBh)

Ha Stop üzenettel leállítottuk a lejátszást, akkor Continue üzenettel lehet jelezni vevőnek, hogy ne nullázza a saját belső 'Song Position Pointer'-ét, hanem onnan folytassa a lejátszást, ahol abbahagyta.

Stop (FCh)

A lejátszás leállítását jelezhetjük a vevőnek. A vevőnek nem szabad törölnie a belső 'Song Position Pointer'-ét, mert 'Continue' üzenet esetén onnan kell folytatnia a lejátszást.

Nem definiált (FDh)

Active Sensing (FEh)

A vevő ebből az üzenetből értesülhet arról, hogy a bemenete összeköttetésben áll egy adóval. Az adónak akkor kell ezt az üzenetet 300 milliszekundumonként küldenie, ha közben semmilyen más üzenetet nem küldött. Használata opcionális.

System Reset (FFh)

Hatására a vevő alapállapotba kerül. A még hallható hangokat leállítja, kontrollereit alaphelyzetbe hozza, törli az 'SPP'-t.

4.2.5. Exkluzív rendszerüzenetek (System Exclusive Messages)

System Exclusive Messages (F0h id xx ... yy F7h)

Így lehet olyan üzenetet átvinni, ami csak egy meghatározott hangszernek szól.

Az üzenet mindig F0h bájjal kezdődik.

Az 'id' kétféle dolgot jelenthet:

- A hangszer gyártójának azonosítóját (00h..7Ch)
- Az üzenet jellegét (7Dh, 7Eh, 7Fh)

Ha az 'id' a hangszer gyártójának azonosítóját jelenti:

A vevő csak akkor fog reagálni az üzenetre, ha az ő azonosítója megegyezik az üzenetben találhatóéval. A hangszer azonosítóját a gyártó adja, ez nem módosítható. Értéke 1..124 (01h..7Ch) között lehet. Ha 'id' értéke nulla akkor a következő két bájt határozza meg az azonosítót. Ilyen módon az azonosítók lehetséges számát még 16384-el kiterjesztették. A gyártók szerinti egyedi azonosítók kiosztásáról MIDI ajánlás született. Az 'xx'..'yy' tartalmazza az átvitt adatokat. Az adatok értéke 00h..7Fh között lehet. Az adatok mennyisége nincs meghatározva, jelentésüket maguk a gyártók döntik el, a MIDI erre vonatkozólag semmilyen megkötést nem ír elő.

Ha az 'id' nem a gyártó azonosítóját jelenti:

- 125 (7Dh): Nem üzleti célú felhasználásra van tartalékolva (fejlesztőknek, egyetemeknek).
- 126 (7Eh): Univerzális, nem valósidejű üzenetekre szolgál. Nagyon sokféle üzenet található ebben a kategóriában, a fájlok átvitelétől kezdve a hangminták továbbításáig, stb. Részletes leírásukat a MIDI specifikáció tartalmazza.
- 127 (7Fh): Univerzális, valósidejű üzenetekre szolgál. Ez is rendkívül népes kategória, az idő kódok átvitelétől kezdve a fényberendezések vezérléséig sokminden megtalálható benne. Részletes leírásukat a MIDI specifikáció tartalmazza.

Az üzenet végét minden esetben az EOX (F7h) bájt jelzi.

5. A készülék leírása és használata

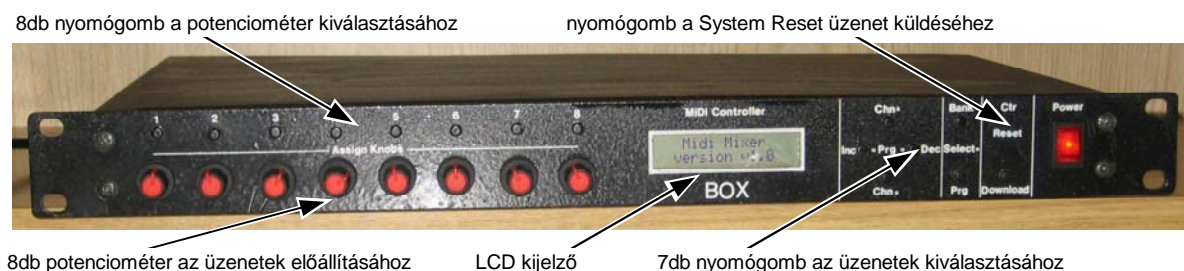


6. ábra - A készülék fényképe

A készülék a MIDI szekvencerrel létrehozott többsávós zene utólagos keverésére, valamint az egyes sávok hangzásának módosítására, vagy felvétel közbeni beállítására szolgál. Pl. ha mind a 8db. potenciométerhez hangerő-állítási funkciót rendelünk, majd a MIDI csatornákat is megfelelően beállítjuk, akkor a 8 forgatógombbal a 8 sáv hangerejét tudjuk külön-külön szabályozni. Ha ezt a szekvencer felvételi állásában tesszük meg, akkor a szekvencer ezt el is tárolja, és a visszajátszásnál eszerint fogja lejátszani a felvett zeneművet.

A készülék tartalmaz:

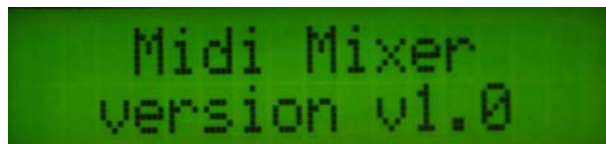
- 8 db potenciométert a MIDI üzenetek előállításához
- 16 db nyomógombot, amivel a potenciométerekhez rendelt MIDI üzeneteket (programokat) lehet kiválasztani. Ebből 8 db szolgál a potenciométer kiválasztásához, 7 db pedig arra, hogy meghatározzuk a kiválasztott potenciométerhez tartozó üzenetet. Egy pedig a System Reset (Rendszer reset) üzenetet küldésére szolgál.
- 2 db MIDI csatlakozót (MIDI IN, MIDI OUT)
- 1 db 2x16 karakteres LCD kijelzőt



Megjegyzés: A készüléken jelenleg két nyomógomb feliratozása hibás (Preset- helyett prg, Preset+ helyett Download a felirat

7. ábra - A készülék kezelőszervei

A készülék bekapcsolásakor a következő bejelentkező üzenet látható az LCD kijelzőn:



8. ábra - LCD bejelentkező felirat

MIDI üzenetet a 8 potenciométer forgatásával, valamint a **System Reset** nyomógomb megnyomásával tudunk előállítani. Programnak nevezzük azt az összerendelést, hogy egy potenciométer elforgatása esetén milyen MIDI üzenetet fog a készülék előállítani, és a MIDI OUT csatlakozóján kiküldeni. Mindegyik potenciométerhez a nyomógombok segítségével hozzárendelhetjük a saját programját, azaz hogy milyen üzenetet állítson elő, ha elfordítjuk a potenciométer forgatógombját. Bármelyik potenciométerhez bármelyik üzenetet hozzárendelhetjük, és a MIDI csatornáját is szabadon beállíthatjuk. A potenciométerek egyenrangúak, egyiknek sincs kitüntetett szerepe. A programok csoportosítva vannak. A csoportokat bankoknak hívjuk. Ennek az a szerepe, hogy a különböző hangszerekhez tartozó programok külön bankokba kerüljenek. Jelenleg 2 bankot tartalmaz a készülék, az egyikben vannak az általános programok (hangerő, panoráma, stb.), a másikban a Soundblaster AWE32/AWE64/Live hangkártyához tartozó programok. Ha ebből a bankból választjuk ki valamelyik programot, csak az előbb felsorolt hangkártyák fognak reagálni az így küldött üzenetekre.

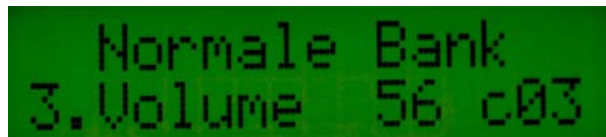
A "programs.asm" fájl tartalmazza a programokat és a bankokat. Ha egy új hangszer vezérlésére szeretnénk a készüléket megtanítani, akkor a fájl bővítésével, a program újrafordításával, majd a készülékben található mikrovezérlő újraprogramozásával lehetőség van erre. Ehhez természetesen szükség van a hangszer gépkönyvére, hiszen abban vannak leírva, hogy melyik paraméter állításához milyen bájtsorozatot vár a hangszer. Hogy ezt a bájtsorozatot hogyan tudjuk bevinni a "programs.asm" fájlba, arról a későbbiekben még szó lesz.

Példa egy programbeállításra:

- 1. potenciométerhez hangerő állítás az 1. MIDI csatornán,
- 2. potenciométerhez panoráma szabályozás az 1. MIDI csatornán
-
- 8. potenciométerhez Filter Cutoff paraméter állítása a 3. MIDI csatornán

A program beállítása a következőképpen történik:

A potenciométerek felett található 8 gombbal kiválasztjuk, hogy melyik programját szeretnénk módosítani.

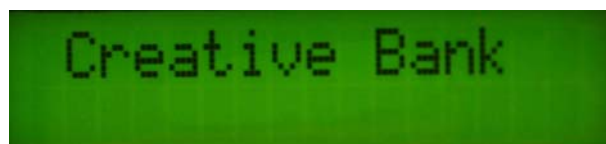


9. ábra - LCD felirat (3.Volume 56 c03)

Ekkor az LCD kijelző felső sorában megjelenik, hogy a jelenleg kiválasztott program melyik bankban található, alsó sorában pedig a következők:

- kiválasztott potenciométer száma (jelenleg 3)
- program neve (jelenleg Volume)
- potenciométer aktuális állása (jelenleg 56)
- midi csatorna száma (jelenleg 3).

Ha nem az aktuális bankban található programok közül szeretnénk választani, a **Bank Select** gombbal lépkedhetünk a bankok között.



10. ábra - LCD felirat (Creative Bank)

A kijelzőn megjelenik az éppen kiválasztott bank neve. A bankok között csak előre lépkedhetünk, az utolsó bank után újra az elsőbe lépünk.

A bank kiválasztása után a **Prg+** vagy a **Prg-** gombokkal választhatunk a bankban található programok közül.



11. ábra - LCD felirat (3.FiltCt 76 c01)

Mivel itt mindkét irányba léphetünk, nincs körbejárási lehetőség. Az első programon állva a **Prg-**, az utolsó programon állva a **Prg+** gomb hatástalan.

Bankváltás után nem történik meg automatikusan a program kiválasztása is, hanem az csak valamelyik **Prg** gomb megnyomása esetén. Bankváltáskor mindig az új bank elejéről indulunk. Ha ekkor megnyomjuk a **Prg-** gombot, akkor az első, ha a **Prg+** gombot, akkor a második programot választjuk ki. Az LCD kijelzőn természetesen nyomon követhetjük a kiválasztott programunkat. Ha meggondoljuk magunkat, és mégsem akarunk programot váltani, akkor a bankváltás után valamelyik potenciométer **Select** gombját nyomjuk meg. Ekkor semmilyen programváltás nem történik.

Miután kiválasztottuk a programot, állítsuk be, hogy melyik MIDI csatornán küldje az üzenetet. Ezt a **Chn+** vagy **Chn-** gombokkal tehetjük meg. Itt sincs körbejárási lehetőség. Csatornaszámnak 1-16 közötti értéket adhatunk. Csak olyan program esetén lesz hatásos, ami MIDI csatornához köthető. Jelenleg minden program ilyen, amit a készülék tartalmaz (kivéve a hangerőállítás az összes csatornán - 'Volume All' program), de megvalósítható olyan program is, ami nem köthető MIDI csatornához. Egyes szintetizátorok szoktak ilyet használni a különféle hangszintetizálási paraméterek beállításához. Ezek mindegyik eszközön egyediek, a szintetizátor dokumentációja tartalmazza ezeknek az üzeneteknek a kódjait. Általában „System Exclusiv” üzenetekkel szokták megvalósítani.

Preset+ vagy Preset-: A készülék több program-összeállítást képes eltárolni. Miért is jó ez? Tegyük fel, hogy először a sávok hangerőit szeretnénk állítani. Ekkor a 8 db potenciómétert beállítjuk, hogy „Main Volume” üzeneteket küldjön különböző csatornákon. Utána a hangszerek hangjának kórus-szinezetét szeretnénk szabályozni. Ehhez kiválasztjuk a kórus „Chorus” (Chorus Depth 5Dh) üzenetet. Beállítjuk a kívánt hangzást. Ha ezt követően megint a hangerőket szeretnénk módosítani, ilyenkor hasznos, hogy több program-összeállítást is eltárolhatunk. A műveletet egy gomb megnyomásával elvégezhetjük, nem kell az összes potencióméter programját újra beállítani. A készülék 10 ilyen program-összeállítást képes tárolni. Ezek a program-összeállítások akkor sem vesznek el, ha a készüléket kikapcsoljuk.

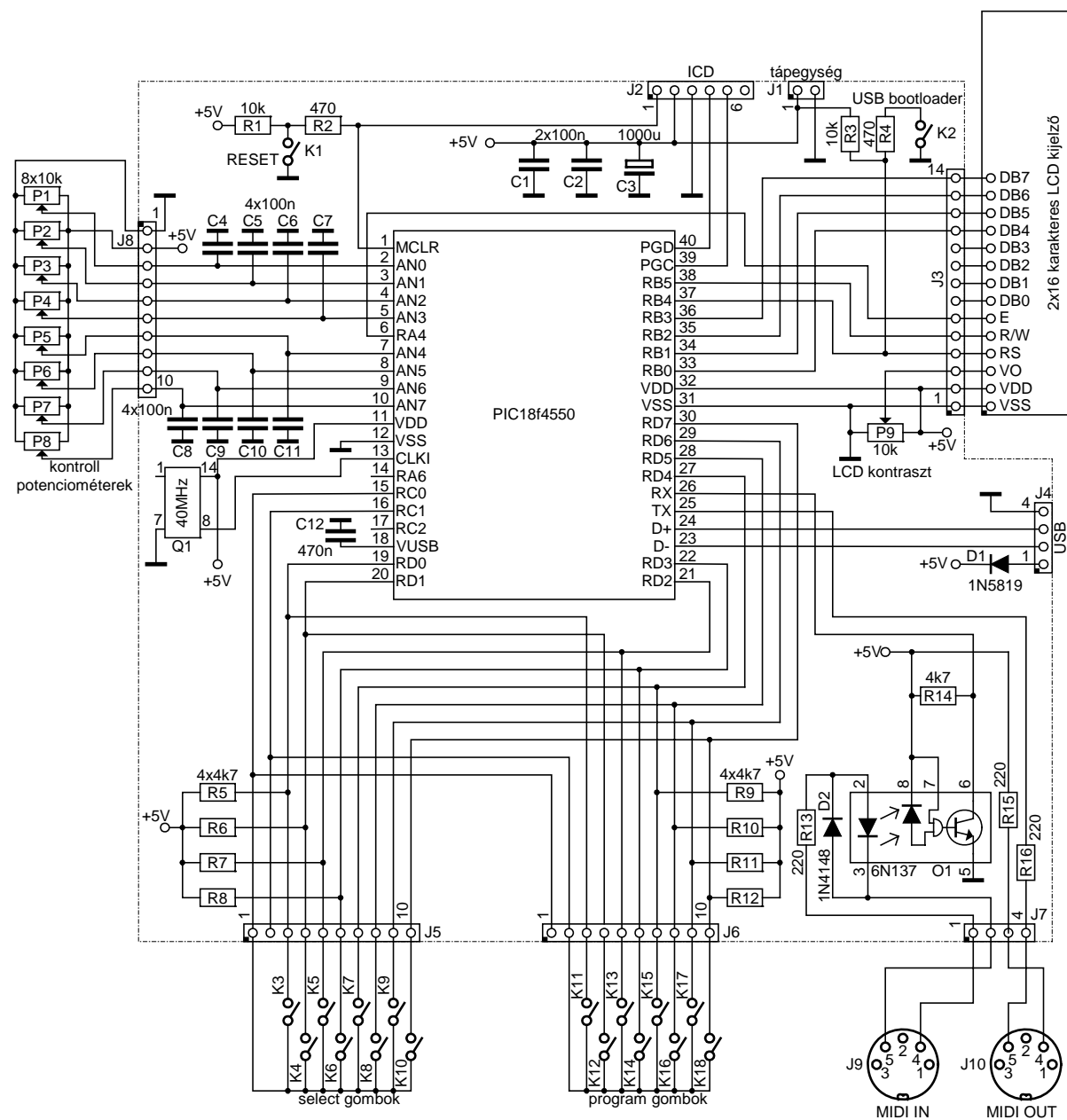
Használati példa:

- „Preset 1”-ben beállítjuk a potencióméterekhez a „Main Volume” programokat.
- Miután elvégeztük a keverést, megnyomjuk a **Preset+** gombot. Ekkor a „Preset 2”-re váltunk.
- A potencióméterekhez hozzárendeljük a kórus effekt üzenetet.
- Ha ezután ismét a csatornák hangerejét szeretnénk állítani, a **Preset-** gomb segítségével visszatérhetünk a „Preset 1” összeállításba. Ezzel visszkapjuk a „Main Volume” programokat.

A **System Reset** gomb megnyomásakor, egy System Reset üzenetet (FFh) küld a készülék a vevőnek. Ezzel minden kitartott hang elhallgat, valamint a hangparaméterek alapállapotba kerülnek.

6. A készülék hardvere

A készülék lelke egy, a Microchip által gyártott PIC18F4550 típusú, DIP tokozású, 40 kivezetéssel ellátott mikrovezérlő. Ebben a mikrovezérlőben minden szükséges perifériaillesztő áramkör megtalálható. A készülék kapcsolási rajza az alábbi ábrán látható:



12. ábra - A készülék kapcsolási rajza

A készülék a tápfeszültséget a J1 csatlakozón kapja. Stabil 5V-os feszültség szükséges a helyes működéshez. A tápfeszültség szűréséről a C1, C2, C3 kondenzátor gondoskodik.

A K1 reset nyomógomb csak a készüléken belül hozzáférhető, mert kizárólag a program fejlesztésekor van rá szükség. Az R1 ellenállás a reset bemenet magas logikai szinten tartásához szükséges. A nyomógomb megnyomásakor alacsony szintre kerül a bemenet. Az R2 áramkorlátozó ellenállásra azért van szükség, mert a reset bemenetet az ICD (In Circuit Debugger) is használja (J2 csatlakozón keresztül).

A K2 nyomógomb az R3 felhúzó ellenállással és az R4 áramkorlátozó ellenállással az USB letöltőprogram indítását szolgálja majd, a későbbi bővítés érdekében. Jelenleg még nem működik.

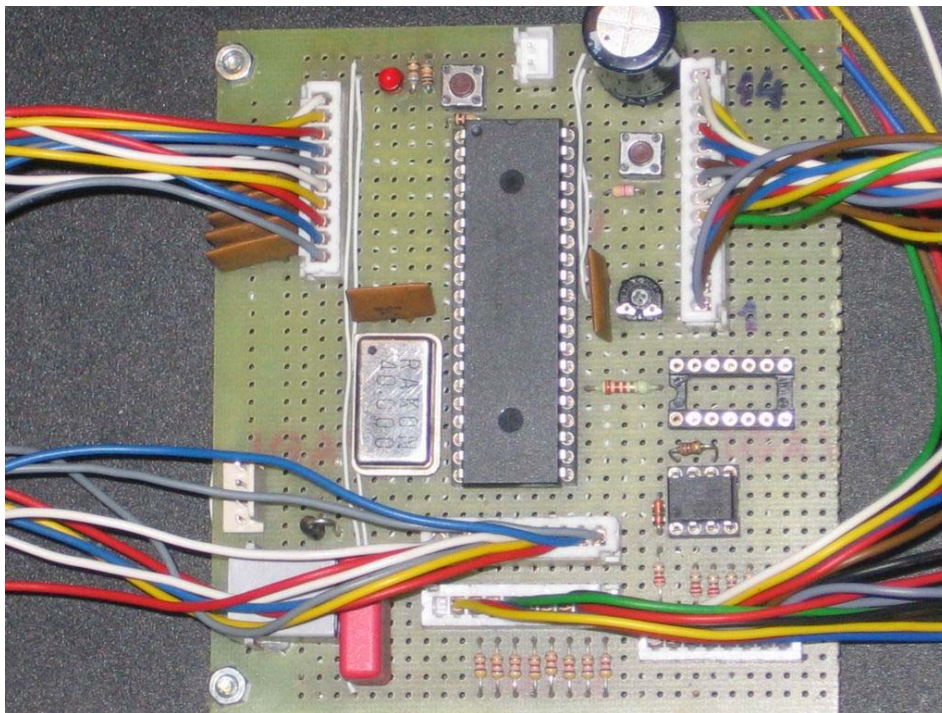
A J4 csatlakozón keresztül kapcsolható a készülék a számítógép USB portjára. A tápfeszültség ellátás a számítógépről is megoldható. A tápfeszültség nemkívánatos visszairányú áramát a D1 shottky dióda szünteti meg. Az alacsony nyitóirányú feszültségesés érdekében van szükség shottky diódára. Az adatlábak közvetlenül a mikrovezérlő D+ és D-lábaira kapcsolódnak. A C12 kondenzátor a mikrovezérlő 3,3V-os USB feszültségének szűréséhez szükséges.

Az órajelről a Q1 pozíciójú, 40MHz-es, 4 lábú kvarcoszcillátor modul gondoskodik.

A soros kimenet illesztőtranzisztor nélkül, az R16 áramkorlátozó ellenálláson keresztül kapcsolódik a kimenetre, ugyanis a mikrovezérlő maximális kimeneti árama 25mA lehet. Ezt az áramot még a MIDI kimenet esetleges rövidrezárása esetén sem érjük el. Az R15-nek úgyszintén kimeneti áramkorlátozó funkciója van. A bemenet az R13 áramkorlátozó ellenálláson keresztül jut az O1 optocsatoló LED-jére. A D2 dióda helytelen MIDI kábel bekötés esetén látja el a védelmet. Az optocsatoló kimeneti tranzisztorának kollektora csatlakozik a mikrovezérlő soros bemenetére. Az R14 az optocsatoló tranzisztorának kollektor ellenállása.

A 8db potenciométer (P1..P8) egyszerű feszültségosztóként van bekötve, az egyik vége a 0V-ra, a másik vége a +5V-os tápfeszültségre csatlakozik. A csúszkák kapcsolódnak a mikrovezérlő AN0..AN7 lábaira. Ezeket a lábakat analóg bemenetként használjuk. A C4..C11 kondenzátorokkal megvalósított aluláteresztő szűrők a potenciométerek esetleges kontakthibái miatt keletkező zavarokat csökkentik. A mikrovezérlőben csak egy db 10 bites A/D átalakító van. Azt, hogy melyik bemenetet kapcsolja az A/D átalakítóra, analóg kapcsolóáramkörrel valósítja meg. A bemenet kiválasztását ADCON regiszter 2.5 bitjeivel végezzük (CHS0..CHS3). A 4 bittel 16 bemenet közül választhatunk. A mikrovezérlő csak 13 analóg bemenettel rendelkezik. A maradék három kiválasztható ugyan, de az A/D átalakító bemenete sehová sem fog kapcsolódni. A készülékben csak az első 8 analóg bemenet van felhasználva.

A 16 nyomógomb 2x8-as mátrixba van kötve. A két mátrix sor a mikrovezérlő RC0 és RC1 lábaira csatlakozik. A potenciométer kiválasztó gombok az RC0, a többi nyomógomb az RC1-re van kötve. Valamelyik sorban levő 8 nyomógomb lekérdezésekor az aktuális sorhoz kapcsolódó lábat kimenetként programozzuk, és logikai alacsony (L) szintre állítjuk. A lenyomott nyomógombok az RD0..RD7 (bementként programozott) lábakon logikai alacsony (L) szintet eredményeznek. Az R5..R12 felhúzó ellenállások szerepe a felengedett nyomógomb esetén a logikai magas (H) szint biztosítása.



13. ábra - A készülék áramköri lapja

6.1. Az LCD kijelző

A 2x16 karakteres kijelző Hitachi HD44780 vagy azzal kompatibilis chip-et tartalmaz. A felhasznált 2x16 karakteres változat a kijelző család egyik tagja. Léteznek másféle (1, 2 és 4 soros, illetve 16, 20 és 40 karakter széles) változatok is. Eltérés csak a fizikai méretben, a háttérvilágításban és a kivezetések elhelyezkedésében van.

A meghajtó chip annyira elterjedt, hogy szinte az összes, a kereskedelemben kapható, hasonló tulajdonságokkal rendelkező kijelző (programozási és lábkiosztási szempontból) csereszabatos egymással. A család összes tagja 14+2 kivezetéssel van ellátva. A +2 kivezetésre a háttérvilágításért felelős LED-ek vannak kötve. Jelen alkalmazásban, a háttérvilágítás működtetése közvetlenül az LCD kijelző paneljén történt, áthidalással. A tápfeszültség az 1. és 2. lábra csatlakozik. A 3. láb a kijelzőn megjelenő karakterek kontrasztjának beállítására szolgál. Ide csatlakozik a P9 potenciométer csúszkája, egyszerű feszültségosztót képezve. A 4-6. kivezetések vezérlőlábak, a 7-14. pedig adatlábak. A jelenlegi 4 bites adatátvitel miatt csak a 11-14. lábak vannak bekötve.

A kijelző 3 vezérlő lábon, valamint 4 vagy 8 adatlábon keresztül programozható. Az adatvonal bitszélességét a kijelző inicializálásakor tudjuk kiválasztani. A 4 bites üzemmódban a DB0..DB7 adatlábakból csak a DB4..DB7 lábakat használjuk. Ebben az esetben természetesen egy 8 bites adat átvitele két lépésben történik meg, oly módon, hogy előbb a felső 4 bitet, majd az alsó 4 bitet vesszük át. A 3 vezérlőláb közül az R/W (Read/Write) lábon azt közölhetjük a kijelzővel, hogy írni szeretnénk (R/W = H) vagy olvasni szeretnénk belőle (R/W = L). Az RS (Instruction/Data Register Selection) lábon azt jelezzük, hogy az adat lábakon adatot, vagy utasítást szeretnénk továbbítani. Az L szint utasítást, a H szint adatot jelent. Az adatátvitel az E (Enable input) lábra adott vezérlés lefutó élénél történik.

6.2. A PIC18F4550 mikrovezérlő ismertetése

A PIC18F4550 típusú mikrovezérlő az amerikai Microchip Technology Inc. terméke. Weboldaluk címe: <http://www.microchip.com/> Termékeiket Magyarországon a ChipCAD Kft. forgalmazza. Weboldaluk címe: <http://www.chipcad.hu/>

A felhasznált típus a PIC18xxxx nyolcbites mikrovezérlő család egyik tagja. A nyolc bit azt jelenti, hogy 8 bites számokkal lehet aritmetikai és logikai műveleteket végezni. A műveleteket általában az akkumulátorral végezzük. A Microchip ezt a regisztert W-nek nevezte el. Az elnevezés a work (munka) szóból származik. Az utasításkészlet RISC jellegű, azaz viszonylag kis számú utasítás van. Az utasítások hossza (4 kivétellel) 16 bit, azaz az utasításkód az operandussal és a módosító bitekkel együtt legfeljebb ilyen hosszú lehet. Ezeket hívjuk egyszavas utasításoknak. A kivételt képező 4 utasítás 2x16 bit hosszúságú, kétszavas utasítás. Mivel a programszámláló bájtban tartalmazza az éppen végrehajtás alatt levő utasítás címét, minden egyszavas utasítás végrehajtásakor a PC értéke 2-vel növekszik. Minden utasítás páros címen kezdődik, ezért a PC regiszter legkisebb helyiértékű bitje mindig 0 értékű. A kétszavas utasítások második szavának kódja minden esetben 1111 bitekkel kezdődik. Azért van így, mert ez az üres utasítás (NOP) kódja, így ha egy kétszavas utasítás második szavára kerülne a vezérlés, üres utasításként fogja értelmezni.

Az adatokat és az utasításokat külön memória tárolja. Ezt Harvard architektúrának hívjuk, aminek előnye, hogy eltérő lehet a program és az adatmemória bitszélessége. A család 16 bit széles programmemóriával és 8 bit széles adatmemóriával rendelkezik. A programmemória maximális mérete 2MB, ami nagyjából 1 millió egyszavas utasítást jelent. Az adatmemória mérete ehhez képest igen szerény, maximum 4096 bájt. Az operandusban az adatmemória megcímezéséhez viszont mindössze 8 bit van fenntartva, ezért az utasításban 256 bájt adatmemória címezhető meg. Az ellentmondás feloldásához a lapozásos technikát alkalmazták. A külön program és adatmemória másik előnye, hogy megszűnik a „sorbanállás” a memóriáért, mert egy időben elérhető a program és az adatmemória.

A Harvard architektúrának azért hátránya is van:

- Nem osztható fel a memória tetszés szerinti arányban adat ill. programterületre.
- Nehézkes a programból a programmemória tartalmát elérni.

A hardver verem 31db 21 bit szélességű rekeszből áll. Szubrutinhívás és megszakítás esetén ide kerül elmentésre a PC tartalma, a rutin végén pedig innen töltődik vissza.

A felhasznált PIC18F4550 típusú mikrovezérlőben található programmemória mérete 32768 bájt. Ebben 16384 db egyszavas utasítás fér el. A programmemória Flash típusú. Ez azt jelenti, hogy a programmemória tartalma egyszerűen módosítható. A módosításra akár programból is van lehetőség, viszont figyelembe kell venni, hogy a programmemória törlése csak 64 bájtos egységekben, az írása pedig csak 8 bájtos egységekben lehetséges. Íráskor a beírt bájt először egy belső 8 bájtos pufferbe kerül. Miután megtöltöttük a 8 bájtos puffert, csak azután lehet átírni a Flash memóriába. Olvasni bájtonként is lehetséges. Az adatmemória SRAM típusú és 2048 bájt méretű. A mikrovezérlőt kiegészítették még 256 bájt adat EEPROM memóriával. Ez bájtonként írható/olvasható. Az EEPROM típusú memória elektromosan törölhető és újraírható, de csak korlátozott számban. A gyártó a Flash program memóriára 100 000, az adatmemóriára 1 000 000 törlési/újraírási ciklust vállal.

Nem létezik külön a perifériák elérésére szolgáló utasítás. A perifériák elérése memóriába ágyazott módon lehetséges, mint általában a RISC jellegű processzoroknál. A perifériák elérésére szolgáló memóriaterületet SFR regisztereknek, azaz Speciális Funkciójú Regisztereknek nevezzük. Az SFR regiszterek kiosztását a Microchip határozta meg. A perifériákhoz szükséges regisztereken túl vannak köztük a processzor állapotát tartalmazó, lapválasztó, indirekt címezést megvalósító regiszterek is (pl. a jelzőbiteket a STATUS nevű regiszter tartalmazza).

Az utasításkészlet öt fajta utasításcsoportra bontható fel:

1. **Byte-oriented operations** (bájt orientált műveletek): A művelet egyik operandusa az utasítás operandusában kijelölt adatmemória tartalma, ha szükséges másik operandus az pedig a W regiszter. Az utasítás módosító bitjeitől függ, hogy az eredmény a W regiszterben vagy pedig az adatmemóriában keletkezik. Ezek között található egy 2 szavas utasítás is, a MOVFF, ami egy adatmemória rekesz tartalmát átviszi egy másik adatmemória rekeszbe. Az utasítás különlegessége, hogy a forráscím és a célcím is 12 bites, ezért nem kell törődni a lapozással.

2. **Bit-oriented operations** (bit orientált műveletek): Mivel gyakran van szükség elemi bitekkel történő műveletvégzésre (főleg a perifériáknál), előnyös, ha egy utasítással tudunk egy bitet 0-ra vagy 1-re állítani, esetleg ellentétesre váltani. A BCF, BSF, BTG utasításokkal ezt anélkül tehetjük meg, hogy a regiszter többi bitjeivel foglalkoznunk kellene. A BTFSC és BTFSS utasításokkal lehetőségünk van egy bit állapotától függően a soron következő utasítást kihagyni.
3. **Control operations** (vezérlő műveletek): Ide tartoznak a különböző feltételes és feltétel nélküli vezérlésátadó utasítások, a szubrutinhívások, a szubrutinból vagy megszakításból visszatérést eredményező, a veremkezelő, az alvó üzemmódot bekapcsoló, a processzort resetelő, a watchdog-timert nullázó és az üres (NOP) utasítások. Ebben a csoportban kettő kétszavas utasítás található. Az egyik a GOTO, a másik a CALL utasítás. Ezeknél az egyszavas utasításba már az operandus sem fér bele, hiszen a PIC család programmemória címezése 21 bites. Mint említésre került, a programszámláló mindig páros értékű, ezért az operandusban nem kell a legalsó bitnek helyet biztosítani, így csak 20 bit az operandus hossza. Ennek a két utasításnak létezik rövid, egyszavas verziója is (BRA, RCALL). Ezeknél az operandus 11 bites relatív ugrási ill. szubrutinhívási címet tartalmaz.
4. **Literal operations** (konstans értékkel dolgozó utasítás): az egyik operandus a W regiszter, a másik pedig az utasítás operandusában található 8 bites szám, az eredmény pedig a W-ben keletkezik, két kivétellel. Az egyik kivétel a MOVLB utasítás, ami a lapcím betöltésére szolgál. Ez a PIC család 4096 adatbájt címezésére képes, ez pedig 16 lapon fér el. Ebből következik, hogy 4 bites értékkel tölti fel a lapválasztó regisztert (BSR). A másik kivétel az LFSR utasítás, amivel a három indirekt címregiszter valamelyikét tölthetjük fel egy 12 bites értékkel. Ez az utasítás 2 szavas.
5. **Data memory - program memory operations** (adatmemória - programmemória közötti műveletek): ezzel a 8 utasítással valósíthatjuk meg a programmemória és az adatmemória közötti adatátvitelt. Lényegében az olvasó és az író utasítás 4-4 változatáról van szó. A változatok közötti különbség mindössze annyi, hogy a programmemória mutató csökkentésével vagy növelésével van összevonva az utasítás.

7. A Program felépítése

7.1. A fejlesztői környezet

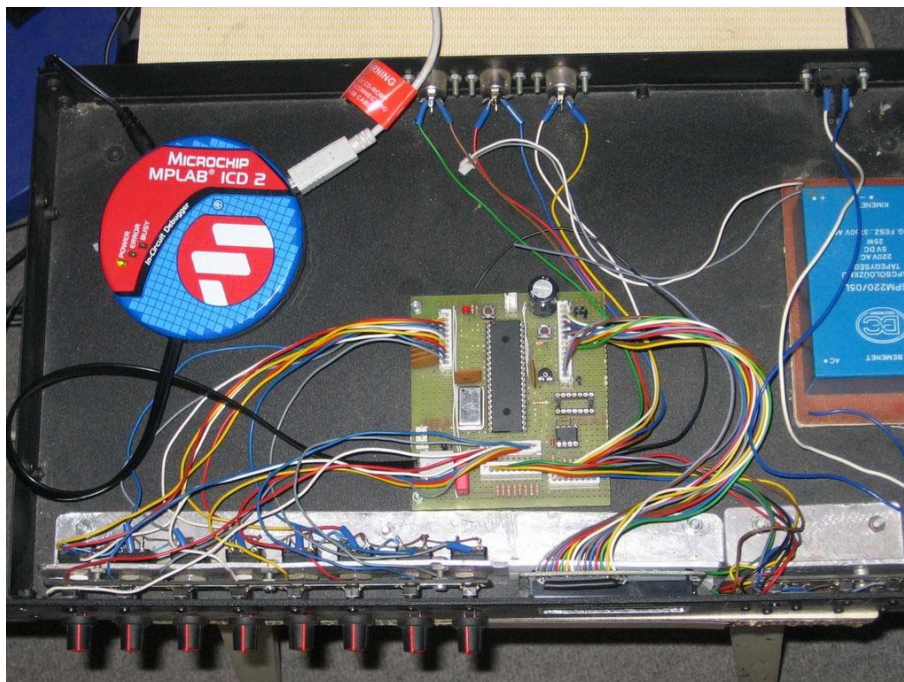
A program a Microchip MPLAB V7.50 fejlesztőkörnyezetét felhasználva készült assembly nyelven. A fejlesztői környezet mindent tartalmaz, ami a fejlesztéshez szükséges. Ezt integrált fejlesztőkörnyezetnek (Integrated Developping Enviroment – IDE) nevezzük. Az MPLAB telepítője a Microchip honlapjáról letölthető (<http://www.microchip.com>).

A fejlesztői környezet a következőket tartalmazza:

- Editor: Az assembly forráskód elkészítéséhez.
- Fordító és linker: A forráskódból a gépi kódot előállítani.
- Szimulátor: A programok teszteléséhez.
- Picstart Plus kezelőprogramja: A mikrovezérlőbe ennek segítségével lehet a programot beírni.
- ICD2 kezelőprogramja (In Circuit Debugger 2): Szerepe, hogy a programot a valós környezetében lehessen tesztelni.

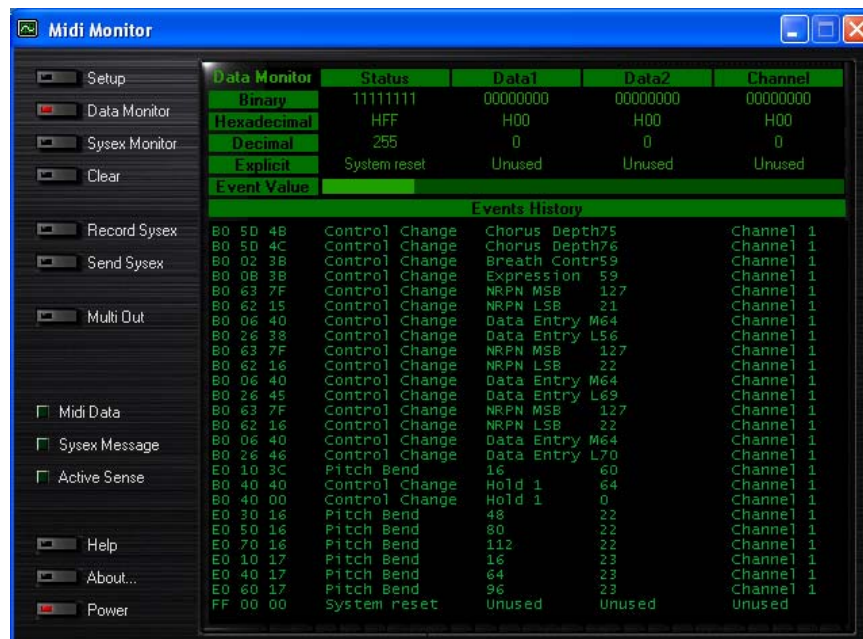
A Picstart Plus és az ICD2-n kívül még számos programozó és debugger eszközt támogat.

A program fejlesztése az ICD2 felhasználásával történt.



14. ábra - Fejlesztés az MPLAB ICD2-vel

A készülék tesztelését egy MIDI monitor program segítette.



15. ábra - MIDI monitor program az üzenetek teszteléséhez

A program több modulból épül fel. A fordító a modulokat külön-külön fordítja le. A fordítás eredményei az object fájlok. Az objectek már gépi kódot tartalmaznak, de a címhivatkozások még nincsenek kitöltve bennük. Ezeket majd a linker fogja elkészíteni, mikor az object fájlokból összeállítja a végleges tárgykódot. A több modul használata azért célszerű, mert ha a feladatot több részfeladatra osztjuk, és ezeket külön modulokba tesszük, sokkal áttekinthetőbb lesz a program. A modulok közötti kapcsolatot interfészekon keresztül biztosíthatjuk. Az interfész megvalósítása egyszerű. Ha azt szeretnénk, hogy a modul valamelyik szubrutinjához vagy adatához másik modul is hozzáférjen, a szubrutinok ill. adatok neveit soroljuk fel egy „GLOBAL” direktíva után. Ha egy modulban hivatkozni szeretnénk egy másik modulban található szubrutinra vagy adatra, akkor pedig az „EXTERN” direktíva után soroljuk fel az elérni kívánt szubrutinok ill. adatok neveit.

Nézzünk erre egy példát, ahol a „String.asm” modulban levő **CurPos** változót és **WriteChar** szubrutint szeretnénk a „Keyb.asm” modulból elérni.

String.asm

GLOBAL CurPos, WriteChar ;ezek másik modulból is elérhetőek

Keyb.asm

EXTERN CurPos, WriteChar ;ezek másik modulban vannak

7.2. A program feladatának meghatározása:

Vizsgáljuk meg, hogy milyen be és kimeneti perifériákat kell kezelni:

- Soros bemenet (Usart RX)
- Soros kimenet (Usart TX)
- LCD kijelző (3 kimenet, 4 ki/bemenet)
- 8 db analóg bemenet A/D átalakítása (8 analóg bemenet)
- 16 db nyomógomb (2 ki, 8 bemenet)

Csoportosítsuk a feladatokat:

- Usart RX eljárások (soros adat vétel)
- Usart TX eljárások (soros adat küldés)
- LCD kijelző és stringkezelő eljárások
- Billentyűzetkezelő eljárások
- Potenciométer (A/D) eljárások
- Beállítások adat EEPROM-ba mentése

Határozzuk meg, hogy milyen feladatot milyen esemény bekövetkezésekor kell elvégezni. A rendszeres lekérdezést igénylő perifériákat „polling” módon, a Timer0 időzítő megszakítását felhasználva alkalmazzuk.

- Perifériák inicializálása: Bekapcsolás után vagy reset után.
- Usart RX megszakítás-kiszolgálás: Soros port Rx megszakításakor.
- Usart TX megszakítás-kiszolgálás: Soros port Tx megszakításakor.
- Billentyűzet lekérdezése: Timer0 időzítő megszakításakor.
- Potenciométerek (A/D) lekérdezése: Timer0 időzítő megszakításakor.
- ActivSens MIDI üzenet elküldése: Timer0 időzítő megszakításakor.
- LCD kijelző frissítése: Szövegkiírás után.
- Beállítások adat EEPROM-ba mentése: Beállítások megváltozása után.

Bekapcsolás után a processzor a nulladik memóriacímtől indul. Először az **Init** szubrutin inicializálja a perifériákat, majd betölti az adat EEPROM-ból az utoljára mentett beállítást. Ez a betöltés egyben a változókat is beállítja. Ezután lehet a globális megszakítást engedélyezni. A főprogram ezután végtelen hurokban kering. Ebben a hurokban nyert elhelyezést két feltételesen végrehajtandó szubrutin.

Az egyik szubrutin az LCD kijelzőn levő felirat frissítésére, a másik a programösszeállítás mentésére szolgál. A feltételt két darab jelzőbit (**LcdRefresh**, **Save**) értéke adja. Ez a két funkció azért került a fő programhurokba, mert esetükben a perifériára várakozás olyan hosszú idejű is lehet, ami a többi funkcióban zavart okozhat. Ugyanis az LCD kijelzőt az adatok küldése előtt le kell kérdezni, hogy készen áll-e az adat fogadására. Nemleges válasz esetén várni kell, nem küldhetünk adatot, különben hiányos lehet a megjelenített szöveg. Mivel ez a főprogramban fut, ez a várakozási ciklus megszakítható. A kiírandó szöveg egy 32 bájt hosszúságú tömbben (**LcdString**) található. A megszakításban futó programrutinok (a stringkezelő szubrutinokat igénybe véve) ebbe a memóriatömbbe helyezik el megjelenítendő szöveget. Ha a teljes szöveg bekerült ebbe a tömbbe, az **LcdRefresh** jelzőbittel jelezzük, hogy megjeleníthető. A főprogram **StringToLcd** eljárása csak ekkor fogja a kijelző teljes tartalmát frissíteni.

A másik, hosszabb időt igénylő tevékenység az adat EEPROM írási művelete, azaz a programösszeállítás mentése. Ezt a **PresetSave** eljárás végzi. A mentés itt a **Save** jelzőbittel történő értesítés alapján történik.

A MIDI protokoll 31250 bit/másodperces átviteli sebességet ír elő. Ez azt jelenti, hogy 1 másodperc alatt kb. 3000 byte adat továbbítható rajta a start és stop biteket is figyelembe véve. Ebből kiindulva 320 mikroszekundumonként számíthatunk arra, hogy az Usart RX bemenetre adat érkezik. Ennyi idő alatt (40MHz-es processzor órajellel számolva) $320 \cdot 10 = 3200$ egyciklusú utasítás hajtódik végre. Ez első ránézésre elég soknak tűnik, de látni fogjuk, hogy egy hosszabb MIDI üzenet generáláskor szükség is van erre a sebességre. Ugyanis egy üzenet előállításához két (ciklust is tartalmazó) szubrutin végrehajtása szükséges: Az üzenet generálása, valamint az üzenethez tartozó felirat LCD kijelzőn történő megjelenítése. Itt van kihasználva a PIC18x sorozatban alkalmazható 2 szintű megszakítás lehetősége. Bármely periféria megszakítását két megszakításvektorhoz rendelhetjük. Létezik egy magas prioritású megszakítás vektor, valamint egy alacsony prioritású megszakítás

vektor. Ennek lényege, hogy az alacsony prioritású megszakítás kiszolgáló rutin futását megszakíthatja egy magas prioritású. Fordítva ez persze nem lehetséges.

A prioritást úgy kell szerveznünk, hogy a soros vonalon beérkező adatoknál ne történjen adatvesztés (a vevő a vételről nem küld az adónak visszaigazolást). Ezért a magas prioritást kizárólag a soros vétel (Usart RX) kapta. A többi megszakítás-forrás az alacsony prioritáshoz lett rendelve. A magas prioritású megszakítást egy esetben mégis le kell tiltani: Saját üzenet készítésekor, amikor elkezdjük az Usart TX felé küldeni az adatokat. Ugyanis ekkor ugyanabba a FIFO regiszterbe küldjük tovább az adatokat, mint az Usart RX eljárása is tenné. (Ha nem tiltanánk le, előfordulhatna, hogy a saját MIDI üzenet közepébe belekerülne egy bemenetre érkezett MIDI üzenet.)

Alacsony szintű megszakítással három periféria kezelése van megvalósítva.

A billentyűzet, és az A/D átalakítók (potenciométerek) lekérdezése a Timer0 számláló túlcsordulása (3,2 milliszekundumonként, azaz másodpercenként 305-ször) esetén következik be. A megszakításkezelő program minden kilencedik alkalommal fogja a billentyűzetet lekérdezni, míg a többi 8 alkalommal egy-egy A/D átalakítóval foglalkozik. Ebből kifolyólag $305/9 = 34$ -szer kerül sor a billentyűzet lekérdezésére, valamint az egyes A/D átalakítókra kötött potenciométerekre. Ezért, ha valamelyiket folyamatosan forgatjuk, 34 MIDI üzenetet fog a készülék küldeni másodpercenként. Azt is mondhatjuk, hogy a mintavételezési frekvencia 34 Hz. Ez elég finom beállításra ad lehetőséget, de a rendszert nem terheli fölöslegesen sok üzenettel.

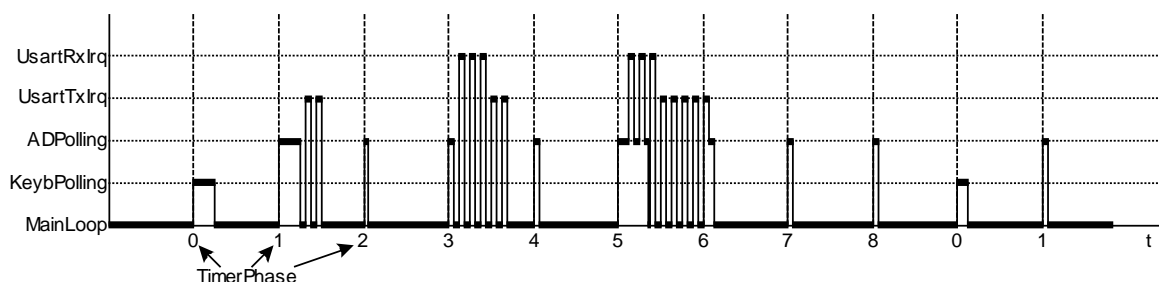
A billentyűzet gombjait három csoportba sorolhatjuk:

- Egy nyolcas csoport szolgál a potenciométer kiválasztására (melyik programján szeretnénk a módosítást végrehajtani).
- Egy hetes csoportot használhatunk a programok módosításához.
- Egy nyomógomb pedig a **System Reset** üzenet küldésére szolgál. Ezzel lehet alapba állítani a hangszer.

A harmadik periféria az Usart TX (soros port kimenet). Ha adatot akarunk küldeni, az első bájtot betesszük a küldőpufferbe (**TXREG**). Az első bájt azonnal átkerül a küldő shiftregiszterébe (**TSR**), így a második bájtot is be tudjuk tenni a **TXREG**-be. A további bájtokat **TXREG** felszabadulásáig a szoftveres FIFO tárolóba tesszük. Minden bájt elküldése után a soros adó alacsony prioritású megszakítást fog kiváltani. Ilyenkor a FIFO-ból kiolvasva küldjük a következő bájtot, amíg az ki nem ürül. Ha kiürült, leállítjuk az adatok küldését.

Magas prioritású megszakítást használunk a soros adatok vételére (**Usart RX**). Ez azért fontos, mivel a MIDI adatfolyam nem tartalmaz sem hibajavító információt, sem visszajelzést az adó irányába. A magas prioritású megszakítás lényege, hogy az adatvesztés elkerülése érdekében a program azonnal a beérkezett adat feldolgozásával foglalkozhat. Igaz ez abban az esetben is, ha ezért egy alacsony prioritású megszakítás kiszolgáló eljárást kell félbeszakítani.

Vizsgáljuk meg a következő esetet:



16. ábra - Futási idődiagram

A függőleges tengelyen az eljárások megnevezése, a vízszintesen az idő látható. Az időtengelyen található számok a lekérdezési fázisokat jelentik. Haladjunk végig balról jobbra a folyamaton:

A lekérdezéseket (Polling) a **Timer0** számláló túlszordulása váltja ki 3,2 milliszekundumonként.

A 0. fázisban a billentyűzet lekérdezése történik. Most hosszabb ideig fut, mert valamelyik lenyomott nyomógomb eseményét fel kell dolgoznia.

Az 1. fázisban az első potenciométerre kapcsolt A/D átalakító lekérdezése történik. Láthatjuk, hogy mozgatva volt, mert hosszabb időt tölt el itt a program. Midi üzenetet is generál. Az üzenet első 2 bájtját közvetlenül be tudja tenni a soros periféria regiszterébe, a 3. bájt azonban már nem fér bele, ezért FIFO tárolóba kerül. Amikor a soros periféria

ismét képes új adatot küldeni (**TXREG** kiürült), megszakítást generál. Ekkor az **UsartTxIrq** eljárás kiolvassa a FIFO-ból az üzenet 3. bájtját, és beírja a soros port küldő regiszterébe. A következő bájt elküldése után a **TXREG** ismét kiürül, ezért újabb megszakítást generál. **UsartTxIrq** viszont már nem tud a FIFO-ból adatot olvasni, mert az már üres. Ekkor leállítja az adást.

A 2. fázisban lekérdezi a második potenciométer állását, de változás hiányában nem történik semmi.

A 3. fázisban a harmadik potenciométer állását olvassa be, de az sem változott. Viszont ekkor a soros bemenetre érkezik egy 3 bájtos MIDI üzenet. Minden beérkezett bájt megszakítást vált ki, amit **UsartRxIrq** eljárás dolgoz fel. Miután az üzenet utolsó bájtja is megérkezett, a teljes üzenet továbbítódik a soros kimenetére. A küldés menete az 1. fázisban leírt módon történik.

A 4. fázisban ismét továbblépünk a következő potenciométerre. Itt sincs semmi változás, ezért hamar végez a feladattal.

Az 5. fázisban érdekes szituáció alakult ki. A potenciométer állását beolvassa kiderül, hogy az változott, ezért MIDI üzenetet kell küldeni. Az üzenet bájtjainak küldése alatt a megszakításokat letiltjuk, hogy egy, a soros bemenetre érkezett üzenet ne keveredhessen bele a saját üzenet bájtjai közé. Amint befejezte az üzenet bájtjainak elküldését, megszüntetjük a megszakítás letiltását. Ezalatt az idő alatt viszont érkezett a bemenetre egy bájt. Így mielőtt az LCD kijelzőre kiírandó szöveget összeállítaná, átkerül a vezérlés a magas megszakítási prioritással rendelkező **UsartRxIrq**-ra. Miután az befejezte a bemenetre érkezett adat feldolgozását, az **ADPolling** folytathatja az LCD-re írandó szöveg összeállítását. Mielőtt végezne vele, beérkezik a 2. bájt is. Az **ADPolling** újból félbeszakítja a munkát és ismét az **UsartRxIrq** foglalkozik a beérkezett adattal. Ha végzett, visszakerül a vezérlés az **ADPolling**-hoz, ami végre befejezheti a munkáját. Időközben megérkezik a bejövő üzenet 3. bájtja is. Ekkor az **UsartRxIrq** továbbküldi a soros kimenetre a most már teljes 3 bájtos üzenetét. Mivel az adópuffer foglalt (a már elindított adás miatt), a teljes üzenet a FIFO-ba kerül. Közben a **TXREG** is felszabadul, ezért megszakítást generál. A megszakítást kiszolgáló **UsartTxIrq** pedig küldi a FIFO-ból kiolvasott következő adatot. Ez addig tart, amíg a FIFO ki nem ürül.

A 6. fázisban befejeződik az előző fázisban elkezdődött esemény. Ezt követően lekérdezi a hatodik potenciométer állását, de változás hiányában nem történik semmi.

A 7. fázisban lekérdezi a hetedik potenciométer állását, de változás hiányában most sem történik semmi.

A 8. fázisban eljutunk az utolsó potenciométer lekérdezéséhez, de változás hiányában most sincs végrehajtandó feladat.

Ezután ismét a 0. fázis következik, és a lekérdezési folyamat kezdődik előlről.

7.3. A főprogram

A főprogram a „Main.asm” modulban kapott helyet.

Ez tartalmazza a Reset, valamint a 2 megszakítás vektor programját. Ezek címei:

- Reset vektor : 0000h.
- Magas prioritású megszakítás vektor: 0008h.
- Alacsony prioritású megszakítás vektor: 0018h.

Bekapcsoláskor először a 0000h címen található utasítást hajtja végre a mikrovezérlő. Innen ugrik a **Main** címkevel ellátott utasításra, mely tulajdonképpen a főprogram kezdete. Itt meghívódik az **Init** szubrutin, ami sorban meghívja a többi modulban található, a perifériák inicializálását végző eljárásokat. Ezek a következők:

- **LcdInit**: Inicializálja az LCD kijelzőt.
- **ADInit**: Inicializálja az analóg-digitális átalakítót és beolvassa a nyolc analóg bemenet.
- **UsartInit**: Beállítja a soros port átviteli sebességét (31250 bit/másodperc) és az üzemmódját (aszinkron 8 bites, paritás nélküli). Inicializálja a FIFO regisztert a **FifoInit** eljárással.
- **KeybInit**: Inicializálja a billentyűzetkezelő eljárás változóit, beállítja a készülék utoljára használt programjait.

Ezután elvégzi a **Timer0** időzítő beállítását, valamint a megszakítások engedélyezését.

A program ezután végtelenített hurokban fut, ahol szükség esetén elvégzi az LCD kijelző frissítését (**StringToLcd**), valamint az összeállítás mentését (**PresetSave**). A szükségességet az LCD kijelző esetén az **LcdRefresh**, a beállítások mentése esetén a **Save** jelzőbit tartalmazza. Az összes programrutin ezen biteknek az 1-re állításával jelzi, ha az LCD kijelzőre új szöveget kell kiírni, vagy az összeállítást el kell menteni, mert módosultak a beállítások.

A magas prioritású megszakítást kiszolgáló eljárás az **UsartRxIrq** eljárást hívja meg. Itt nem szükséges vizsgálni a megszakítás forrását, mert csak a soros bemenethez lett magas prioritás rendelve.

Alacsony prioritású megszakítást (**LowInt**) több periféria is kiválthat. A megszakítás forrását jelzőbitek lekérdezésével dönthetjük el. Ha **PIR1**, **TXIF** bit értéke 1, a soros kimenet, ha **INTCOM**, **TMR0IF** bit értéke 1, az a **Timer0** megszakítását jelzi. A soros kimenet megszakításakor **UsartTxIrq** eljárást hívjuk meg. A **Timer0** megszakítását a rendszeres lekérdezésekre (Polling) használjuk. Ekkor a **TimerPhase** változó nulla értéke esetén a billentyűzetet (**KeybPolling**), egyébként az A/D átalakítót (**ADPolling**) kérdezzük le. Minden lekérdezési ciklusban csökkentjük **TimerPhase** értékét, ha negatív lesz, akkor nyolcat írunk bele. Így kilenc ciklus alatt érünk körbe. Az „Active Sens” üzenet elküldése is itt valósul meg, minden 93-adik ciklusban, azaz 300 milliszekundumonként..

A modulban található kettő, időzített várakozást megvalósító eljárás, melyek az LCD kijelző inicializálása során vannak felhasználva:

- **Delayusec:** 1..255 mikroszekundumos várakozás. Hívásakor a várakozás idejét a W regiszterbe kell betenni.
- **Delaymsec:** 1..255 milliszekundumos várakozás. Hívásakor a várakozás idejét itt is a W regiszterbe kell betenni.

A **PresetSave** eljárás (összeállítás mentése) is ebben a modulban van kifejtve. Feladata az **ADBanks**, **ADPrgs** és **ADChns** tömbök, valamint a **PresetNum** változó, adat EEPROM megfelelő helyére történő bemásolása. Ennek során használjuk az **EeWrite** eljárást, amely egy bájtot beír az adat EEPROM-ba.

7.4. A MIDI üzenetek táblázatai a program memóriában

A MIDI üzeneteket leíró táblázatok a „Programs.asm” modulban találhatóak. A táblázatok a mikrovezérlő programmemóriájában helyezkednek el. Három féle leíró táblázat van. Ezek három mélységi szintet alkotnak, fa struktúrába szervezve.

A legfelső szinten található a „bankok” táblázata (**PrgBanks**).

PrgBanks	BankNum (4)
	PrgBank 0 Ptr
	PrgBank 1 Ptr
	PrgBank 2 Ptr
	PrgBank 3 Ptr

17. ábra – A bankok táblázata

Ebből csak egy darab van. Tartalmazza a bankok számát, valamint a bankok táblázatainak mutatóit a programmemóriában. A fenti példában 4 bank létezik, ezért a BankNum mező értéke: 4. Utána következik a 4 mutató, ami a 4 bank táblázatának kezdőcímeit tartalmazza.

A középső szinten a programbank táblázatok találhatóak (**PrgBank[n]**),

PrgBank 0	BankSize 0 (4)
	Prg 0 Ptr
	Prg 1 Ptr
	Prg 3 Ptr
	Prg 2 Ptr
	"Normale Bank", 0

18. ábra - Egy programbank táblázat

Ebből már több példány is lehet. Felépítésük hasonló, mint a legfelső szinten lévő „bankok táblázata”, azzal a különbséggel, hogy a táblázat végén egy string is található, ami a bank nevét tartalmazza. A készülék bizonyos esetben a stringben található szöveget írja ki az LCD kijelzőre. A stringet #0 karakterrel kell lezárni. A hossza minden esetben 16 karakter, hogy kitöltse az LCD kijelző egyik sorát. A mutatók fogják megcímezni a programok táblázatait. Ugyanaz a mutató több bankban is felhasználható, ekkor az adott program több bankban is elő fog fordulni. Egy bankon belül is többször felhasználható ugyanaz a mutató, bár ez értelmetlen. A példában szereplő banknak „Normale Bank” a neve, és négy mutatót (programot) tartalmaz.

A legalsó szinten vannak program táblázatok. Minden ilyen táblázat egy MIDI üzenetnek felel meg, és 4 mezőre lehet felosztani:

- **Sens:** Ez mondja meg, hogy mekkora potenciométer-elfordulás esetén szükséges MIDI üzenetet küldeni.
- **Size:** A MIDI üzenet hossza (ennyi bájt).
- **Msg:** Az elküldendő üzenet bájtjai. Minden elküldendő bájt két bájtól állítunk elő, az első a bájt típusát, a második a tartalmát határozza meg. Ha a forrásprogramba szavanként írjuk be ezt a két értéket, akkor előre kell írni a tartalmat, majd mögé a típust. Pl. B001h érték esetén a tartalom=B0h, a típus=01h. A típus az alábbi módon befolyásolja a tartalmat:
 - 00h: A tartalom változatlan marad.
 - 01h: A tartalom értékéhez hozzá kell adni a beállított csatorna számát (0..15)
 - 02h: A tartalmat az A/D konverzió 7 bites felbontására átalakított értéke adja (0..127)
 - 03h: A tartalmat a következő módon képezzük: Az A/D konverzió 10 bites értékét 7 bites felbontásra alakítjuk. Az így kapott számot kiegészítjük 14 bitesre és hozzáadunk 8128-at. A kapott szám felső 7 bitje lesz a tartalom (3Fh vagy 40h).
 - 04h: A tartalmat a következő módon képezzük: Az A/D konverzió 10 bites értékét 7 bites felbontásra alakítjuk. Az így kapott számot kiegészítjük 14 bitesre és hozzáadunk 8128-at. Most a kapott szám alsó 7 bitje lesz a tartalom.
 - 05h: A tartalmat az A/D konverzió 14 bites felbontására átalakított értékének a felső 7 bitje adja.
 - 06h: A tartalmat most az A/D konverzió 14 bites felbontására átalakított értékének az alsó 7 bitje adja.
 - 07h: A tartalom az A/D konverzió eredményétől függően 0 vagy 64 lesz (kapcsoló típusú üzenetekhez). Részletes leírása a **Damper Pedal** üzenet kifejtésénél található.
 - 08h..FFh: Bővítéshez fenntartva.

- **Text:** Ez tartalmazza az LCD kijelzőre kiírandó szöveget, amit a #0 lezáró karakterig kell a kijelzőre kiírni. Az F0h..FFh közötti tartomány vezérlőkaraktereket tartalmaz. A vezérlőkarakterek a következők lehetnek:
 - F0h: A potenciométer számát kell kiírni (1..8).
 - F1h: A MIDI csatorna számát kell kiírni (01..16).
 - F2h: A potenciométer állását (előjel nélkül, 7 biten) kell kiírni (0..127).
 - F3h: A potenciométer állását (előjelesen, 7 biten) kell kiírni (-64..63).
 - F4h: A potenciométer állását (előjel nélkül, 14 biten) kell kiírni (0..16383).
 - F5h: A potenciométer állását (előjelesen, 14 biten) kell kiírni (-8192..8191).
 - F6h: A potenciométer kapcsolóként értelmezett állását kell kiírni (off, on).
 - F7h..FFh: Bővítéshez fenntartva.

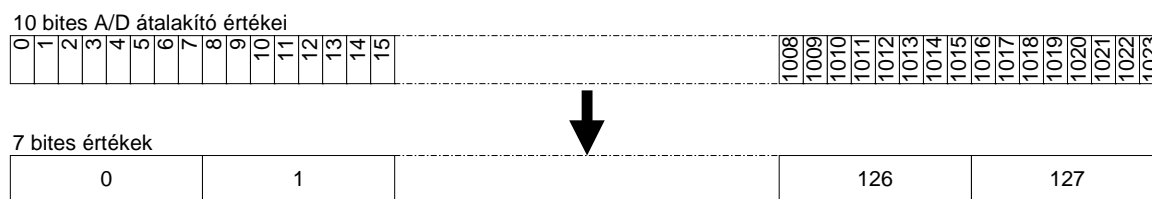
Nézzük meg néhány konkrét üzenet táblájának megvalósítását.

A "Volume" üzenet:

PrgVolume	Sens : 8
	Size : 3
	Msg: B001h, 0700h, 0002h
	Text: F0h, ".Volume ", F2h, " c", F1h, 0

19. ábra – A Volume üzenet táblázata

A Sens: (8) az érzékenység. Ez azt jelenti, hogy a 10 bites A/D átalakítással kapott értéknek legalább 8-al meg kell változnia az előzőleg elküldött üzenethez tartozó A/D értékhez képest. Így nem fordul elő, hogy kicsi potenciométer forgatáskor feleslegesen elküldjük ugyanazt az üzenetet. Nézzük meg, hogy a 10 bites A/D átalakítás során kapott értékekhez hogyan rendelhetjük hozzá a 7 bites értékeket:



20. ábra - 7 bites hozzárendelés

Láthatjuk, hogy ha az A/D átalakítás értékének változása legalább 8, akkor már az üzenethez is más érték tartozik.

A **Size:** (3) a méret. A MIDI üzenet elküldendő bájtjainak a száma. Egy számlálónak ezt adjuk kezdőértéknek, majd minden elküldött bájttal csökkentjük a számláló értékét. Ha ez eléri a 0-t, akkor vége az üzenetnek.

Az **Msg** az üzenet adatai. A hangerő állítás üzenet a Control Change üzenettípus egyike. Az üzenet első bájtja az első MIDI csatorna esetén B0h, a második csatornánál B1h, és így tovább, a tizenhatodik csatornánál BFh. A B001h értékből a 01h jelenti, hogy a B0h értékéhez hozzá kell adni a MIDI csatorna számát. A csatorna száma 0..15 között lehet. A 0700h értékből a 00h rész jelenti azt, hogy a 07h értéket változatlan formában kell elküldeni. Ez a Control Change üzenettípusban a Main Volume üzenetnek felel meg. A 0002h értékből a 02h jelenti, hogy az A/D átalakítás során kapott értéket kell 7 bitesre átalakítva elküldeni. A 00h tulajdonképpen bármi lehet, hiszen ide majd az A/D értéke lesz behelyettesítve.

A **Text** az LCD kijelzőre kiírandó szöveg. Az F0h azt jelenti, hogy a potenciométer számát jelenítsük meg. Ennek értéke 1..8 között lehet. Ebből láthatjuk, hogy melyik A/D átalakítóra kötött potenciométer váltotta ki a MIDI üzenet generálását. A ".Volume " szöveget változtatás nélkül kiírjuk. Az F2h karakter helyére az A/D átalakításkor kapott szám 7 bites, előjel nélkülivé átalakított értékét fogjuk behelyettesíteni. Ez a potenciométer bal szélső állásában 0, jobb szélső állásában 127 lesz. A "_c" karaktereket is változatlanul kiírjuk. (A csatorna rövidítésére „ch” kívánczozna, de ne férne bele a 16 karakter hosszúságú sorba.) Az F1h helyére a MIDI csatorna számát helyettesítjük be. A programban a csatornaszámot 0..15-ig tároljuk, de a kijelzőn 01..16-ig jelenítjük meg.

A **"Pan"** üzenet:

PrgPan	Sens : 8
	Size : 3
	Msg: B001h, 0A00h, 0002h
	Text: F0h, ".Pan ", F3h, " c", F1h, 0

21. ábra – A Pan üzenet táblázata

Csak a változásokat nézzük meg a "Volume"-hoz képest. A 0A00h értékből a 0Ah a Control Change üzenet típus „Panorama” üzenet kódja. A Text részben az F3h az LCD kijelzőn előjelesen fogja a potenciométer állását megjeleníteni. Bal szélső állás esetén –64, jobb szélső állásnál 63, míg középállásnál 0.

10 bites A/D átalakító értékei																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
0	1	2	3	4	5	6	7																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							</

22. ábra - 7 bites, előjeles hozzárendelés

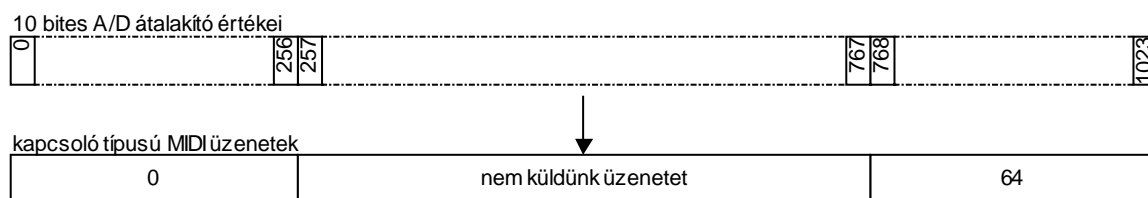
Az elküldendő MIDI üzenetben nem lesz negatív érték, mert csak a kijelzőn megjelenő szám lehet negatív. Az üzenetben a bal szélső állás 0, a jobb szélső 127, míg a középállás 64 értéknek felel meg.

A **"Damper Pedal"** üzenet:

PrgDamper	Sens : 512
	Size : 3
	Msg: B001h, 4000h, 0007h
	Text: F0h, ".Damper ", F6h, " c", F1h, 0

23. ábra – A Damper pedal üzenet táblázata

Az üzenet kétféle (ki-be) állapotú lehet. A kikapcsolt állapotnak a 0, a bekapcsolt állapotnak a 64 érték felel meg. Az érzékenység (**Sens**) értéke 512.



24. ábra - kétállapotú hozzárendelés

A potenciométer bal szélső negyede felel meg a kikapcsolt, a jobb szélső negyede a bekapcsolt állapotnak. Ahhoz, hogy üzenet keletkezzen, az egyik szélső negyed pozícióból kell a potenciométert a másik szélső negyedbe átforgatni. Ezt úgy tudjuk realizálni, hogy az érzékenység (**Sens**) értékét 512-re választjuk. Ha a potenciométert a bal szélső negyedbe tekerjük (függetlenül attól hogy melyik részébe), az utoljára elküldött üzenet A/D értéke (**ADMsgVal**) mindig 256, a jobb szélső negyedbe forgatás esetén 768 lesz. Ebben az esetben, ha a változás legalább 512, akkor biztos, hogy a másik szélső negyedig átforgattuk a potenciométert. Ily módon tulajdonképpen egy hiszterézises komparátor valósult meg, ahol a komparálási szintek a 256 és a 768, a hiszterézis értéke pedig ezek különbsége, azaz 512.

A **"Pitch Bend"** (hajlítás) üzenet:

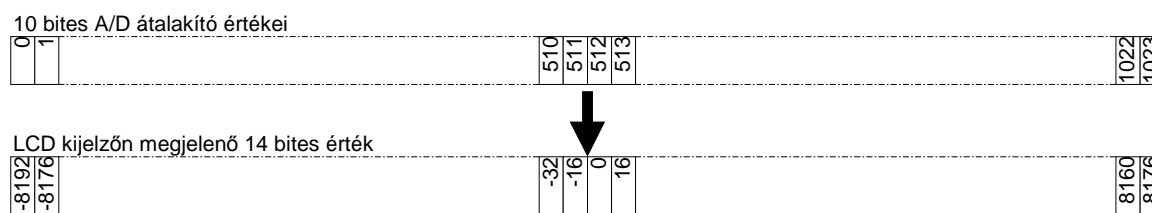
PitchBend	Sens : 2
	Size : 3
	Msg: E001h, 0006h, 0005h
	Text: F0h, ".Pitch", F5h, " c", F1h, 0

25. ábra - A Pitch Bend üzenet táblázata

Az érzékenység (**Sens**) értéke 2. Logikailag ennek egynek kellene lennie, de az A/D átalakítás során kapott érték legkisebb helyiértékű bitje bizonytalan, ezért az értéknek legalább kettővel meg kell változnia, hogy üzenetet küldjünk. Mivel az előállított üzenet 14 bites, az A/D átalakító meg csak 10 bites, az üzenet alsó 4 bitje mindig 0 lesz. Ez az LCD kijelzőn is látszik, hiszen a kiírt érték mindig a 16 egész számú többszöröse. A gyakorlatban ennek nincs jelentősége, hiszen egy forgatópotenciométerre a 10 bites felbontás elegendő. Ugyanis a forgatópotenciométer 270 fokos körbefordíthatóságával és a 10 bites felbontással számolva közelítőleg $\frac{1}{4}$ fok a készülék felbontóképessége. A beállított érzékenységből következik, hogy legalább $\frac{1}{2}$ fokos elfordítás szükséges egy hajlítási üzenet küldéséhez.

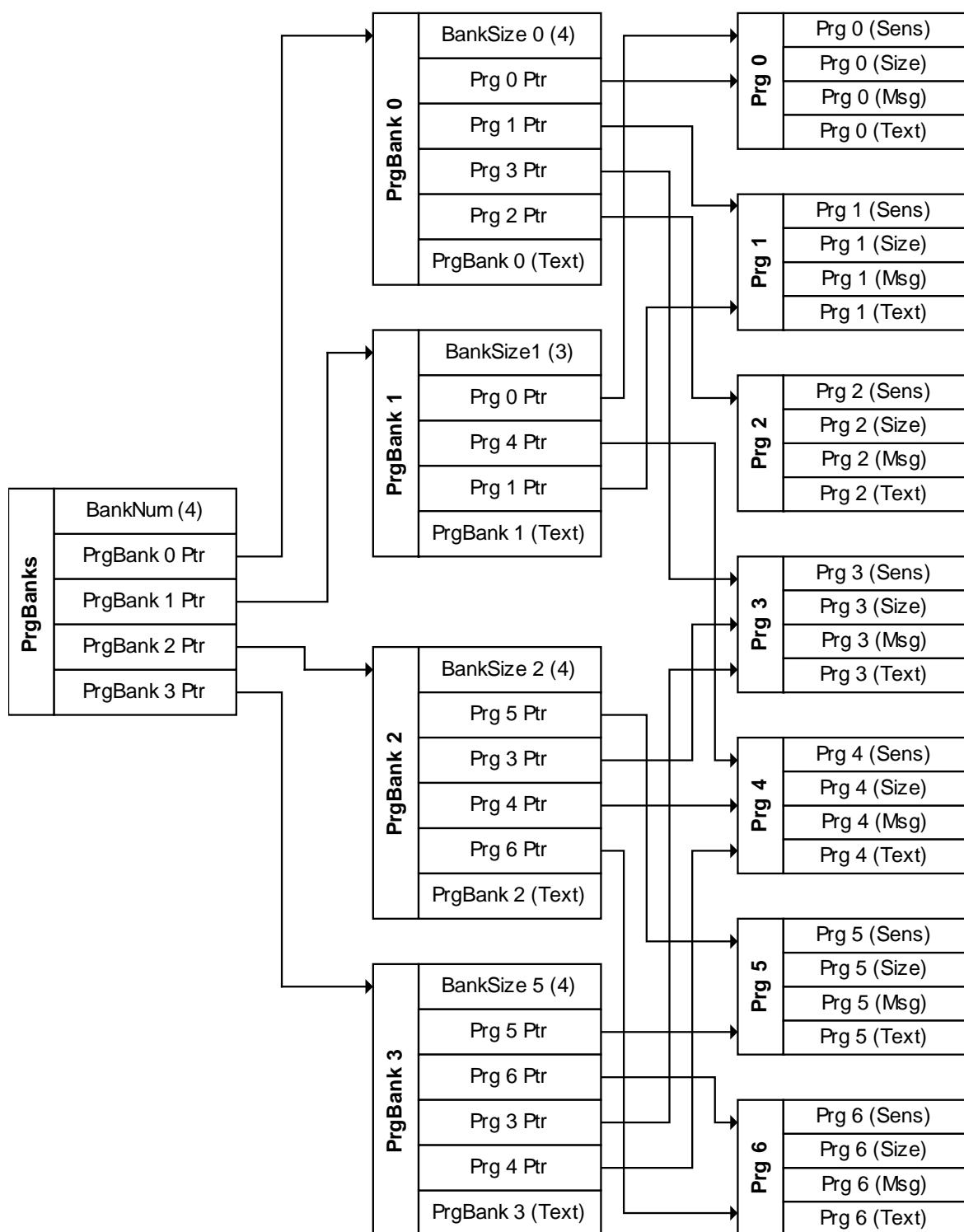
Az üzenet adatbájttjai (**Msg**): Az E001h-ból az E0h a hajlítási üzenet kódja, a 01h pedig a csatornaszám hozzáadása. A 0006h azt jelenti, hogy az A/D átalakító által kapott érték alsó 7 bitjét kell elküldeni. A 10 bites átalakítás miatt ez csak 3 hasznos bit, a többi 0. A 0005h pedig az A/D eredmény elküldendő felső 7 bitje.

A **Text** részben az F5h az A/D érték LCD kijelzőn történő megjelenítése, ami 14 bites és előjeles (-8192..8191).



26. ábra - 14 bites, előjeles hozzárendelés

Példa egy teljes táblázat-hierarchiára:



27. ábra – Példa egy teljes táblázat-hierarchiára

7.5. A program összeállítások táblázata.

Ez táblázat is a „Programs.asm” modulban található, az adat EEPROM-ban helyezkedik el. Egy összeállítás tárolásához 24 bájt szükséges. Ebből következik, hogy a 256 bájt méretű adat EEPROM-ba 10 összeállítás fér bele. Egy összeállítás 24 bájt területére három 8 bájtos tömbből áll, amelyek sorrendben az alábbi adatokat tartalmazzák:

- A 8 potenciométerhez rendelt bankok száma
- A 8 potenciométerhez tartozó program száma
- A 8 potenciométerhez tartozó MIDI csatorna száma

Példa egy olyan összeállítás programkódjára, ahol a 8 potenciométerhez Panoráma szabályozás üzenet van hozzárendelve, a MIDI csatorna pedig 1-től 8-ig terjed:

```
de 0, 0, 0, 0, 0, 0, 0, 0 ; bank = 0 (mert a 0. bankban van a Pan)
de 1, 1, 1, 1, 1, 1, 1, 1 ; program = 1 (mert az 1. üzenet a Pan)
de 0, 1, 2, 3, 4, 5, 5, 7 ; csatorna = 1..8
```

A táblázat eleje a „SaveData” címkével van megjelölve. Az összeállítások folyamatosan követik egymást a memóriában, így pl. a hetedik összeállítás címét úgy kaphatjuk meg, ha a 24-et 7-el megszorozzuk, majd hozzáadjuk a táblázat kezdőcímét.

A táblázat után található egy egybájtos változó (**SavePresetNum**), ami azt tartalmazza, hogy melyik összeállítás (Preset) az aktuális.

7.6. A billentyűzetkezelés megvalósítása

A billentyűzet kezelését megvalósító programok a „Keyb.asm” modulban kaptak helyet.

A modul interfész része két fő programrutint tartalmaz:

- **KeybInit:** Inicializálja a modul változóit, valamint betölti az utoljára használt összeállítást az adat EEPROM memóriából.
- **KeybPolling:** Lekérdezi a nyomógombokat, és ha szükséges, akkor a készülék beállításait módosítja és megjeleníti az LCD kijelzőn. Módosítás esetén gondoskodik a mentés (**Save**) jelzőbit beállításáról is. A potenciométer-szelektáló gomboknál a lenyomott állapotot, a programmódosító gomboknál csak a gomb felengedését figyeljük. A lekérdezésekre másodpercenként 34-szer kerül sor. A szubrutin meghívásáról a „Main.asm” modul gondoskodik, az időzítő (Timer0) által kiváltott megszakítás segítségével. Ez az eljárás a fejezet későbbi részében részletesen kifejtésre kerül.

Két belső felhasználású szubrutin található még a modulban:

- **PresetLoad:** Az adat EEPROM memóriából betölt egy program összeállítást. A betöltendő összeállítás számát **PresetNum** változóban adjuk meg. Az eljárás az adat EEPROM memóriából bemásolja a bankszámokat, a programszámokat valamint a csatornaszámokat rendre az **ADBanks**, az **ADPrgs** és az **ADChns** tömbökbe. Az így beállított értékek szerint az **MsgPtrLoad** szubrutin feltölti az **ADMsgPtr** mutatótömböt az üzenetek címeivel.
- **MsgPtrLoad:** Az **ADBanks** és az **ADPrgs** tömbök tartalma szerint beállítja az **ADMsgPtr** mutatótömbben az **ADSelect** változó által kijelölt potenciométerhez tartozó üzenet címét.

Az eljárások által használt konstans:

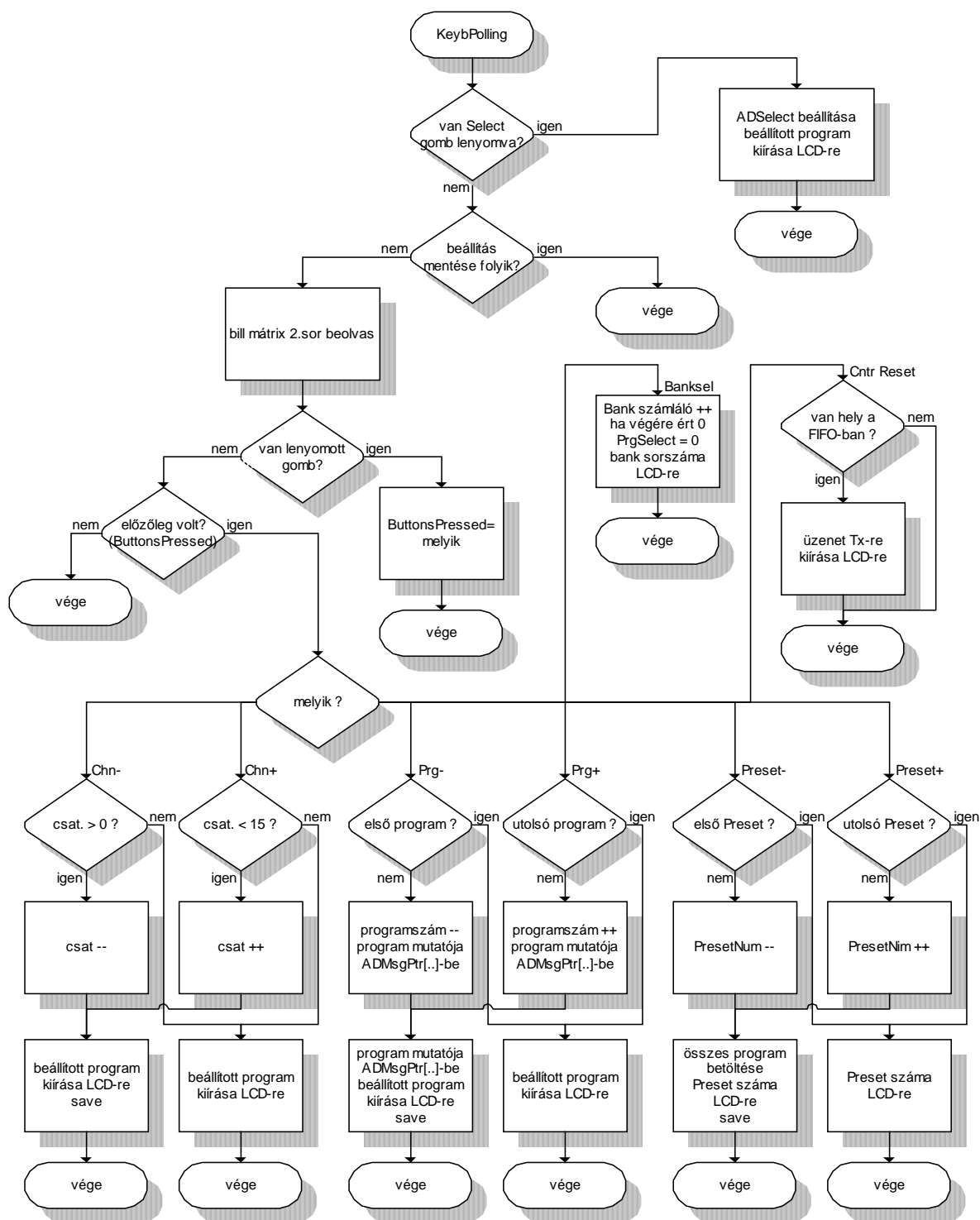
- **PresetSize:** ennyi számú összeállítást tudunk kiválasztani, ill. eltárolni (jelenleg 10)

Az eljárások által használt változók:

- **ADSelect:** Az aktuálisan kiválasztott potenciométer száma (0..7). A kiválasztás a potenciométer felett található gombbal történik. A programváltások a kiválasztott potenciométerrel végezhetők.

- **ADBanks:** Ez a 8 bájtos tömb a 8 potenciométerhez tartozó üzenet bankját határozza meg. Értéke mondja meg, hogy az üzenet bankjának kezdőcímét hányadik cím tartalmazza a **PrgBanks** táblázatban.
- **ADPrGs:** Ez a 8 bájtos tömb a 8 potenciométerhez tartozó üzenet számát határozza meg. Ez a szám mondja meg, hogy a bank hányadik mutatója tartalmazza az üzenet kezdőcímét.
- **PresetNum:** Egy bájtos szám, az aktuális Preset értékét tárolja.
- **BankSelet:** A kiválasztott potenciométerhez tartozó üzenet hányadik bankban van.
- **PrgSelect:** A kiválasztott potenciométerhez tartozó üzenet hányadik a bankban.
- **Buttons:** Egy bájtos szám. Bitjei az aktuális billentyűmátrix sorában lenyomott gombokat jelzik. A lenyomott gomb logikai 1-nek felel meg.
- **ButtonsPressed:** Egy bájtos szám. Ha lenyomtunk 1 gombot, a gomb sorszámát (1..8) tesszük ide. A gomb felengedésének érzékelésére szolgál. Ha nincs lenyomva egyik billentyű sem, akkor 0 értéket kap.
- **Count:** Számláló.
- **Save:** Ezt a bitet minden változtatás esetén 1-re állítjuk. Ezzel jelezzük a főprogramhurokban futó **PresetSave** eljárásnak, hogy mentés szükséges. A változó a „Main.asm” modulban van definiálva. Azért kellett a define direktívával ebben a modulban is definiálni, mert a direktíva nem exportálható.

A KeybPolling eljárás részletes kifejtése:



28. ábra - A KeybPolling eljárás folyamatábrája

A Potenciométer kiválasztó gombok lekérdezése:

- A Select-potenciométer kiválasztó nyomógombok mátrixvonalát kimeneti irányba és alacsony logikai szintre állítjuk. Visszaolvassuk a mátrix oszlop 8 bites értékét, mindegyik bitjét negáljuk. A mátrixvonalat visszaállítjuk bemeneti irányba. Ha a visszaolvasott érték mindegyik bitje 0, akkor továbblépünk a többi 8 gomb lekérdezésére. Ha nem, megnézzük, hogy hányadik az első logikai 1 értékű bit. Az ADSelect értékét ez alapján módosítjuk, majd kiírjuk az LCD kijelzőre a kiválasztott potenciométerhez tartozó üzenet nevét. Ezután kilépünk az eljárásból.

Többi 8 gomb lekérdezése:

- Megnézzük a **Save** bit értékével, hogy egy esetleges beállítás mentése befejeződött-e. Ha még nem, akkor kilépünk az eljárásból. Ezt azért tesszük, mert inkonzisztens lehet az elmentett állapot, ha mentés közben megváltoztatjuk a beállításokat.
- A második nyomógombcsoport mátrixvonalának kimenetre és alacsony szintre állítása után visszaolvassuk a mátrix oszlop 8 bites értékét.
- Ha nincs lenyomott gomb, megvizsgáljuk, hogy előzőleg volt-e lenyomott gomb (**ButtonsPressed** értékével). Ha volt, akkor azt most engedjük fel, ezért a lenyomott gombhoz tartozó parancsot végre kell hajtani.
- Ha van lenyomott gomb, akkor annak kódját (1..8) be kell rakni a **ButtonsPressed** változóba.

A gombokhoz tartozó parancsok:

Chn+ gomb:

- Ha az **ADSelect** által meghatározott potenciométerhez tartozó **ADChns** értéke < 15 , akkor értékét eggyel megnöveljük.

Chn- gomb:

- Ha az **ADSelect** által meghatározott potenciométerhez tartozó **ADChns** értéke > 0 , akkor értékét eggyel csökkentjük.

Prg+ gomb:

- A **PrgBanks** táblázatból kiolvassuk az üzenet bankjának mutatóját: $ptr = \text{BankArray}[\text{BankSelect}]$ -ik mutatója. Az így kapott mutató fogja megcímezni a kiválasztott bankot.
- A megcímezett tömbből az első elemet kiolvassuk. Ez azt tartalmazza, hogy a kiválasztott bankban hány program található.
- Összehasonlítjuk a **PrgSelect** és a bank üzeneteinek számát, szükség szerint megnöveljük a **PrgSelect** értékét.
- A **PrgSelect** értékét visszaírjuk a programokat tároló tömbbe.
- Kiíratjuk az LCD kijelzőre a kiválasztott üzenet nevét.
- A **Save** bitet 1-re állítjuk.

Prg- gomb:

- Ha **PrgSelect** értéke nagyobb mint nulla, akkor eggyel csökkentjük értékét.
- A **PrgSelect** értékét visszaírjuk a programokat tároló tömbbe.
- Kiíratjuk az LCD kijelzőre a kiválasztott üzenet nevét.
- A **Save** bitet 1-re állítjuk.

Bank Select gomb:

- Lenullázzuk **PrgSelect** változót.
- Megnöveljük eggyel a **BankSelect**-et.
- Ha a megnövelt érték nagyobb, mint a bankok száma, akkor lenullázzuk. Ekkor a legelső bankban leszünk. Mivel a Bank Select gombbal csak egy irányba léphetünk, az utolsó bank után az első következik.
- Az LCD kijelzőre kiíratjuk a kiválasztott bank nevét.

Preset+ gomb:

- Ha nem az utolsó összeállítás (Preset) van kiválasztva, akkor értékét eggyel megnöveljük.
- Az összeállítás betöltése az adat EEPROM-ból.
 - Az **ADBanks** tömb feltöltése.
 - Az **ADPrgs** tömb feltöltése.
 - Az **ADChns** tömb feltöltése.
 - Az **ADBanks** és **ADPrgs** értékeit felhasználva **ADMsgPtr** feltöltése, a **PresetLoad** szubrutin meghívásával.
- Kiíratjuk az LCD kijelzőre az újonnan kiválasztott Preset számát.
- A **Save** bit 1-re állítása, mivel változott a beállítás.

Preset- gomb:

- Ha nem az első összeállítás (Preset) van kiválasztva, akkor értékét eggyel csökkentjük.
- Az összeállítás betöltése az adat EEPROM-ból.
 - Az **ADBanks** tömb feltöltése.
 - Az **ADPrgs** tömb feltöltése.
 - Az **ADChns** tömb feltöltése.
 - Az **ADBanks** és **ADPrgs** értékeit felhasználva **ADMsgPtr** feltöltése, a **PresetLoad** szubrutin meghívásával.
- Kiíratjuk az LCD kijelzőre az újonnan kiválasztott Preset számát.
- A **Save** bit 1-re állítása, mivel változott a beállítás.

System Reset gomb:

- Ha a FIFO-ban nincs legalább 4 bájt szabad hely, akkor kilépünk az eljárásból.
- Az FFh bájtot (System Reset üzenet) elküldjük a MIDI kimenetre.
- Az LCD kijelzőre kiíratjuk a „System Reset” üzenetet.

7.7. A potenciométer (A/D) eljárások

A potenciométerek kezelését megvalósító eljárások az „ADConv.asm” modulban kaptak helyet.

Ebben a modulban két szubrutin található. Mindkét szubrutin elérhető másik modulból is.

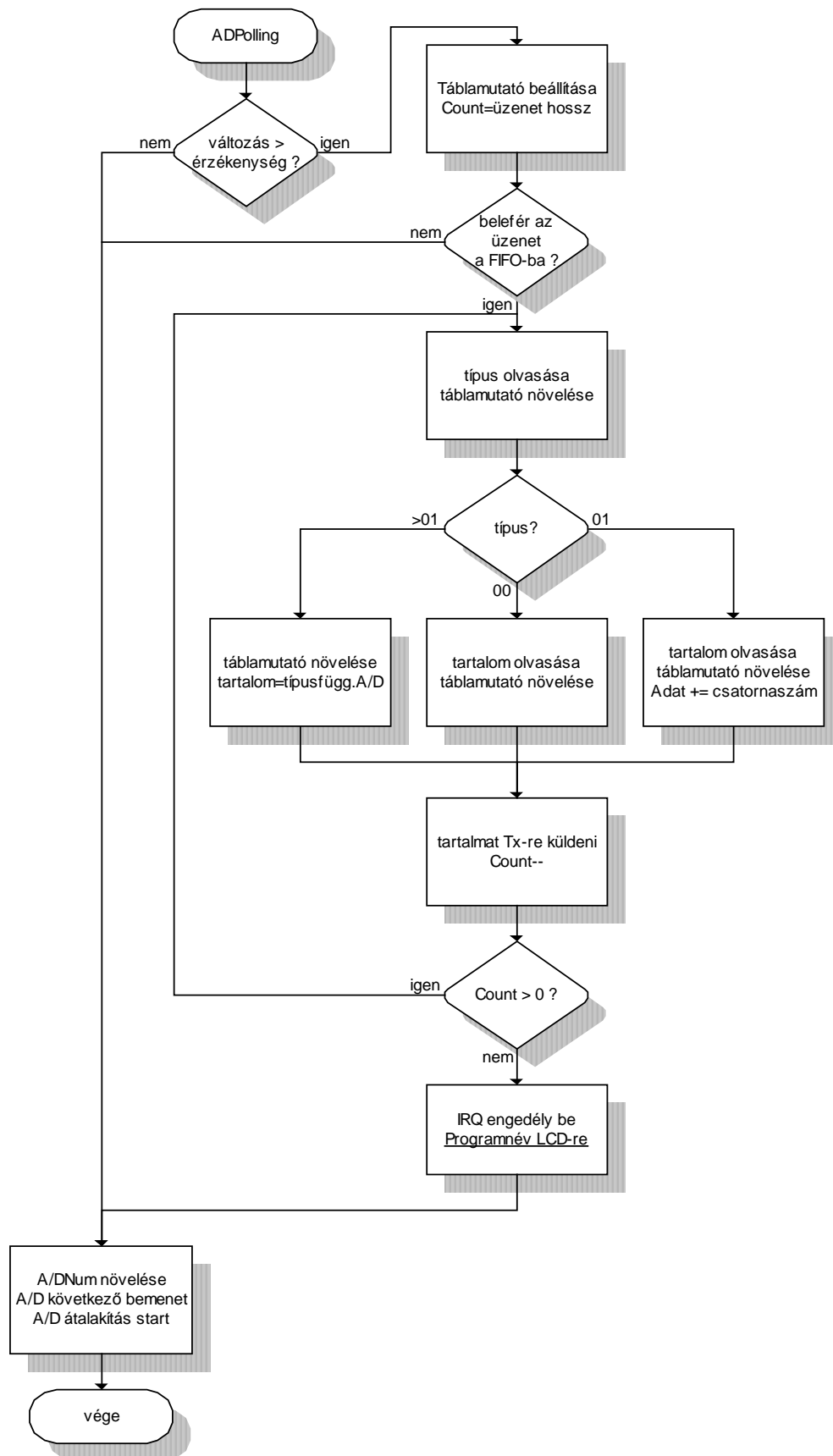
- **ADInit:** Bekapcsolás után inicializálja az analóg digitális átalakító perifériáit. Beállítja az A/D átalakító órajelét. Gondoskodik arról, hogy a kapott érték ADRESL/ADRESH regiszterpárban alulra legyen igazítva. Ez azt jelenti, hogy a 10 bites érték alsó 8 bitje az ADRESL, a felső 2 bitje az ADRESH regiszterbe kerül. (Úgy is be lehetne állítani, hogy a felső 8 bitje az ADRESH, az alsó 2 bitje az ADRESL regiszterbe kerüljön. Ez akkor lenne előnyös, ha csak 8 bites A/D átalakítóként szeretnénk használni, így ugyanis nem kell a biteket tologatni.) Ezután beolvassuk mind a 8 potenciométer aktuális állását, és eltároljuk az **ADVal** valamint az **ADMsgVal** tömbökben. A kettős eltárolás azért szükséges, hogy a készülék bekapcsoláskor ne hozzon létre feleslegesen nyolc db üzenetet.
- **ADPolling:** Lekérdezi a soron következő potenciométer állását, és a változástól függően MIDI üzenetet küld. Üzenet akkor keletkezik, ha a változás nagysága eléri az **ADMsgSens** tömbben tárolt értéket. A program betöltésekor az üzenet táblázatában levő érzékenységi értékét (**Sens**) bemásoljuk az **ADMsgSens** tömb megfelelő helyére. Az üzenet elküldésekor az LCD kijelzőre kiírja az üzenet nevét és a paramétereit. A szubrutin meghívásáról 3,2 milliszekundumonként (305 lekérdezés másodpercenként) a „Main.asm” modul gondoskodik, a Timer0 időzítő által kiváltott megszakítással.

Az eljárások által használt adatok:

- **ADNum:** A soron levő A/D bemenet sorszáma (0..7). Minden beolvasáskor megnöveljük az értékét eggyel. Ha elérte a 8-at, lenullázzuk.
- **ADVal:** 8 elemű, 10 bites értékeket tároló tömb (16 bitet, azaz 2 bájtot tartunk fent egy elemre). A potenciométerek aktuális állásait ebben tároljuk. Minden A/D konverzió után frissítjük az itt tárolt értéket.
- **ADMsgVal:** 8 elemű, 10 bites értékeket tároló tömb (16 bitet, azaz 2 bájtot tartunk fent egy elemre). Ebben tároljuk, hogy a potenciométer által kiváltott utolsó üzenethez mekkora A/D érték tartozott. A továbbiakban a változást ehhez az értékhez fogjuk viszonyítani. Értéke csak a legközelebbi MIDI üzenet elküldésekor módosul.

- **ADMsgSens:** 8 elemű, 10 bites értékeket tároló tömb. Ez határozza meg, hogy mekkora A/D eredmény változás esetén szükséges MIDI üzenetet küldeni.
- **ADMsgPtr:** 8 elemű, 16 bites mutatókat tároló tömb. Az egyes potenciométerekhez rendelt MIDI üzenetet tartalmazó tömb kezdetére mutat, ami a programmemóriában található.
- **Count:** 1 bájtos számláló.
- **Temp:** 1 bájtos átmeneti tároló.

Az **ADPolling** eljárás működése:



29. ábra – Az ADPolling eljárás folyamatábrája

Az A/D átalakítás oly módon történik, hogy a szubrutin végén indítjuk el az A/D konverziót, melynek eredményét a következő lekérdezési ciklusban (a szubrutin elején) olvassuk be. Ez azért előnyös, mert a konverzió a lekérdezési ciklusok közötti időben zajlik, így nem a szubrutinban kell megvárni a konverzió befejeztét.

Az eljárás elején megvizsgáljuk, hogy befejeződött-e az A/D átalakítás. Ha még nem, akkor kilépünk. Ha befejeződött, beolvassuk az A/D átalakítás eredményét, majd összehasonlítjuk az **ADMsgVal** tömb megfelelő értékével. Ha a különbség abszolút értéke kisebb, mint ami az **ADMsgSens** (érzékenység) tömbben szerepel, akkor nem kell üzenetet küldeni. Ebben az esetben megnöveljük az **ADNum** értékét eggyel, az A/D bemenetere a következő potenciometert kapcsoljuk. Ezután elindítjuk az A/D átalakítási folyamatot. Az eredményt a következő lekérdezési ciklusban olvassuk be.

Ha a különbség nagyobb, akkor MIDI üzenetet kell küldeni, ami a következő módon történik:

1. **ADMsgPtr** tömb megfelelő elemét felhasználva beállítjuk a táblamutatót az üzenet kezdőcímére.
2. Kiolvassuk az üzenet hosszát, és értékét betesszük a számlálóba (**Count**).
3. Megvizsgáljuk, hogy az üzenet befér-e a FIFO tárolóba.
4. Ha nem fér be, akkor az A/D átalakítóra a következő potenciometert kapcsoljuk, elindítjuk az A/D konverziót, majd kilépünk.
5. Ha befér, letiltjuk a megszakítást. Ez azért szükséges, hogy a generált MIDI üzenet belsejébe ne keveredhessen bele egy esetlegesen a soros vonalon érkező üzenet. Az lényeges, hogy a megszakítás tiltása a lehető legrövidebb ideig tartson, mert így elkerülhető MIDI bemenetre érkező üzenetekben az adatvesztés.
6. Beolvasunk a táblázatból egy bájtot, ez lesz az adat típusa.
7. Beolvasunk még egy bájtot, ez lesz az adat tartalma. A tartalmat a típustól függő módosítás után elküldjük a soros kimenetre (**UsartTx**).
8. A számláló értékét eggyel csökkentjük. Ha az még nem nulla, akkor visszatérünk 6. pontra.
9. Engedélyezzük a megszakítást. A soros vétel okozta esetleges megszakítás már nem okozhat keveredést, mert az üzenet küldése befejeződött.
10. **PrgNameWrite** eljárást felhasználva kiíratjuk az LCD kijelzőre az üzenet nevét és paramétereit.
11. A/D átalakítóra a következő potenciometert kapcsoljuk, elindítjuk az A/D konverziót, majd kilépünk.

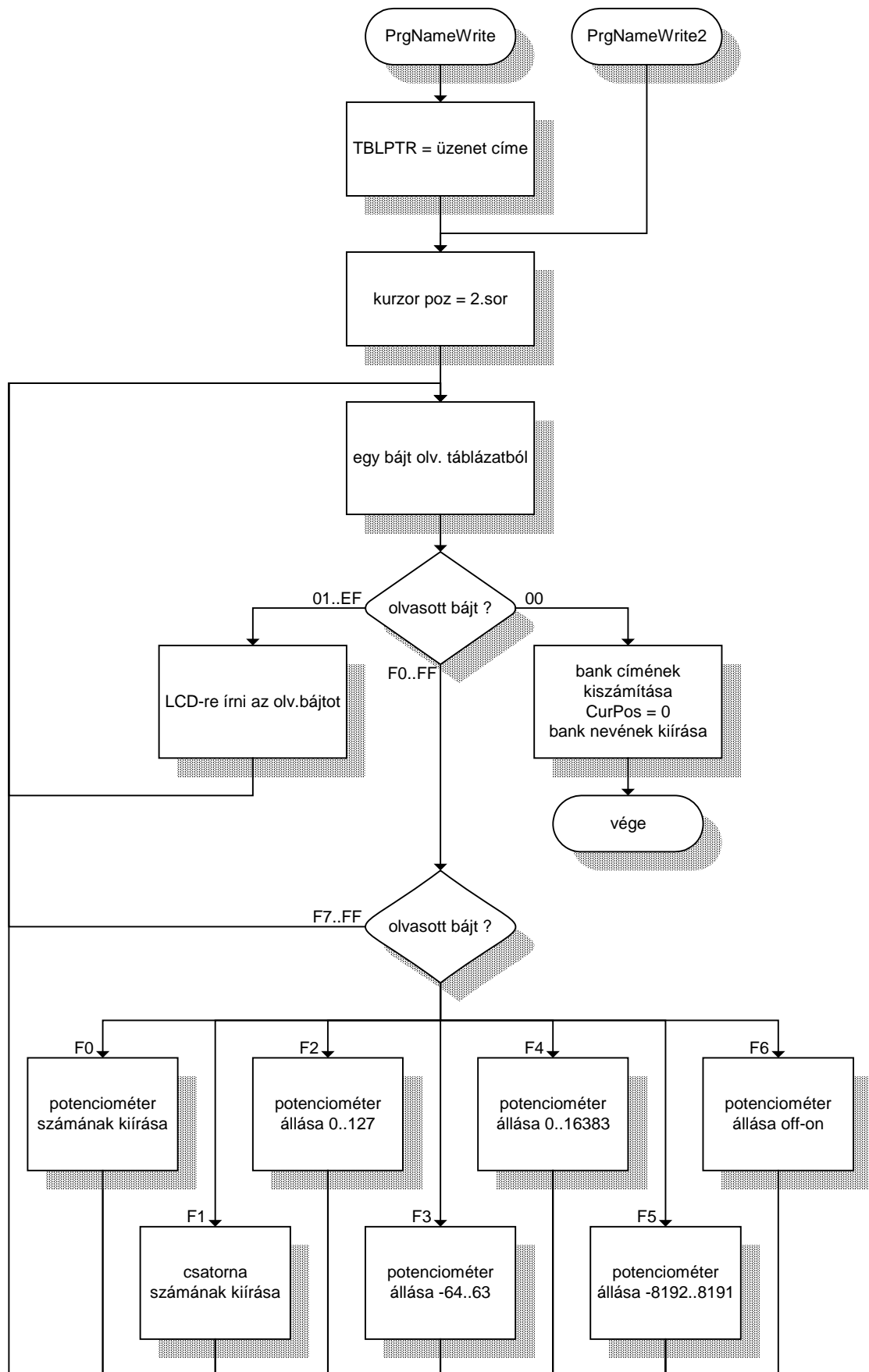
7.8. Az üzenet nevének kiírása

Az üzenet nevének kiírását elvégző eljárás a „PrgWrite.asm” modulban található.

A modulban egy eljárás van, az viszont két belépési ponttal is rendelkezik.

A **PrgNameWrite** és a **PrgNameWrite2** annyiban különbözik egymástól, hogy az utóbbi kihagyja azt a programrészt, ahol a kiírandó szöveg címének kiszámítása történik. Ezt azért lehet kihagyni, mert a MIDI üzenet adatainak elküldése után a táblamutató egyébként is az üzenet nevének kezdetén áll.

Az eljárás mindkét belépési pontja másik modulból elérhető. A **PrgNameWrite** belépési pontot a „Keyb.asm” modulból, a **PrgNameWrite2**-t pedig az „ADConv.asm” modulból hívjuk meg. Azt, hogy melyik potenciométerhez tartozó üzenet nevét jelenítse meg, az **ADNumLcd** változóban kell megadnunk. A kijelző felső sorában az üzenetet tartalmazó bank neve is megjelenítésre kerül. A bank nevét a **PrgBanks** tömb **ADNumLcd**-edik eleme határozza meg. Az eljárás folyamata a következő ábrán látható: 30. ábra – A PrgNameWrite eljárás folyamatábrája.



30. ábra – A PrgNameWrite eljárás folyamatábrája

7.9. Az Usart adás és vétel megvalósítása

Az Usart adást és vételt megvalósító szubrutinok az „Usart.asm” modulban kaptak helyet.

A modulban 4 szubrutin található. Mindegyik elérhető másik modulból is.

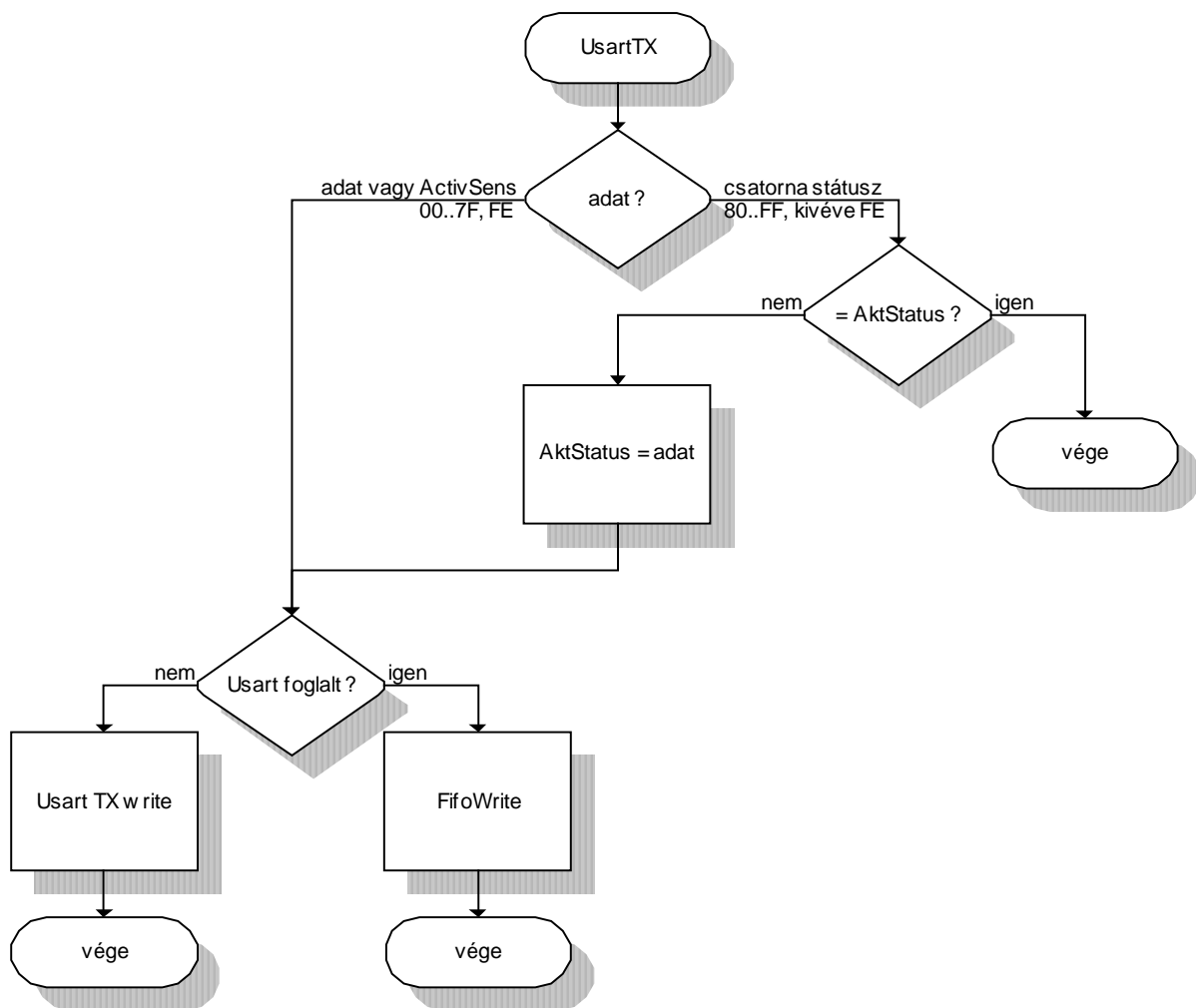
- **UsartInit:** A soros port inicializálását végzi. Hívása a „Main.asm” modulból, inicializáláskor történik. Elvégzi az adáshoz és a vételhez kapcsolódó regiszterek beállítását. Beállítja az aszinkron üzemmódot, a 31250 bit/szekundumos bitsebességet, valamint az egyéb jellemzőket. Engedélyezi a soros port megszakításokat. Az adáshoz alacsony, a vételhez magas prioritási szintet rendel. Gondoskodik a FIFO regiszter inicializálásáról, a **FifoInit** eljárás meghívásával.
- **UsartTx:** Feladata az adatok eljuttatása a soros kimenetre. Minden MIDI adás rajta keresztül valósul meg. Ha a soros port küldő regisztere foglalt (**TXREG**), akkor a **FifoWrite** eljárással a FIFO tárolóba helyezi az elküldendő adatokat. Ha a küldő regiszter ismét szabaddá válik, a soros port megszakítást okoz. Meghívása több helyről is történik (UsartRxIrq, ADPolling, KeybPolling, stb.).
- **UsartTxIrq:** Feladata a FIFO tárolóban elhelyezett adatok eljuttatása a soros kimenetre. Meghívása a „Main.asm” modulból történik, ha a soros kimeneti port szabaddá válik és ezáltal megszakítást okoz. Ekkor a **FifoRead** eljárással kiolvassunk egy bájtot a FIFO-ból, és beírjuk a soros port küldő regiszterébe (**TXREG**). Ha a FIFO üres, akkor leállítjuk az adást.

- **UsartRxIrq:** A soros vételről gondoskodó eljárás. Meghívása a „Main.asm” modulból történik, a soros bemenet kiváltotta megszakítás következtében. Feladata a soros vonalon beérkezett (Usart RX) üzenetek továbbküldése a MIDI kimenetre (Usart TX). Csak a teljes üzenet megérkezésekor juttatjuk azt a soros kimenetre. Így oldható meg, hogy a továbbküldött üzenet bájtjai közé ne ékelődhessen be egy saját készítésű üzenet. Ezért átmenetileg el kell tárolnunk a beérkező üzenet bájtjait, kivéve ha System Exclusive üzenetet továbbítunk a bemenetről a kimenet felé. Ennek oka, hogy egy ilyen üzenet hossza nincs meghatározva. Ebben az esetben leállítjuk a saját üzenet generálását, és a beérkezett bájtokat azonnal továbbítjuk a kimenetre. Az üzenet végén található EOX jel (F7h) után visszaállunk a normál vételi üzemmódba, és ismét lehetőség van saját üzenet előállítására. A normál üzenetek legfeljebb 3 bájt hosszúak, ezért csak 3 bájtos átmeneti tárat kell létrehoznunk.

Az eljárások által használt adatok:

- **AktStatus:** A státusztartás megvalósításához szükséges. Ha a küldendő bájt megegyezik ezzel, akkor nem szükséges elküldeni a kérdéses bájtot.
- **RxData:** Az éppen fogadott adat.
- **RxPuff:** A fogadott üzenetet ez a 3 bájtos tömb tárolja.
- **RxCount:** Számláló, melyet a beérkező üzenet bájtjai működtetnek. Értéke bájtonként eggyel növekszik (0, 1, 2 lehet).
- **RxSize:** Azt tartalmazza, hogy a beérkező üzenet hány bájt hosszúságú. Értékét a státuszbajtból határozzuk meg (1, 2, 3 lehet). Ha RxCount+1 ezzel megegyezik, akkor a teljes üzenet megérkezett, tovább küldhető Tx-re.

UsartTx eljárás:

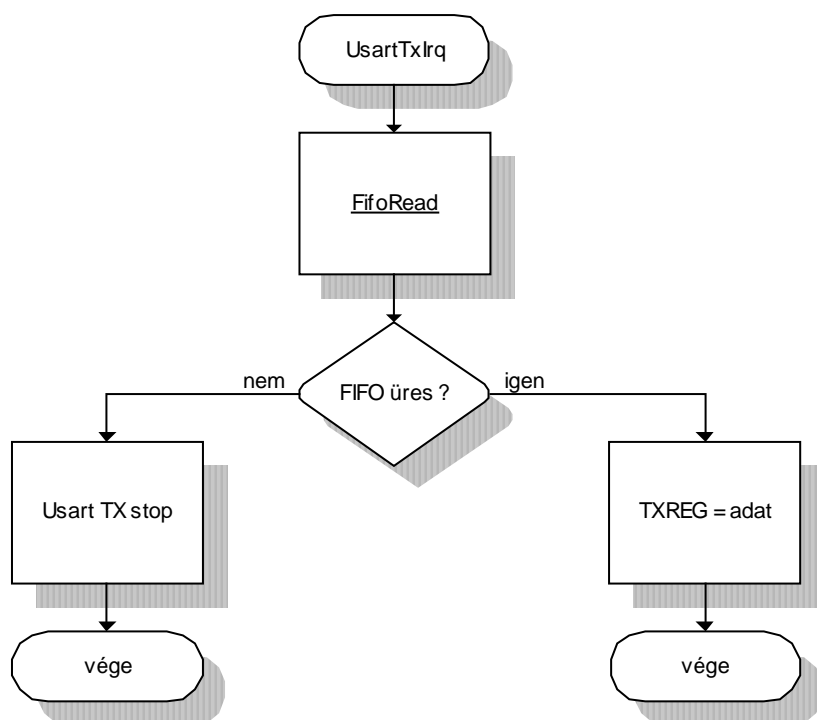


31. ábra – Az UsartTx eljárás folyamatábrája

A küldendő bájtot a W regiszterben kell megadni. Ha ez státuszbájt, akkor megvizsgáljuk, hogy megegyezik-e az **AktStatus** változó értékével. Ha megegyezik, akkor a státusztartás miatt nem szükséges az adatot elküldeni, hiszen a vevő úgyis tárolja az utolsó státusz értékét. Ezzel a módszerrel az átvitt adatok mennyiségét csökkenthetjük. Ha státuszbájt, de nem egyezik meg a most küldendővel, akkor aktualizáljuk **AktStatus** értékét.

Ezután megvizsgáljuk, hogy a soros port adás regisztere foglalt-e. Ha nem, akkor azonnal a küldés regiszterébe (**TXREG**) írhatjuk az adatot. Ha foglalt, akkor nem várjuk meg, amíg felszabadul, hanem **FifoWrite** eljárással a FIFO tárolóba írjuk.

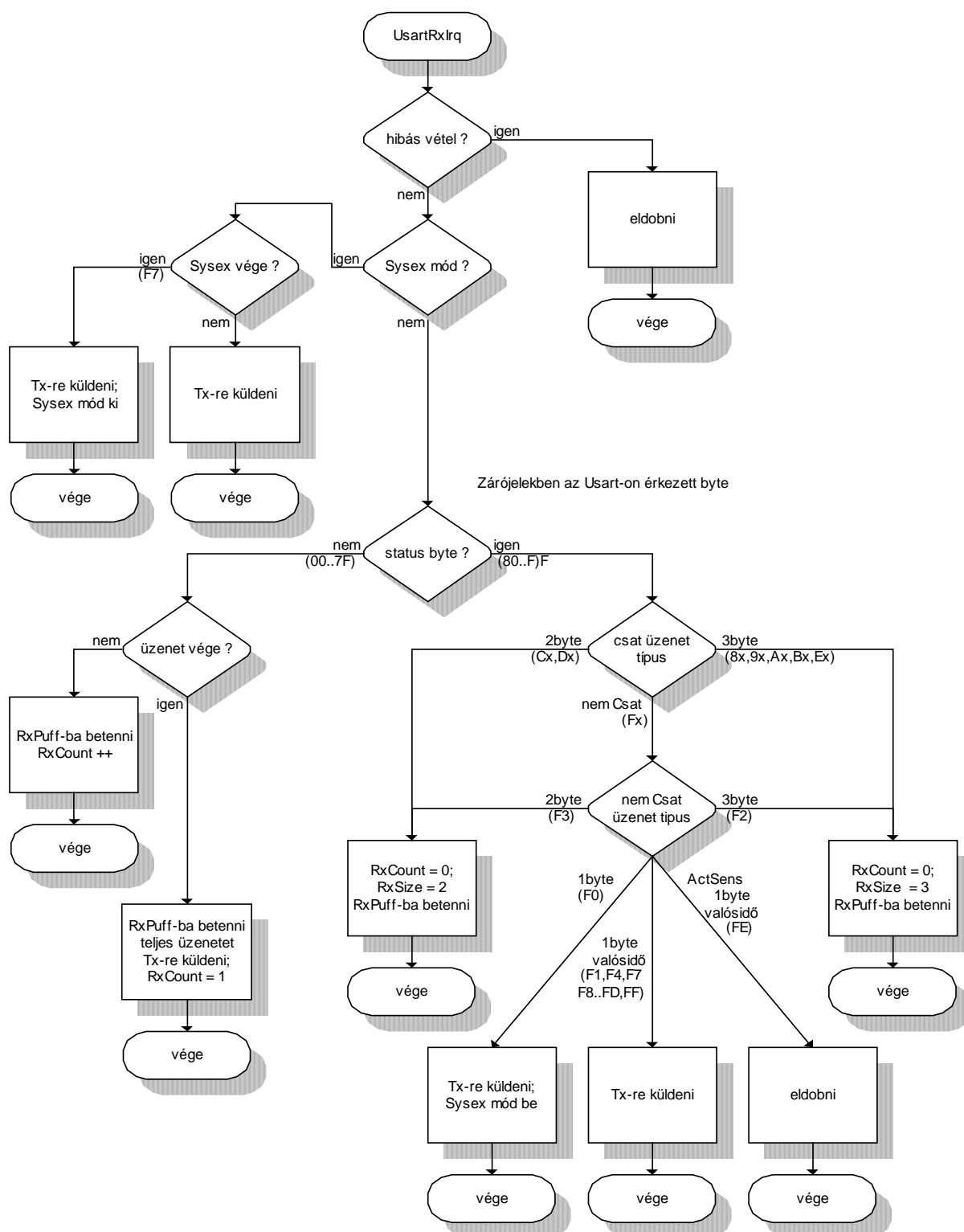
UsartTxIrq folyamatábrája:



32. ábra – Az UsartTxIrq eljárás folyamatábrája

Kiolvasunk egy bájtot a FIFO regiszterből. Ha van benne adat, akkor **TXREG** regiszterbe beírva elküldjük a soros vonalon. Ha üres, akkor leállítjuk a soros adást.

Az alábbi ábrán az **UsartRxIrq** eljárás folyamatábrája látható:



33. ábra – Az UsartRxIrq eljárás folyamatábrája

Az **UsartRxIrq** eljárás működésének részletes leírása a következő:

- Ha a vétel során hiba történt, azt az **OERR** (overrun error – túlfutás hiba) és/vagy a **FERR** (framing error – keret hiba) jelzőbit jelzi. A hibásan vett adatot eldobjuk, és kilépünk.
- Megnézzük, hogy Sysex (System Exclusive) módban vagyunk-e. Ha abban vagyunk, megvizsgáljuk, hogy az éppen vett bájt a Sysex üzenet végét jelenti-e (**F7h**). Ha igen, akkor kikapcsoljuk a Sysex módot. Végül a vett bájt továbbítjuk a kimenetre.
- Ha nem Sysex módban vagyunk, megnézzük, hogy a vett bájt státuszbajt-e. Ha nem az, eltároljuk a 3 bájtos tárolóban (**RxPuff**), leellenőrizzük hogy az üzenet teljes hosszában megérkezett-e. Ha igen, akkor a teljes üzenetet továbbítjuk a kimenetre.
- Ha a vett bájt státuszbajt, akkor több lépcsős vizsgálatnak vetjük alá. Először megvizsgáljuk, hogy csatorna üzenettípusba tartozik-e. Ha igen, akkor lehet kettő (**Cx**, **Dx**), vagy három bájt hosszúságú (**8x**, **9x**, **Ax**, **Bx**, **Ex**) üzenet első bájtja. A hosszúságot tartalmazó bájt (**RxSize**) ennek megfelelően 2-re vagy 3-ra állítjuk, a számlálót (**RxCount**) lenullázzuk, és az érkezett bájtot a hárombájtos puffer (**RxPuff**) elejére beírjuk. Nem csatornaüzenet esetén (**F0h..FFh**) is lehet 2 vagy 3 bájtos üzenet (**F3h** ill. **F2h**). Ez esetben ugyanaz a dolgunk, mint a csatornaüzenetnél. Ha **F0h** (System Exclusive üzenet kezdete) érkezett, akkor be kell kapcsolni a Sysex üzemmódot, és a vett bájtot továbbküldeni. Ha az **FEh** kódú ActSens üzenet érkezett, azt eldobjuk, mert a készülék ezt maga állítja elő. A többi esetben (egy bájtos üzenetek) a beérkezett bájtot egyszerűen továbbküldjük a kimenetre.

7.10. A FIFO tároló megvalósítása

A FIFO tárolót megvalósító szubrutinok a „Fifo.asm” modulban kaptak helyet.

A FIFO (First In First Out – Előbb érkezett előbb megy ki) olyan tároló, amiből a bevétel sorrendjében vesszük ki az adatokat. Ez tulajdonképpen felfogható egy csőnek. A cső egyik vége lesz a bejárat, a másik vége pedig a kijárat. A bejáratnál tehetjük be az elemeket, a kijáratnál pedig kivehetjük őket. A kivételkor mindig olyan sorrendben kapjuk vissza az elemeket, ahogy betettük őket. A cső hossza mindig a benne tartózkodó elemek számától függ. Ha üres, akkor 0 hosszúsággal rendelkezik. Maximális hossza jelen esetben 128. Amikor beteszünk egy elemet, a hossza is megnő, ha kiveszünk belőle, akkor csökken. Betenni csak akkor tudunk elemet, ha még nincs tele, kivenni pedig csak akkor, ha nem üres. A sikertelen betételt vagy kivételt C jelzőbit 1-re állításával jelezzük.

Interfészek:

- **FifoInit:** Alapba állítja az összes változót.
- **FifoWrite:** A W regiszterben található adatot beteszi a csőbe. Ha már nem fér bele C = 1 jelzőbittel jelzi.
- **FifoRead:** A W regiszterbe teszi a cső kijáratánál található adatot. Ha üres a tároló, C = 1 jelzőbittel jelzi.
- **FifoLength:** Ez a változó mondja meg, hogy hány elemünk van a tárolóban, azaz a cső aktuális hosszát jelenti.
- **FifoFree:** Ez a változó megadja, hogy hány elem fér még a csőbe.
- **FifoSize:** Ez egy konstans, a cső maximális hosszát adja meg. Jelenleg 128.

A **FifoFree** tulajdonképpen ki is számítható ($\text{FifoFree} = \text{FifoSize} - \text{FifoLength}$), de gyorsabb és egyszerűbb külön tárolni, mint mindig számolgatni.

Belső változók:

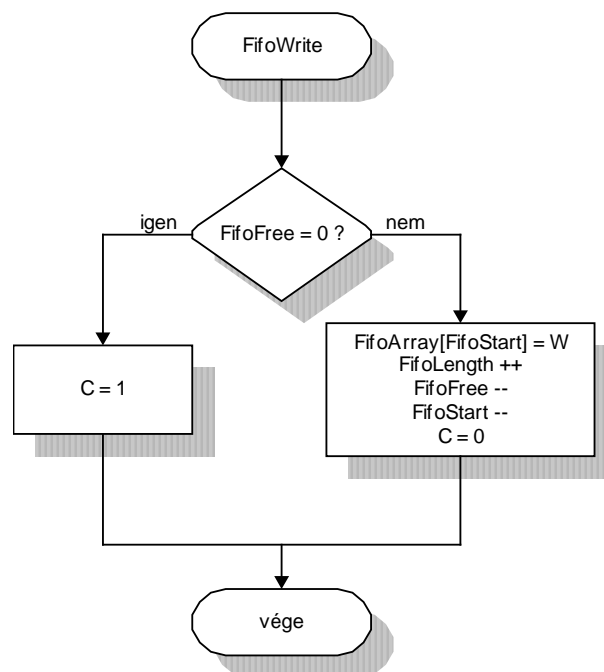
- **FifoArray:** Ebben a tömbben tároljuk a csőben levő elemeket.
- **FifoStart:** Ez mondja meg, hogy a tömbben hol található a cső eleje.

Kijelölünk egy tömböt a memóriában (**FifoArray**). Ha beteszünk egy elemet, a csőben már bent levő elemeket eggyel el kell tolni. Erre két lehetőség van:

- az összes elemet eggyel beljebb toljuk (hátránya, hogy mivel minden elemet mozgatni kell, hosszadalmas).
- adat beírásakor a cső elejét mozgatjuk eggyel hátrább. Ha a cső eleje a tömb elejére ér, lehet újra a tömb végén kezdeni. Ezt körkörös csőnek hívjuk.

A második megoldás esetén van szükség a **FifoStart** változóra. Ez tartalmazza, hogy éppen hol van a cső eleje. A cső maximális hosszát azért érdemes kettő valamelyik hatványára választani, mivel így a tömb elejét és végét egyszerű AND művelettel össze tudjuk kötni.

Az alábbi ábrán a FifoWrite eljárás folyamatábrája látható:

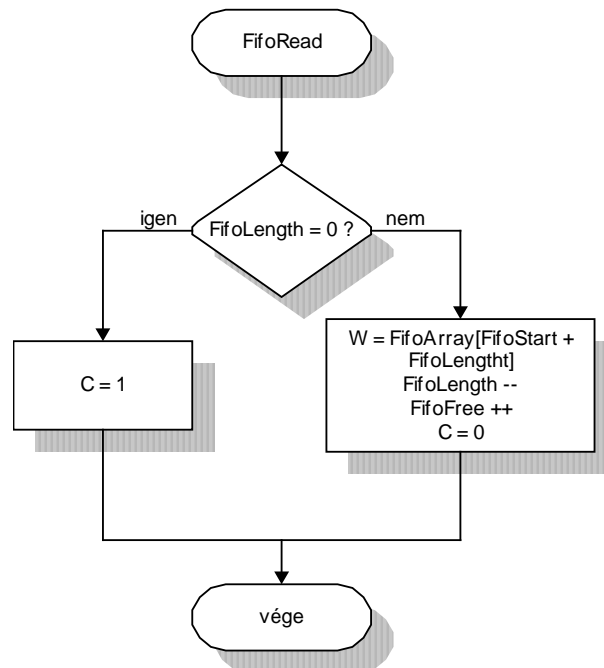


34. ábra – A FifoWrite eljárás folyamatábrája

Működés:

1. Ha tele van a cső, C=1 jelzőbittel jelezzük és kilépünk.
2. Beírjuk a cső elejére az elemet (a cső elejét a FifoStart tartalmazza).
3. Megnöveljük a cső hosszát (FifoLength).
4. Csökkentjük a szabad helyet (FifoFree).
5. Visszább toljuk a cső elejét (FifoStart). Azt, hogy körbejárhassunk, a következő művelettel oldhatjuk meg: $\text{FifoStart} = \text{FifoStart} \text{ AND } (\text{FifoSize} - 1)$.
6. C = 0 jelzőbittel jelezzük a sikeres műveletet.

Az alábbi ábrán a FifoRead eljárás folyamatábrája látható:



35. ábra – A FifoRead eljárás folyamatábrája

Működés:

1. Ha üres a cső, C=1 jelzőbittel jelezzük és kilépünk.
2. Kivesszük a cső végéről az elemet. A cső végének helyét a következő művelettel kapjuk: $(\text{FifoStart} + \text{FifoLength}) \text{ AND } (\text{FifoSize} - 1)$.
3. Csökkentjük a cső hosszát (FifoLength).
4. Növeljük a szabad helyet (FifoFree).
5. C = 0 jelzőbittel jelezzük a sikeres műveletet.

7.11. Az LCD kijelző

Az LCD kijelző eljárásai az „Lcd.asm” modulban kaptak helyet.

A modul két olyan eljárást tartalmaz, ami más modulból is elérhető:

- **LcdInit:** Feladata az LCD kijelző inicializálása. A kijelzőt 4 bites módban működtetjük. Hívása a „Main.asm” modulból történik a bekapcsolás utáni inicializáláskor. Az LCD kijelző inicializálása során több időzített várakozás is szükséges. Erre a célra a „Main.asm” modulban található **Delayusec** és **Delaymsec** eljárások szolgálnak.
- **StringToLcd:** Feladata az LcdString-ben levő szöveg kiírása az LCD kijelzőre. Mind a 32 karaktert kiírja, függetlenül attól, hogy változott-e azok tartalma. Hívása a „Main.asm” modulból történik, ha **LcdRefresh** jelzőbit = 1.

7.12. A stringkezelő eljárások

A stringkezelő eljárások a „String.asm” modulban kaptak helyet.

A stringkezelő eljárások minden esetben az LcdString nevű, 32 bájtos tömbben állítják elő a megjelenítendő szöveget. A kijelző működtetése az „Lcd.asm” modul feladata, a „String.asm” modul ezzel nem foglalkozik.

Az modul eljárásai, és változói:

- **Curpos:** A kurzor pozíciója. Egybájtos érték, írható és olvasható. Minden kiíró eljárás ettől a pozíciótól kezdi a kiírást. Értéke a felső sornál 0..15, az alsó sornál 16..31 között lehet. A kiíró eljárások a jobb alsó sarok után a bal felső sarokban folytatják az írást.
- **ClrScr:** Szóközökkel tölti fel az LcdString tömböt (képernyő törlés).
- **WriteChar:** A W regiszterben levő karaktert kiírja az aktuális kurzorpozícióba, amit eggyel megnövel.
- **ClearLine:** 16 karakternyi szóközt ír az aktuális kurzor pozíciójától kezdve (egy sor törlése).
- **WriteTableStr:** A programmemória mutató (**TBLPTR**) által megcímzett szöveget beírja az **LcdString**-be. Az szöveg végét #0 karakter jelzi.
- **Num:** Ebbe a 16 bites változóba kell elhelyezni a kiírandó számot a **WriteWord** vagy a **WriteInt** eljárások meghívásakor.
- **WriteByte:** Előjel nélküli, egy bájtos szám kiírása. A kiírandó számot a W regiszterben adjuk meg. Három változata van. A **WriteByte1** egy karakterhelyen, a **WriteByte2** kettő karakterhelyen, a **WriteByte3** pedig három karakterhelyen jeleníti meg a számot. Ha a megjelenítendő szám nem fér el a kijelölt karakterhelyen, a szám elejét csonkolja. Mindhárom eljárás a **WriteByte** makrót alkalmazza, a makro megfelelő paraméterezésével. Az 1 vagy 2 karakterhelyes kiírást akkor célszerű használni, ha tudjuk, hogy a szám értéke nem lesz nagyobb 9-nél vagy 99-nél, és nem akarunk 3 karaktert elfoglalni hozzá. Ilyen érték például a potenciométer száma (1-8), a csatornaszám (01-16) stb.

- **WriteShort:** Kettes komplement kódban tárolt, előjeles, egy bájtos szám kiírása. A változatai megegyeznek a **WriteByte** változatokkal. Mindhárom eljárás a **WriteByte** makrót alkalmazza, a makro megfelelő paraméterezésével.
- **WriteWord:** Előjel nélküli, kétbájtos szám kiírása. A számot a **Num** változóba kell beírni. A változatok **WriteWord3**, **WriteWord4** és **WriteWord5**. A megjelenítés ennek megfelelően 3, 4 vagy 5 karakter szélességben történik. Ha nem fér el a szám a kijelölt helyen, az elejét csonkolja. Mindhárom eljárás a **WriteWord** makrót alkalmazza, a makro megfelelő paraméterezésével.
- **WriteInt:** Kettes komplement módon tárolt, előjeles, kétbájtos szám kiírása. A változatok megegyeznek a **WriteWord** változataival. Mindhárom eljárás a **WriteWord** makrót alkalmazza, a makro megfelelő paraméterezésével.

7.13. A makrók

A makrókat a „macro.inc” fájl tartalmazza. A következő makrók vannak definiálva:

ltblptr: A programszámláló mutató feltöltése 16 bites konstans értékkel.

Paraméter:

1. A mutatóba beltöltendő cím. (Címke is használható).

tblptrs: A programmemória mutató mentése egy 16 bites változóba.

Paraméter:

1. Egy 16 bites változó, ahová a mentés történjen.

tblptrl: A programmemória mutató feltöltése egy 16 bites változóból.

Paraméter:

1. Egy 16 bites változó, aminek tartalma áttöltődik a programmemória mutatóba.

lfsrb: Egy adatmemória mutató beállítása egy bájtokat tartalmazó tömb index-edik elemére.

Paraméterek:

1. Melyik adatmutatót állítsa be (0, 1, 2).
2. A tömb kezdőcíme (konstans).
3. 8 bites változó, amelyik az indexet tartalmazza.

lfsrw: Egy adatmemória mutató beállítása egy szavakat tartalmazó tömb index-edik elemére.

Paraméterek:

1. Melyik adatmutatót állítsa be (0, 1, 2).
2. A tömb kezdőcíme (konstans).
3. 8 bites változó, amelyik az indexet tartalmazza.

fsrs: Egy adatmemória mutató mentése egy 16 bites változóba.

Paraméterek :

1. Melyik adatmutatót állítsa be (0, 1, 2).
2. Egy 16 bites változó, ahová a mentés történjen.

fsrl: Egy adatmemória mutató betöltése egy 16 bites változóból.

1. Melyik adatmutatót állítsa be (0, 1, 2).
2. Egy 16 bites változó, ahol a betöltendő mutató található.

WriteByte: Egy 8 bites szám karakteressé alakítását végzi. A paraméterezéstől függően képes előjel nélküli, és előjeles számok átalakítására is. Lehetőség van a bevezető '0' karakterek 'SPACE' karakterre történő cseréjére. Képes kisebb szélességű kiírásra is. Ebben az esetben, ha a szám nem fér el a megadott helyen, az elejét csonkolja. Az átalakítandó számnak W regiszterben kell lennie.

Paraméterek:

1. Ennyi számjeggyel írjuk ki (1, 2, 3). Ez arra szolgál, hogy meghatározzuk a kiírás szélességét. Ha a szám nem fér bele a megadott szélességű helyre, az elejét csonkolja. Az 1 vagy 2 érték akkor hasznos, ha a számunk csak egy, vagy kétszámjegyű lehet.
2. A szám elején előforduló nullákat mivel helyettesítse. Ha szóköz karaktert adunk paraméternek, akkor a szám elején nem lesznek felesleges nullák.
3. Ha '-' karaktert adunk paraméternek, akkor a számot előjelhelyesen írja ki. Úgy működik, hogy negatív szám esetén a számot pozitívvá alakítja, és '-' jelet ír az elejére.
4. A karakterkiíró eljárás címe.

Egy pozitív, három karakterhely igényű szám átalakításának algoritmus a következő:

1. WTemp változónak '0' értéket adunk.
2. A számból 100-at kivonunk.
3. Ha nem csordul alul, megnöveljük WTemp-et és visszamegyünk a 2. pontra.
4. WTemp karakteres formában tartalmazza a százask számát, amit ki is írunk.
5. A számhoz hozzáadunk 100-at, hogy az utolsó, negatív eredményt adó kivonást visszavonjuk.
6. WTemp változónak '0' értéket adunk.
7. A számból 10-et kivonunk.
8. Ha nem csordul alul, megnöveljük WTemp-et és visszamegyünk a 7. pontra.
9. WTemp karakteres formában tartalmazza a tízesek számát, amit ki is írunk.
10. A számhoz hozzáadunk 10-et, hogy az utolsó, negatív eredményt adó kivonást visszavonjuk.
11. WTemp változónak '0' értéket adunk.
12. Hozzáadjuk a számot, ami most az egyesek számát tartalmazza.
13. WTemp karakteres formában tartalmazza az egyesek számát, ezt is kiírjuk.

WriteWord: 16 bites szám karakteressé alakítását végzi. A paraméterezéstől függően képes előjel nélküli, és előjeles számok átalakítására is. Lehetőség van a bevezető '0' karakterek 'SPACE' karakterre történő cseréjére. Képes kisebb szélességű kiírásra is. Ebben az esetben, ha a szám nem fér el a megadott helyen, az elejét csonkolja. Az átalakítandó számnak a Num változóban kell lennie.

Paraméterek:

1. Ennyi számjeggyel írjuk ki (3, 4, 5). Ez arra szolgál, hogy meghatározzuk a kiírás szélességét. Ha a szám nem fér bele a megadott szélességű helyre, az elejét csonkolja. A 3 vagy 4 érték akkor hasznos, ha tudjuk, hogy a számunk csak három vagy négyszámjegyű lehet.
2. A szám elején előforduló nullákat mivel helyettesítse. Működése megegyezik a WriteByte makróval.
3. Ha '-' karaktert adunk paraméternek, akkor a számot előjelhelyesen írja ki. Működése megegyezik a WriteByte makróval.
4. A karakterkiíró eljárás címe.

Algoritmus alapvetően megegyezik a WriteByte makró algoritmusával. A különbség csak az, hogy 16 bites számokkal dolgozik, és a kivonást 10000-el kezdi.

8. Irodalomjegyzék

Sík Zoltán:

MIDI Alapozás

h.n.; Pixel Graphics Kft.; 1992

Gerényi Gábor:

MIDI Protokoll

h.n.; Pixel Graphics Kft.; 1992

Bajusz Péter, Bors Gábor, Csibra Gergő, Horváth Tamás:

A PC-k hangja

Budapest; COM-WARE Kft.; 1995

Dr. Kónya László:

PIC mikrovezérlők alkalmazástechnikája (Második, bővített kiadás)

Budapest; ChipCAD Elektronikai Disztribúció Kft.; 2003

Dr. Kónya László:

PC-elektronika

Budapest; Műszaki Könyvkiadó.; 1991

Microchip Technology Inc.:

PIC18F2455/2550/4455/4550 Data Sheet (DS39632D)

USA; Microchip Technology Inc.; 2007

<http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf>

Hitachi:

HD44780U Data Sheet

<http://www.electronic-engineering.ch/microchip/datasheets/lcd/hd44780.pdf>

9. Ábrajegyzék

1. ábra - A MIDI keverő beillesztése a rendszerbe	5
2. ábra - Az aszinkron soros adatátvitel jelalakja	9
3. ábra - A MIDI kommunikáció tipikus áramkörü megvalósítása	9
4. ábra - A MIDI kábel bekötése	10
5. ábra - A MIDI Thru alkalmazása	10
6. ábra - A készülék fényképe	21
7. ábra - A készülék kezelőszervei	21
8. ábra - LCD bejelentkező felirat	22
9. ábra - LCD felirat (3.Volume 56 c03)	23
10. ábra - LCD felirat (Creative Bank)	23
11. ábra - LCD felirat (3.FiltCt 76 c01)	24
12. ábra - A készülék kapcsolási rajza	26
13. ábra - A készülék áramkörü lapja	28
14. ábra - Fejlesztés az MPLAB ICD2-vel	33
15. ábra - MIDI monitor program az üzenetek teszteléséhez	34
16. ábra - Futási idődiagram	38
17. ábra - A bankok táblázata	43
18. ábra - Egy programbank táblázat	43
19. ábra - A Volume üzenet táblázata	45
20. ábra - 7 bites hozzárendelés	45
21. ábra - A Pan üzenet táblázata	46
22. ábra - 7 bites, előjeles hozzárendelés	46
23. ábra - A Damper pedal üzenet táblázata	47
24. ábra - kétállapotú hozzárendelés	47
25. ábra - A Pitch Bend üzenet táblázata	48
26. ábra - 14 bites, előjeles hozzárendelés	48
27. ábra - Példa egy teljes táblázat-hierarchiára	49
28. ábra - A KeybPolling eljárás folyamatábrája	53
29. ábra - Az ADPolling eljárás folyamatábrája	59
30. ábra - A PrgNameWrite eljárás folyamatábrája	62
31. ábra - Az UsartTx eljárás folyamatábrája	65
32. ábra - Az UsartTxIrq eljárás folyamatábrája	66
33. ábra - Az UsartRxIrq eljárás folyamatábrája	67
34. ábra - A FifoWrite eljárás folyamatábrája	70
35. ábra - A FifoRead eljárás folyamatábrája	71

10. A CD melléklet tartalma

Midi keverő.pdf

A jelenleg olvasott dokumentum

\MidiMixer

A program forráskódja, valamint a project fájldjai (18 db).

Használatához a teljes könyvtárat a C:\ meghajtóra kell másolni,

majd a project fájlt (MidiMixer.mcp) meg kell nyitni az MPLAB-ban (Project/Open).

\DataSheet

39632D.pdf

A PIC18F4550 mikrovezérlő adatlapja

hd44780.pdf

Az LCD kijelző adatlapja

Main.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
#include "macro.inc"

ActSens      EQU      0xFE      ; ezt a bájtot kell küldeni Activ Sens üzenetként
ActSensTime  EQU      .93      ; ilyen sűrűséggel küldjük (300ms)

#define      LcdRefresh PrgStatus,0 ; 1-re állításával jelezzük hogy a LCD-t frissíteni
kell
#define      SysExMode PrgStatus,1 ; ha 1 akkor Sysex módban vagyunk
#define      Save      PrgStatus,2 ; ha 1 akkor beállítás mentése módban vagyunk
#define      ADKeyb     PrgStatus,3 ; az A/D és a bill. lekérdezésének váltására

GLOBAL      Delayusec ; mikroszekundumos várakozás, idő W-ben
GLOBAL      Delaymsec ; milliszekundumos várakozás, idő W-ben
GLOBAL      PrgStatus ; program állapot bitek

EXTERN      UsartInit, UsartRxIrq, UsartTx, UsartTxIrq
EXTERN      LcdInit, StringToLcd

EXTERN      KeybInit, KeybPolling ; billentyűzet
EXTERN      ADInit, ADPolling ; A/D
EXTERN      PresetNum, SaveData, SavePresetNum
EXTERN      ADBanks, ADPrgs, ADChns ;beállítás mentéshez

; String interface
EXTERN      ClrScr, CurPos, WriteChar, WriteTableStr, ClearLine, LcdString, Num
EXTERN      WriteByte1, WriteByte2, WriteByte3 ; 1 bájtos előjel nélkül
EXTERN      WriteShort1, WriteShort2, WriteShort3 ; 1 bájtos előjeles
EXTERN      WriteWord3, WriteWord4, WriteWord5 ; 2 bájtos előjel nélkül
EXTERN      WriteInt3, WriteInt4, WriteInt5 ; 2 bájtos előjeles

UDATA_ACS
; megszakítnál regisztermentéshez
WregTemp     RES      1 ; W regiszter mentéséhez
StatusTemp   RES      1 ; STATUS regiszter mentéséhez
BsrTemp      RES      1 ; BSR regiszter mentéséhez

; általános változói
Delay         RES      2 ; időzítéshez számláló
PrgStatus     RES      1 ; jelzőbiteket tartalmaz
Count        RES      1 ; Mentésnél használt számláló
ActSensCount  RES      1 ; itt számláljuk hogy mikor kell Actív Sens-t küldeni
TimerPhase    RES      1 ; 0 -> Keybord Irq, 1..8 -> AD Irq

;=====
; Reset vector
RESET_VECTOR  CODE     0x0000
              goto     Main

;=====
; magas prioritású megszakítás vector
HI_INT_VECTOR CODE     0x0008

              call     UsartRxIrq
              retfie   FAST

;=====
; alacsony prioritású megszakítás vector
LOW_INT_VECTOR CODE     0x0018
              bra      LowInt

```

```

;=====
CODE
; alacsony prioritású megszakítás rutin
LowInt      movff    STATUS, StatusTemp    ; STATUS regiszter mentése
            movff    WREG, WregTemp        ; W regiszter mentése
            movff    BSR, BsrTemp          ; BSR regiszter mentése

;-----
            btfss    PIR1, TXIE
            bra      TxIrqEnd              ; tiltva van TX irq
            btfss    PIR1, TXIF            ; TX megszakítás ?
            bra      TxIrqEnd              ; nem
TxIrqEnd    call     UsartTxIrq

;-----
            btfss    INTCON, TMR0IF        ; Timer0 Irq ?
            bra      TimerIrqEnd            ; nem

            movf     TimerPhase, W
            btfss    STATUS, Z              ; skip ha == 0
            call     ADPolling              ; ha TimerPhase != 0 -> ADIrq

            movf     TimerPhase, W
            btfsc    STATUS, Z              ; skip ha <> 0
            call     KeybPolling            ; ha ADKeyb = 1 -> KeybIrq

; ha alulcsordul akkor 8-ról kezdjük újra
            movlw    8
            decf     TimerPhase, f
            btfss    STATUS, C
            movwf    TimerPhase

; aktív Sens üzenet küldése
            decf     ActSensCount, f
            bnz      NoSendActSens
            movlw    ActSens
            call     UsartTx
            movlw    ActSensTime
            movwf    ActSensCount
NoSendActSens

            btg      ADKeyb
            bcf      INTCON, TMR0IF

            nop
TimerIrqEnd

;-----
            movff    BsrTemp, BSR          ; BSR visszaállítása
            movff    WregTemp, WREG        ; W visszaállítása
            movff    StatusTemp, STATUS    ; STATUS visszaállítása
            retfie

;=====
; főprogram kezdete
Main        call     Init

; főprogram hurok eleje
MainLoop    btfsc    LcdRefresh            ; kell kijelzőt frissíteni ?
            call     StringToLcd            ; igen
            btfsc    Save                    ; kell menteni ?
            call     PresetSave              ; igen
            bra      MainLoop                ; vissza a hurok elejére

```

```

;=====
; szükséges perifériák és memóriaterületek beállítása
Init          bcf      INTCON, GIEH          ; magas irq tiltás
              bcf      INTCON, GIEL          ; alacsony irq tiltás

              movlw    0x07
              movwf    ADCON1                ; AN0..AN7: analóg, AN8..AN12 digitális

              clrf     PrgStatus              ; PrgStatus = 0
              clrf     TimerPhase            ; fázis = 0

              call     LcdInit                ; LCD kijelző bekapcs
              call     ADInit                ; A/D csatornák
              call     UsartInit              ; soros port
              call     KeybInit              ; billentyűzet

              bsf      RCON, IPEN             ; kettős szintű megszakítás eng.

;-----
; Timer0 irq beállítás
              movlw    B'11000110'          ; 8bites, 1:128
              movwf    TOCON
              bcf      INTCON2, TMR0IP        ; alacsony prioritás
              bsf      INTCON, TMR0IE         ; megsz. eng.
              bsf      INTCON, GIEH          ; magas irq eng.
              bsf      INTCON, GIEL          ; alacsony irq eng.

;-----
; szoveg kiirasa '   Midi mixer   '
              clrf     CurPos
              ltblptr  StartString
              call     WriteTableStr
              bsf      LcdRefresh
              ; "0123456789ABCDEF"
StartString   da      "   Midi Mixer   "
              da      "   version v1.0   ", 0
              return

;=====
; 1..256 mikroszekundumos várakozás
; idő = W mikroszekundum, távoli call hívással teljesen pontos időzítést ad 40MHz-en
Delayusec     movwf    Delay
              nop
              bra      $ + 10
              nop
              nop
              nop
              nop
              nop
              nop
              nop
              decfsz    Delay
              bnz      Delayusec + 6
              return

```

```

;=====
; 1..256 milliszekundumos várakozás
; idő = W milliszekundum, távoli call hívással teljesen pontos időzítést ad 40MHz-en
Delaymsec      movwf    Delay + 1
                nop
                nop
                nop
                nop
                movlw    .249
                bra      $ + 4
                movlw    .250
                rcall    Delayusec
                movlw    .250
                rcall    Delayusec
                movlw    .250
                rcall    Delayusec
                movlw    .249
                rcall    Delayusec
                nop
                nop
                nop
                decfsz    Delay + 1
                bnz       Delaymsec + .14
                return

;=====
; beállítások adat EEPROM-ba történő elmentése
PresetSave      movlw    SavePresetNum
                movwf    EEADR
                bcf       EECON1, EEPGD
                bcf       EECON1, CFGS
                bsf       EECON1, RD
                movf       PresetNum, W
                cpfseq     EEDATA                      ; csak akkor írjuk át ha különböző
                rcall     EeWrite

                movf       PresetNum, W
                mullw      .24
                movf       PRODL, W
                addlw      SaveData
                movwf      EEADR

;-----
; bankok mentése
PSave1          lfsr      0, ADBanks
                movlw      8
                movwf      Count
                bcf       EECON1, EEPGD
                bcf       EECON1, CFGS
                bsf       EECON1, RD
                movf       POSTINC0, W
                cpfseq     EEDATA                      ; csak akkor írjuk át ha különböző
                rcall     EeWrite
                incf       EEADR, f
                decfsz     Count, f
                bra        PSave1

;-----
; programok mentése
PSave2          lfsr      0, ADPrgs
                movlw      8
                movwf      Count
                bcf       EECON1, EEPGD
                bcf       EECON1, CFGS
                bsf       EECON1, RD
                movf       POSTINC0, W
                cpfseq     EEDATA
                rcall     EeWrite
                incf       EEADR, f
                decfsz     Count, f
                bra        PSave2

```

```
;-----
; csatornák mentése
        lfsr      0, ADChns
        movlw     8
        movwf     Count
PSave3  bcf      EECON1, EEPGD
        bcf      EECON1, CFGS
        bsf      EECON1, RD
        movf      POSTINC0, W
        cpfseq    EEDATA
        rcall     EeWrite
        incf      EEADR, f
        decfsz    Count, f
        bra      PSave3

        bcf      Save
        return

;=====
; egy bájt írása adat EEPROM-ba (adat = W, cím = EEADR)
EeWrite  movwf     EEDATA
        bcf      EECON1, EEPGD
        bcf      EECON1, CFGS
        bsf      EECON1, WREN
        bcf      INTCON, GIE
        movlw     55h
        movwf     EECON2                ; írási szekvencia 55h
        movlw     0AAh ;
        movwf     EECON2                ; írási szekvencia 0AAh
        bsf      EECON1, WR
        btfsc     EECON1, WR
        bra      $-2                    ; megvárjuk míg végzett
        bsf      INTCON, GIE
        bcf      EECON1, WREN
        return

END
```

Programs.asm

LIST P=18F4550, F=INHX32

GLOBAL SaveData, SavePresetNum, PrgBanks

Eedata CODE 0xF00000

```
SaveData      ; Volume az 1..8 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 0 bank
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 0 program
de            0, 1, 2, 3, 4, 5, 6, 7 ; preset 0 chn

; Pan az 1..8 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 1 bank
de            1, 1, 1, 1, 1, 1, 1, 1 ; preset 1 program
de            0, 1, 2, 3, 4, 5, 6, 7 ; preset 1 chn

; Reverb az 1..8 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 2 bank
de            2, 2, 2, 2, 2, 2, 2, 2 ; preset 2 program
de            0, 1, 2, 3, 4, 5, 6, 7 ; preset 2 chn

; Chorus az 1..8 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 3 bank
de            3, 3, 3, 3, 3, 3, 3, 3 ; preset 3 program
de            0, 1, 2, 3, 4, 5, 6, 7 ; preset 3 chn

; Több paraméter a 10 csatornára
de            0, 0, 0, 0, 0, 0, 1, 1 ; preset 4 bank
de            0, 1, 2, 3, 4, 5, 0, 1 ; preset 4 program
de            9, 9, 9, 9, 9, 9, 9, 9 ; preset 4 chn

; Volume a 9..16 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 5 bank
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 5 program
de            8, 9,.10,.11,.12,.13,.14,.15 ; preset 5 chn

; Pan a 9..16 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 6 bank
de            1, 1, 1, 1, 1, 1, 1, 1 ; preset 6 program
de            8, 9,.10,.11,.12,.13,.14,.15 ; preset 6 chn

; Reverb a 9..16 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 7 bank
de            2, 2, 2, 2, 2, 2, 2, 2 ; preset 7 program
de            8, 9,.10,.11,.12,.13,.14,.15 ; preset 7 chn

; Chorus a 9..16 csatornára
de            0, 0, 0, 0, 0, 0, 0, 0 ; preset 8 bank
de            3, 3, 3, 3, 3, 3, 3, 3 ; preset 8 program
de            8, 9,.10,.11,.12,.13,.14,.15 ; preset 8 chn

; Reverb, Chorus, SBfiltr, SBRes 1,2 csatornára
de            0, 0, 1, 1, 0, 0, 1, 1 ; preset 9 bank
de            2, 3, 0, 1, 2, 3, 0, 1 ; preset 9 program
de            0, 0, 0, 0, 1, 1, 1, 1 ; preset 9 chn

SavePresetNum de            0 ; bekapcsoláskor ez lesz kiválasztva
```

CODE

```

PrgBanks      dw      2
               dw      PrgBank0
               dw      PrgBank1

PrgBank0      dw      .10
               dw      PrgVolume
               dw      PrgPan
               dw      PrgReverb
               dw      PrgChorus
               dw      PrgBreath
               dw      PrgExprsion
               dw      PrgModulat
               dw      PrgVolumeAll
               dw      PrgPitchBend
               dw      PrgDamperPedal
               db      " Normale Bank ", 0

PrgBank1      dw      2
               dw      PrgSBFilCutoff
               dw      PrgSBResonance
               db      " Creative Bank ", 0

PrgVolume     dw      8      ; érzékenység
               dw      3      ; küldendő üzenet hossza
               dw      0B001, 0700, 0002
               db      0F0, ".Volume ", 0F2, " c", 0F1, 0
;Minta:      "3.Volume 127 c02"

PrgPan        dw      8
               dw      3
               dw      0B001, 0A00, 0002
               db      0F0, ".Pan ", 0F3, " c", 0F1, 0
;Minta:      "4.Pan -12 c02"

PrgReverb     dw      8
               dw      3
               dw      0B001, 5B00, 0002
               db      0F0, ".Reverb ", 0F2, " c", 0F1, 0
;Minta:      "8.Reverb 57 c12"

PrgChorus     dw      8
               dw      3
               dw      0B001, 5D00, 0002
               db      0F0, ".Chorus ", 0F2, " c", 0F1, 0
;Minta:      "2.Chorus 75 c05"

PrgBreath     dw      8
               dw      3
               dw      0B001, 0200, 0002
               db      0F0, ".Breath ", 0F2, " c", 0F1, 0
;Minta:      "7.Breath 98 c16"

PrgExprsion   dw      8
               dw      3
               dw      0B001, 0B00, 0002
               db      0F0, ".Exprsn ", 0F2, " c", 0F1, 0
;Minta:      "5.Exprsn 112 c10"

PrgModulat    dw      8
               dw      3
               dw      0B001, 0100, 0002
               db      0F0, ".Modula ", 0F2, " c", 0F1, 0
;Minta:      "5.Modula 103 c09"

PrgPitchBend  dw      2
               dw      3
               dw      0E001, 0006, 0005
               db      0F0, ".Pitch", 0F5, " c", 0F1, 0
;Minta:      "7.PichB-3600 c11"

PrgDamperPedal dw      .512 ; érzékenység
               dw      3      ; küldendő üzenet hossza
               dw      0B001, 4000, 0007
               db      0F0, ".Damper ", 0F6, " c", 0F1, 0
;Minta:      "3.Damper off c02"

```

```
PrgVolumeAll    dw      8
                 dw     30
                 dw    0B000, 0700, 0002
                 dw    0B100, 0700, 0002
                 dw    0B200, 0700, 0002
                 dw    0B300, 0700, 0002
                 dw    0B400, 0700, 0002
                 dw    0B500, 0700, 0002
                 dw    0B600, 0700, 0002
                 dw    0B700, 0700, 0002
                 dw    0B800, 0700, 0002
                 dw    0B900, 0700, 0002
                 dw    0BA00, 0700, 0002
                 dw    0BB00, 0700, 0002
                 dw    0BC00, 0700, 0002
                 dw    0BD00, 0700, 0002
                 dw    0BE00, 0700, 0002
                 dw    0BF00, 0700, 0002
                 db    0F0, ".Volume ", 0F2, " all", 0
;Minta:          "6.Volume 103 all"

PrgSBFilCutoff  dw      8
                 dw      9
                 dw    0B001, 6300, 7F00, 6200, 1500
                 dw    0600, 4000, 2600, 0002
                 db    0F0, ".FiltCt ", 0F2, " c", 0F1, 0
;Minta:          "5.FiltCt 109 c09"

PrgSBResonance  dw      8
                 dw      9
                 dw    0B001, 6300, 7f00, 6200, 1600
                 dw    0600, 4000, 2600, 0002
                 db    0F0, ".Reso   ", 0F2, " c", 0F1, 0
;Minta:          "5.Reso      19 c09"

END
```

Keyb.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
#include "macro.inc"

#define      LcdRefresh  PrgStatus,0 ; ha 1, akkor a kijelzőt frissíteni kell
#define      SysExMode   PrgStatus,1 ; ha 1, akkor Sysex módban vagyunk
#define      Save        PrgStatus,2 ; ha 1, akkor összeállítás mentése szükséges

; billentyűzet mátrix
#define      KeybCol      PORTD          ; 8 mátrix oszlop
#define      KeybRow1     PORTC, 0       ; potenciométer szelektáló gombok
#define      KeybRow2     PORTC, 1       ; programválasztó funkciók gombjai
#define      KeybRow3     PORTC, 2       ; jelenleg nincs használva
#define      KeybRow1Tr   TRISC, 0       ; adat irány
#define      KeybRow2Tr   TRISC, 1       ; adat irány
#define      KeybRow3Tr   TRISC, 2       ; adat irány

; billentyűzet kiosztása
bPrgMinus    EQU 1    ; Prg- gomb
bPrgPlus     EQU 2    ; Prg+ gomb
bChnMinus    EQU 3    ; Chn- gomb
bChnPlus     EQU 4    ; Chn+ gomb
bSystemReset EQU 5    ; System Reset gomb
bBankSel     EQU 6    ; Bank Select gomb
bPresetMinus EQU 7    ; Preset- gomb
bPresetPlus  EQU 8    ; Preset+ gomb

GLOBAL      KeybInit, KeybPolling, PresetNum
GLOBAL      ADBanks, ADPrgs, ADChns

EXTERN      SaveData, SavePresetNum
EXTERN      PrgStatus
EXTERN      PrgBanks, ADMsgPtr, ADMsgSens, ADVal

; String interface
EXTERN      CurPos, WriteChar, WriteTableStr, ClearLine, LcdString, Num
EXTERN      WriteByte1, WriteByte2, WriteByte3 ; 1 bájtos előjel nélkül
EXTERN      WriteShort1, WriteShort2, WriteShort3 ; 1 bájtos előjeles
EXTERN      WriteWord3, WriteWord4, WriteWord5 ; 2 bájtos előjel nélkül
EXTERN      WriteInt3, WriteInt4, WriteInt5 ; 2 bájtos előjeles

EXTERN      PrgNameWrite, PrgNameWrite2

EXTERN      FifoFree, UsartTx

PresetSize   EQU .10 ; ennyi programösszeállítást használhatunk

UDATA_ACS
ADSelect     RES 1    ; aktuálisan kiválasztott potenciométer száma
BankSelect   RES 1    ; kiválasztott potenciométerhez tartozó bank
PrgSelect     RES 1    ; kiválasztott potenciométerhez tartozó program száma
PresetNum     RES 1    ; aktuális Preset száma (0..9)
Buttons       RES 1    ; billentyűzetmátrix aktuális sora
ButtonsPressd RES 1    ; lenyomott nyomógomb kódja
Count        RES 1    ; ciklusszámláló
ADNumLcd      RES 1    ; programkiírásnál ehhez a potenciométerhez száma

Programs      UDATA 0x0260
ADBanks       RES 8    ; potenciométerekhez hozzárendelt bankok száma
ADPrgs        RES 8    ; potenciométerekhez hozzárendelt programok száma
ADChns        RES 8    ; potenciométerekhez hozzárendelt csatornák száma

```

CODE

```

;=====
; nyolc program feltöltése adat EEPROM-ból Preset értékét PresetNum tartalmazza
PresetLoad      btfsc   Save
                  return                                ; ha mentés folyik, akkor kilépünk

                  movf    PresetNum, W
                  mullw   .24
                  movf    PRODL, W
                  addlw   SaveData
                  movwf   EEADR

; bankok feltöltése
                  lfsr    1, ADBanks
                  movlw   8
                  movwf   Count
PLoad1          bcf      EECON1, EEPGD
                  bcf      EECON1, CFGS
                  bsf      EECON1, RD
                  movff   EEDATA, POSTINC1
                  incf     EEADR, f
                  decfsz   Count, f
                  bra      PLoad1

; programok feltöltése
                  lfsr    1, ADPrgs
                  movlw   8
                  movwf   Count
PLoad2          bcf      EECON1, EEPGD
                  bcf      EECON1, CFGS
                  bsf      EECON1, RD
                  movff   EEDATA, POSTINC1
                  incf     EEADR, f
                  decfsz   Count, f
                  bra      PLoad2

; csatornák feltöltése
                  lfsr    1, ADChns
                  movlw   8
                  movwf   Count
PLoad3          bcf      EECON1, EEPGD
                  bcf      EECON1, CFGS
                  bsf      EECON1, RD
                  movff   EEDATA, POSTINC1
                  incf     EEADR, f
                  decfsz   Count, f
                  bra      PLoad3

; programmutatók feltöltése
                  movlw   8
                  movwf   Count
                  clrf    ADSelect
LoadPtrsLoop    rcall    MsgPtrLoad
                  incf     ADSelect, f
                  decfsz   Count, f
                  bra      LoadPtrsLoop

                  return

```

```

;=====
; ADSelect által kiválasztott üzenet mutatójának betöltése
MsgPtrLoad      ltblptr PrgBanks          ; Bank pointer tömb
                lfsrb 1, ADBanks, ADSelect
                bcf   STATUS, C
                rlcfc INDf1, W              ; bank száma
                addlw 2
                addwf TBLPTRL, f
                btfsc STATUS, C
                incf  TBLPTRH, f

                TBLRD *+
                movf  TABLAT, W
                TBLRD *+
                movff TABLAT, TBLPTRH
                movwf TBLPTRL

                lfsrb 1, ADPrgs, ADSelect
                bcf   STATUS, C
                rlcfc INDf1, W              ; program száma
                addlw 2
                addwf TBLPTRL, f
                btfsc STATUS, C
                incf  TBLPTRH, f

                lfsrw 1, ADMsgPtr, ADSelect

                tblrd *+
                movf  TABLAT, W
                movwf POSTINC1
                tblrd *+
                movff TABLAT, POSTINC1
                movff TABLAT, TBLPTRH
                movwf TBLPTRL

                lfsrw 1, ADMsgSens, ADSelect
                tblrd *+
                movff TABLAT, POSTINC1
                tblrd *+
                movff TABLAT, POSTINC1

                return

;=====
; billentyűzet inicializálása + beállítások betöltése EEPROM-ban tároltak alapján
KeybInit        bcf   Save
                movlw SavePresetNum
                movwf EEADR
                bcf   EECON1, EEPGD
                bcf   EECON1, CFGS
                bsf   EECON1, RD
                movff EEDATA, PresetNum      ; PresetNum
                rcall PresetLoad
                clrf  ADSelect                ; kiválasztott potenciométer = 0 (1.)
                lfsr 1, ADBanks
                movff INDf1, BankSelect      ; kiválasztott bank
                lfsr 1, ADPrgs
                movff INDf1, PrgSelect       ; kiválasztott program
                clrf  Buttons
                clrf  ButtonsPressd
                return

;=====
; Billentyűzet lekérdezése, feldolgozása Polling módszerrel
KeybPolling     bcf   KeybRow1Tr            ; kimenet legyen
                bcf   KeybRow1              ; alacsony logikai szint
                nop
                comf  KeybCol, W              ; sor visszaolvasása és negálása
                movwf Buttons
                bsf   KeybRow1              ; magas logikai szint
                bsf   KeybRow1Tr            ; bemenet legyen
                bz    KeybPrg                ; ha nincs lenyomva semmi

```

```

;-----
; potenciométer kiválasztó gombok lekérdezése, ADSelect beállítása
movlw    .16
movwf    CurPos
movf     ADSelect, W
btfsc    Buttons, 7
movlw    7
btfsc    Buttons, 6
movlw    6
btfsc    Buttons, 5
movlw    5
btfsc    Buttons, 4
movlw    4
btfsc    Buttons, 3
movlw    3
btfsc    Buttons, 2
movlw    2
btfsc    Buttons, 1
movlw    1
btfsc    Buttons, 0
movlw    0
movwf    ADSelect                ; ez lett kiválasztva

; kiválasztott potenciométer bankjának és programjának kiolvasása
lfsrb    1, ADBanks, ADSelect
movff    INDF1, BankSelect        ; Aktuálisan kiválasztott bank száma
lfsrb    1, ADPrGs, ADSelect
movff    INDF1, PrgSelect         ; Aktuálisan kiválasztott program száma

call     PrgNameWrite            ; nevét kiírni

return

;-----
; programválasztó gombok lekérdezése
KeybPrg    btfsc    Save
return

; Buttons 8 bitje = lenyomott gombok
bcf       KeybRow2Tr              ; kimenet
bcf       KeybRow2                ; 2. sor
nop
comf      KeybCol, W              ; sor visszaolvasása és negálása
movwf     Buttons
bsf       KeybRow2
bsf       KeybRow2Tr

bz        KeybNo                  ; egyik gomb sincs lenyomva

; ha le van nyomva, ButtonsPressd -be eltárolni hogy melyik
btfsc     Buttons, 0
movlw     bPrgMinus
btfsc     Buttons, 1
movlw     bPrgPlus
btfsc     Buttons, 2
movlw     bChnMinus
btfsc     Buttons, 3
movlw     bChnPlus
btfsc     Buttons, 4
movlw     bSystemReset
btfsc     Buttons, 5
movlw     bBankSel
btfsc     Buttons, 6
movlw     bPresetMinus
btfsc     Buttons, 7
movlw     bPresetPlus
movwf     ButtonsPressd           ; lenyomott gomb kódja
return                            ; majd felengéskor lekezeljük

```

```

;-----
; nincs lenyomva 1 billentyű sem, de lehet hogy most lett felengedve
KeybNo      movf    ButtonsPressd, W
            btfsc   STATUS, Z           ; előzőleg volt ?
            return  ; akkor sem -> kilépünk

            clrf    ButtonsPressd
            addlw   -1
            bz      KbPrgMinus          ; 'Program-' most felengedve
            addlw   -1
            bz      KbPrgPlus           ; 'Program+' most felengedve
            addlw   -1
            bz      KbChnMinus          ; 'Chn-' most felengedve
            addlw   -1
            bz      KbChnPlus           ; 'Chn+' most felengedve
            addlw   -1
            bz      KbSystemReset       ; 'System Reset' most felengedve
            addlw   -1
            bz      KbBankSel           ; 'Bank Select' most felengedve
            addlw   -1
            bz      KbPresetMinus       ; 'Preset-' most felengedve
            addlw   -1
            bz      KbPresetPlus        ; 'Preset+' most felengedve

            return

;-----
; program léptetés lefelé
KbPrgMinus  tstfsz  PrgSelect           ; skip ha Program == 0
            decf    PrgSelect, f       ; Program csökkentése
            bra     KbPrgEnd           ; ugyanaz a vége mint program+ -nál

;-----
; program léptetés felfelé
KbPrgPlus   ltblptr PrgBanks           ; Bank pointer tömb
            bcf     STATUS, C
            rlcfc   BankSelect, W     ; kiválasztott potenciométer bank száma
            addlw   2
            addwf   TBLPTRL, f
            btfsc   STATUS, C
            incf    TBLPTRH, f

; kiválasztott bank mutatója -> TBLPTR
            TBLRD   *+
            movf    TABLAT, W
            TBLRD   *+
            movff   TABLAT, TBLPTRH
            movwf   TBLPTRL

; ha kiválasztott bank programjainak száma <> PrgSelect + 1 -> PrgSelect++
            TBLRD   *+
            incf    PrgSelect, W      ; kiválasztott bank programjainak száma
            cpfseq  TABLAT
            incf    PrgSelect, f

; kiválasztott program eltárolása
KbPrgEnd    lfsrb   1, ADBanks, ADSelect
            movff   BankSelect, INDF1
            lfsrb   1, ADPrGs, ADSelect
            movff   PrgSelect, INDF1
            bsf     Save

; újonnan kiválasztott program betöltése és nevének kiírása
            rcall   MsgPtrLoad
            movf    ADSelect, W
            call    PrgNameWrite
            return

;-----
; csatorna léptetés lefelé
KbChnMinus  lfsrb   1, ADChns, ADSelect
            tstfsz  INDF1              ; 0 ?
            decf    INDF1, f           ; nem -> csatorna--
            movf    ADSelect, W
            call    PrgNameWrite      ; név kiírása
            bsf     Save              ; menteni
            return

```

```

;-----
; csatorna léptetés felfelé
KbChnPlus      lfsrb     1, ADChns, ADSelect
               movlw     .15
               subwf     INDF1, W           ; 15 ?
               btfss     STATUS, Z
               incf      INDF1, f           ; nem -> csatorna++
               movf      ADSelect, W
               call      PrgNameWrite      ; LCD-re kiírni
               bsf       Save              ; menteni
               return

;-----
; Peset Minus gomb felengedése esetén előző beállításcsoport választása
KbPresetMinus   movf     PresetNum, W
               bz        $ + 6              ; 0 -> nem csökkentjük
               decf      PresetNum, f
               rcall     PresetLoad
               bra       PresetWrite

;-----
; Peset Plus gomb felengedése esetén következő beállításcsoport választása
KbPresetPlus    movf     PresetNum, W
               sublw     PresetSize - 1
               bz        $ + 6              ; már az utolsó -> nem növeljük
               incf      PresetNum, f
               rcall     PresetLoad
               bra       PresetWrite

;-----
; System Reset gomb felengedése esetén control reset üzenet az összes csatornára
KbSystemReset   btfsc    SysExMode
               return                                ; SysEx módban nem küldünk semmit
               movlw     4
               cpfsgt    FifoFree
               return                                ; nincs elég hely a Fifoban

               movlw     0xFF
               call      UsartTx                ; aktuális bájt küldése soros porton

               bsf       INTCON, GIE            ; Irq tiltás feloldása

               clrf      CurPos
               call      ClearLine              ; első sort letörölni
               ltblptr   SysResetText
               call      WriteTableStr          ; második sorba " System Reset "
               bsf       LcdRefresh
               return

;-----
; Bank Select gomb felengedése esetén
KbBankSel       ltblptr  PrgBanks              ; Bank pointer tömb
               tblrd     *

; program szám nullázása hogy a másik bankból előlről induljunk
               clrf      PrgSelect

               incf      BankSelect, f          ; bank megnövelése

; ha elértük az utolsó bankot, akkor kinullázzuk (körbejárunk)
               movf      TABLAT, W
               cpfslt    BankSelect
               clrf      BankSelect

; TBLPTR = kiválasztott bank mutatója
               bcf       STATUS, C
               rlc       BankSelect, W
               addlw     2
               addwf     TBLPTRL, f
               btfsc     STATUS, C
               incf      TBLPTRH, f
               tblrd     *+
               movf      TABLAT, W
               tblrd     *+
               movff     TABLAT, TBLPTRH
               movwf     TBLPTRL

```

```

; TBLPTR = kiválasztott bank szövege
tblrd    *
rlcf     TABLAT, W
addlw    2
addwf    TBLPTRL, f
btfsc    STATUS, C
incf     TBLPTRH, f

; kurzorpoz = 0, első sorba bank neve, második sor törlés
clrf     CurPos
call     WriteTableStr
call     ClearLine
bsf      LcdRefresh
return

;-----
; LCD kijelzőre kiírni az aktuális Preset számát
PresetWrite  clrf     CurPos
              ltblptr PresetText
              call     WriteTableStr
              incf     PresetNum, W
              call     WriteByte2
              bsf      Save
              call     ClearLine
              bsf      LcdRefresh

; kiválasztott potenciométer bankjának és programjának kiolvasása
              lfsrb    1, ADBanks, ADSelect
              movff     INDF1, BankSelect          ; Aktuálisan kiválasztott bank száma
              lfsrb    1, ADPrGs, ADSelect
              movff     INDF1, PrgSelect           ; Aktuálisan kiválasztott program száma
              return

;=====
PresetText    db      "Preset Change ", 0
SysResetText  db      "  System Reset  ", 0

;=====

END

```

ADConv.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
#include "macro.inc"

#define      LcdRefresh PrgStatus,0 ; ha 1, akkor a kijelzőt frissíteni kell
#define      SysExMode  PrgStatus,1 ; ha 1, akkor Sysex módban vagyunk

GLOBAL      ADInit, ADPolling
GLOBAL      ADMsgPtr, ADMsgSens, ADVal

EXTERN      PrgStatus
EXTERN      PrgBanks, PrgNameWrite2, ADChns, ADNumLcd
EXTERN      FifoFree, UsartTx

UDATA_ACS
    ADNum      RES      1                      ; aktuális A/D csatorna (0..7)
    ADAbsDelta RES      2
    Count      RES      1
    Temp       RES      1

ADArrays      UDATA 0x0220
    ADVal      RES      .16
    ADMsgVal    RES      .16
    ADMsgSens   RES      .16
    ADMsgPtr    RES      .16

CODE

;=====
ADInit      movlw    B'00000111'
            movwf    ADCON1                      ; VREF=táp, AN0-AN7 A/D be
            movlw    B'10111110'                  ; jobbra igazít, Acq Time = 20
            movwf    ADCON2                      ; clock = Fosc/64

            clrf     ADNum
            lfsr     0, ADVal
            lfsr     1, ADMsgVal

ADInitLoop
            bcf      STATUS, C
            rlcwf    ADNum, W
            rlcwf    WREG, W
            movwf    ADCON0
            bsf      ADCON0, ADON
            bsf      ADCON0, GO

            btfsc    ADCON0, DONE                  ; befejezte az A/D átalakítást ?
            bra      $ - 2

            movff    ADRESL, POSTINC0
            movff    ADRESH, POSTINC0
            movff    ADRESL, POSTINC1
            movff    ADRESH, POSTINC1

            incf     ADNum, f
            movlw    8
            cpfseq   ADNum
            bra      ADInitLoop

            clrf     ADNum
            clrf     ADCON0
            bsf      ADCON0, ADON
            bsf      ADCON0, GO
            return

```

```

;=====
ADPolling      btfscl ADCON0, DONE          ; befejezte az A/D átalakítást ?
               return                      ; nem
               btfscl SysExMode             ; sysex mód ?
               return                      ; igen -> nem küldünk saját üzenetet

; aktuális érték elmentése ADVal tömbbe
               lfsrw 1, ADVal, ADNum
               movff ADRESL, POSTINCL
               movff ADRESH, POSTINCL

; utolsó üzenetkor mennyi volt az A/D értéke ?
               lfsrw 1, ADMsgVal, ADNum

               movff POSTINCL, ADAbsDelta
               movff POSTDECL, ADAbsDelta + 1

; ADAbsDelta = régi A/D - aktuális A/D
               movf ADRESL, W
               subwf ADAbsDelta, f
               movf ADRESH, W
               subwfb ADAbsDelta + 1, f

               bnn ADPDeltaPoz              ; pozitív különbség

; negatív különbségnél előjelfordítás
               comf ADAbsDelta + 1, f
               negf ADAbsDelta
               btfscl STATUS, C
               incf ADAbsDelta + 1

ADPDeltaPoz    lfsrw 1, ADMsgSens, ADNum    ; érzékenység tömb megcímzése

; ADAbsDelta = változás - érzékenység
               movf POSTINCL, W
               subwf ADAbsDelta, f
               movf INDF1, W
               subwfb ADAbsDelta + 1, f

               bn ADNext                    ; érzékenység alatti változás

;-----
; üzenet küldése
               lfsrw 1, ADMsgPtr, ADNum

               movff POSTINCL, TBLPTRL
               movff POSTDECL, TBLPTRH
               tblrd *+
               tblrd *+

               tblrd *+
               movf TABLAT, W
               addlw 6
               cpfsqt FifoFree
               bra ADNext                  ; nincs elég hely a Fifoban

               movff TABLAT, Count
               tblrd *+

               bcf INTCON, GIE             ; Irq tiltás, Fifo használat miatt

```

```

; üzenetbájt típus szerinti elágazás
ADMsgLoop      tblrd    *+
                bcf      STATUS, C
                movf     PCL, W
                rlcfc    TABLAT, W
                tblrd    *+
                addwf    PCL, f
                bra      ADMsgDT0
                bra      ADMsgDT1
                bra      ADMsgDT2
                bra      ADMsgDT3
                bra      ADMsgDT4
                bra      ADMsgDT5
                bra      ADMsgDT6
                bra      ADMsgDT7
                bra      ADMsgDT0
                bra      ADMsgDT0
                bra      ADMsgDT0
                bra      ADMsgDT0
                bra      ADMsgDT0
                bra      ADMsgDT0
                bra      ADMsgDT0
                bra      ADMsgDT0

;-----
; simán az értéket elküldeni
ADMsgDT0        movf     TABLAT, W
                bra      ADMsgSendByte

;-----
; + csatornaszám
ADMsgDT1        lfsrb    1, ADChns, ADNum
                movf     TABLAT, W
                addwf    INDF1, W
                bra      ADMsgSendByte

;-----
; előjel nélküli 7 bites
ADMsgDT2        movff    ADRESL, Temp
                rrcf     ADRESH, W
                rrcf     Temp, f
                rrcf     ADRESH, W
                rrcf     WREG, W
                rrcf     Temp, f
                bcf      STATUS, C
                rrcf     Temp, W
                bra      ADMsgSendByte

;-----
; előjeles 7 bites hi, -64..63 hi (0..511 -> 0x3F, 512..1023 -> 0x40
ADMsgDT3        movlw    3F
                btfsc    ADRESH, 1
                movlw    40
                bra      ADMsgSendByte

; előjeles 7 bites lo, -64..63 lo
ADMsgDT4        rrcf     ADRESH, W
                rrcf     ADRESL, W
                bcf      STATUS, C
                rrcf     WREG, W
                bcf      STATUS, C
                rrcf     WREG, W
                addlw    40
                andlw    7F
                bra      ADMsgSendByte

```

```

;-----
; 14 bites hi
ADMsgDT5      movff    ADRESL, Temp
              rrcf     ADRESH, W
              rrcf     Temp, f
              rrcf     WREG, W
              rrcf     Temp, f
              bcf      STATUS, C
              rrcf     Temp, W
              bra      ADMsgSendByte

; 14 bites lo
ADMsgDT6      swapf    ADRESL, W
              andlw    070
              bra      ADMsgSendByte

;-----
; kapcsoló jellegű
ADMsgDT7      clrf     ADRESL
              btfsc    ADRESH, 1
              bra      ADMsgDT7on
              movlw    HIGH .256
              movwf    ADRESH
              movlw    0
              bra      ADMsgSendByte

ADMsgDT7on    movlw    HIGH .768
              movwf    ADRESH
              movlw    .64
              bra      ADMsgSendByte

;-----
ADMsgSendByte call     UsartTx                      ; bájt küldése soros porton
              decfsz   Count
              bra      ADMsgLoop

              bsf      INTCON, GIE                  ; Irq tiltás feloldása

; elküldött üzenethez tartozó A/D eltárolása
              lfsrw    1, ADMsgVal, ADNum
              movff    ADRESL, POSTINC1
              movff    ADRESH, POSTINC1

; elküldött üzenet megjelenítése
              movff    ADNum, ADNumLcd
              call     PrgNameWrite2

;-----
; következő A/D csatorna, konverzió start
ADNext        incf     ADNum, f
              bcf      ADNum, 3                      ; csak 0..7 lehet
              bcf      STATUS, C
              rlcw     ADNum, W
              rlcw     WREG, W
              movwf    ADCON0
              bsf      ADCON0, ADON
              bsf      ADCON0, GO

              return

END

```

PrgWrite.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
#include "macro.inc"

#define          LcdRefresh PrgStatus,0 ;ha 1, akkor a kijelzõt frissíteni kell

GLOBAL          PrgNameWrite, PrgNameWrite2, ADNumLcd

EXTERN          PrgStatus
EXTERN          PrgBanks, ADVal, ADMsgPtr, ADBanks, ADChns
EXTERN          CurPos, WriteChar, Num, WriteTableStr
EXTERN          WriteByte1, WriteByte2, WriteByte3      ; 1 bájtos előjel nélkül
EXTERN          WriteShort1, WriteShort2, WriteShort3   ; 1 bájtos előjeles
EXTERN          WriteWord3, WriteWord4, WriteWord5      ; 2 bájtos előjel nélkül
EXTERN          WriteInt3, WriteInt4, WriteInt5         ; 2 bájtos előjeles

UDATA_ACS
    ADNumLcd      RES      1      ; programkiírásnál ehhez a potenciométerhez száma
    Temp          RES      1      ; átmeneti tároláshoz

CODE

;=====
; LCD re egy A/D-hez tartozó üzenet nevét és paramétereit
; be: W - A/D átalakító száma (0..7)
PrgNameWrite     movwf    ADNumLcd
                  lfsrw   1, ADMsgPtr, ADNumLcd
                  movff   POSTINC1, TBLPTRL
                  movff   POSTDEC1, TBLPTRH
                  tblrd   ++
                  tblrd   ++
                  tblrd   ++
                  bcf     STATUS, C
                  rlc     TABLAT, W                ; üzenet adatainak hossza
                  tblrd   ++
                  addwf   TBLPTRL, f
                  btfsc   STATUS, C
                  incf    TBLPTRH, f

; ha nem kell a programmemória táblázat címét kiszámítani itt indulhat
PrgNameWrite2    movlw    .16
                  movwf   CurPos

PrgNameLoop      tblrd   ++

                  movf    TABLAT, W
                  bz      LcdEnd                    ; #0 karakter a szöveg vége

                  movf    TABLAT, W
                  addlw   0x10
                  bc      LcdCmd

                  movf    TABLAT, W
                  call    WriteChar
                  bra     PrgNameLoop

; bank nevének kiírása a felső sorba
LcdEnd           ltblptr PrgBanks
                  lfsrb   1, ADBanks, ADNumLcd
                  bcf     STATUS, C
                  rlc     INDF1, W
                  addlw   2
                  addwf   TBLPTRL, f
                  btfsc   STATUS, C
                  incf    TBLPTRH, f
                  tblrd   ++
                  movf    TABLAT, W
                  tblrd   ++
                  movff   TABLAT, TBLPTRH
                  movwf   TBLPTRL

```

```

; TBLPTR = kiválasztott bank szövege
    tblrd    *
    rlcfc    TABLAT, W
    addlw    2
    addwfc   TBLPTRL, f
    btfsc    STATUS, C
    incf     TBLPTRH, f

    clrfs    CurPos
    call     WriteTableStr
    bsf      LcdRefresh
    return

; vezérlőkarakter (F0..FF -> 00..0F) szerinti elágazás
LcdCmd      bcf      STATUS, C
            movff     PCL, Temp
            rlcfc     WREG, W
            addwfc    PCL, f
            bra       LcdMF0           ; potenciométer száma
            bra       LcdMF1           ; csatorna száma
            bra       LcdMF2           ; A/D eredmény előjel nélküli 7 bites
(0..127)
            bra       LcdMF3           ; A/D eredmény előjeles 7 bites (-64..63)
            bra       LcdMF4           ; A/D eredmény előjel nélküli 14 bites
(0..16383)
            bra       LcdMF5           ; A/D eredmény előjeles 14 bites (-
8192..8191)
            bra       LcdMF6           ; A/D eredmény kétállapotú (off, on)
; ezek a parancskódok jelenleg nincsenek használva, nem csinál semmit (hurok elejére vissza)
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop
            bra       PrgNameLoop

; -----
; potenciométer száma
LcdMF0      movf     ADNumLcd, W
            addlw    '1'
            call     WriteChar
            bra       PrgNameLoop

; -----
; csatorna szám
LcdMF1      lfsrb    1, ADChns, ADNumLcd
            incf     INDF1, W
            call     WriteByte2
            bra       PrgNameLoop

; -----
; A/D eredmény előjel nélküli 7 bites (0..127)
LcdMF2      lfsrw    1, ADVal, ADNumLcd
            movff     POSTINC1, Temp           ; lo
            rrcf      INDF1, W                 ; hi
            rrcf      Temp, f
            rrcf      INDF1, W                 ; hi
            rrcf      WREG, W
            rrcf      Temp, f
            bcf       STATUS, C
            rrcf      Temp, W
            call     WriteByte3
            bra       PrgNameLoop

```

```

;-----
; A/D eredmény előjeles 7 bites (-64..63)
LcdMF3      lfsrw    1, ADVal, ADNumLcd
            movff    POSTINC1, Temp          ; lo
            rrcf     INDF1, W                ; hi
            rrcf     Temp, f
            rrcf     INDF1, W                ; hi
            rrcf     WREG, W
            rrcf     Temp, f
            bcf      STATUS, C
            rrcf     Temp, W
            addlw    -.64
            call     WriteShort2
            bra      PrgNameLoop

;-----
; A/D eredmény előjel nélküli 14 bites (0..16383)
LcdMF4      lfsrw    1, ADVal, ADNumLcd
            swapf    INDF1, W
            andlw    0xF0
            movwf    Num
            swapf    POSTINC1, W
            andlw    0x0F
            movwf    Num + 1
            swapf    INDF1, W
            andlw    0xF0
            iorwf    Num + 1, f
            call     WriteWord5
            bra      PrgNameLoop

;-----
; A/D eredmény előjeles 14 bites (-8192..8191)
LcdMF5      lfsrw    1, ADVal, ADNumLcd
            swapf    INDF1, W
            andlw    0xF0
            movwf    Num
            swapf    POSTINC1, W
            andlw    0x0F
            movwf    Num + 1
            swapf    INDF1, W
            andlw    0xF0
            iorwf    Num + 1, f
            movlw    HIGH .8192
            subwf    Num + 1, f
            call     WriteInt4
            bra      PrgNameLoop

;-----
; A/D eredmény kétállapotú (off, on)
; On/Off szöveg íródik ki az A/D érték legnagyobb helyiértékétől függően
LcdMF6      lfsrw    1, ADVal, ADNumLcd
            btfsc    PREINC1, 1
            bra      LcdMF6on
            movlw    'o'
            call     WriteChar
            movlw    'f'
            call     WriteChar
            movlw    'f'
            call     WriteChar
            bra      PrgNameLoop

LcdMF6on    movlw    ' '
            call     WriteChar
            movlw    'o'
            call     WriteChar
            movlw    'n'
            call     WriteChar
            bra      PrgNameLoop

END

```

Usart.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
#include "macro.inc"

#define SysExMode PrgStatus,1 ; ha 1, akkor Sysex módban vagyunk

GLOBAL UsartInit, UsartRxIrq, UsartTxIrq, UsartTx
EXTERN FifoInit, FifoWrite, FifoRead
EXTERN FifoLength, FifoFree, PrgStatus
EXTERN CurPos, WriteTableStr

    BaudRate    EQU    .79        ; 31250 bit/sec az UART sebesseg (40MHz)

UDATA_ACS
    AktStatus    RES    1        ; milyen státuszbajtot küldtünk utoljára

    RxData        RES    1        ; RX-en vett byte
    RxPuff        RES    3        ; vett üzenet 3 bájtos átmeneti tárolója
    RxCount       RES    1        ; üzenet bájt számlálója (0,1,2)
    RxSize        RES    1        ; éppen vétel alatt levő üzenet hossza (1,2,3)

    Fsr2Tmp       RES    2        ; adat memória muataó mentéshez
    TblPtrTmp     RES    2        ; program memória mutató mentéshez
    CurPosTmp     RES    1        ; kurzor pozíció mentéshez

CODE
;=====
; soros port beállításai
UsartInit
    clrf    AktStatus
    call    FifoInit

    movlw   1
    movwf   RxCount
    movwf   RxSize
    bcf     SysExMode

    movlw   B'00100100'        ; átvitel eng, aszinkron, BRGH = high speed
    movwf   TXSTA
    movlw   BaudRate           ; 31250 bit/sec
    movwf   SPBRG
    bcf     TRISC, 6            ; TX láb kimenet

    movlw   B'10010000'        ; soros port eng, 8bites mód, vétel eng
    movwf   RCSTA

    bcf     IPRI1, TXIP         ; Tx irq alacsony prioritás

    bsf     IPRI1, RCIP         ; RX irq magas prioritás
    bsf     PIE1, RCIE         ; RX irq eng.

    return

;=====
; soros port adás megszakítás kiszolgálás
UsartTxIrq
    call    FifoRead
    bc      UsartStop           ; ha üres adás stop
    movwf   TXREG
    return

UsartStop
    bcf     PIE1, TXIE         ; TX irq tiltás
    return

;=====
; egy bájt küldése soros porton, ha a port foglalt akkor FIFO regiszterbe írja az adatot
; be: W = küldendő bájt
UsartTx
    btfss   WREG, 7            ; státuszbajt ?
    bra     TxSend2            ; nem, ezért biztosan adni kell
    addlw   -0xFE              ; activesense ? (FE)
    bz      TxSend1            ; igen
    addlw   0xFE
    cpfseq  AktStatus          ; == utolsó státuszbajt ?

```

```

        bra    $ + 4                ; nem
        return                       ; igen -> nem kell küldeni
        movwf  AktStatus             ; mostantól ez az aktuális státusz
        bra    TxSend2

TxSend1    addlw  0xFE

TxSend2    btfsc  PIR1, TXIF          ; Usart reg üres ?
        bra    TxSendStart          ; igen

        call   FifoWrite             ; már csak a Fifo-ba fér
        bsf    PIE1, TXIE           ; TX irq eng.
        return

TxSendStart    movwf  TXREG
        return

;=====
; soros port vétel megszakítás kiszolgálás
UsartRxIrq    btfsc  RCSTA, FERR      ; keret hiba ?
        return      ; igen
        btfsc  RCSTA, OERR          ; túlfutási hiba ?
        return      ; igen

        movf    RCREG, W
        movwf   RxData              ; érkezett byte
        btfss   SysExMode           ; Sysex módban vagyunk ?
        bra     NoSysExRx           ; nem
        addlw   -0xF7              ; Sysex mód vége ?
        btfss   STATUS, Z
        bra     RxStatus1b          ; nem
        bcf     SysExMode           ; Sysex mód kikapcs

        fsrs    2, Fsr2Tmp
        tblptrs TblPtrTmp
        movff   CurPos, CurPosTmp
        clrf    CurPos              ; kurzorpozíció mentése
        ltblptr SysExOffText
        call    WriteTableStr       ; Lcd-re kiírni
        movff   CurPosTmp, CurPos   ; kurzorpozíció visszaállítása
        fsrl    2, Fsr2Tmp
        tblptrl TblPtrTmp

        bra     RxStatus1b          ; 1 bájtos üzenet kiírása

; nem Sysex módban vagyunk
NoSysExRx

        btfsc   RxData, 7           ; státusz bájt ?
        bra     RxStatus            ; igen

        decf    RxCount, W
        btfsc   STATUS, Z
        movff   RxData, RxPuff + 1
        addlw   -1
        btfsc   STATUS, Z
        movff   RxData, RxPuff + 2

        decf    RxSize, W
        cpfseq  RxCount             ; üzenet vége ?
        bra     RxNoEnd            ; nem

; Puffer tartalmát kiküldi a soros kimenetre
RxPuffSend    fsrs    2, Fsr2Tmp

        movf    RxPuff, W
        rcall   UsartTx             ; 1. bájt
        movlw   1
        subwf   RxSize, W
        bz      RxPuffSendEnd       ; csak 1 bájt hosszú az üzenet

        movf    RxPuff + 1, W
        rcall   UsartTx             ; 2. bájt
        movlw   2
        subwf   RxSize, W
        bz      RxPuffSendEnd       ; csak 2 bájt hosszú az üzenet

        movf    RxPuff + 2, W

```

```

                                rcall    UsartTx                ; 3. bájt

RxPuffSendEnd    fsrl      2, Fsr2Tmp
                                movlw    1
                                movwf    RxCount
                                return

RxNoEnd          incf      RxCount, f
                                return

; státuszbájt jött
RxStatus         addlw     0x10
                                btfsc    STATUS, C
                                bra       RxStatusNcs          ; >F0
                                addlw     0x30
                                btfsc    STATUS, C
                                bra       RxStatus2b           ; >C0

; 3 bájtos üzenet
RxStatus3b       movlw     1
                                movwf    RxCount
                                movlw     3
                                movwf    RxSize                ; aktuális üzenet hossza = 3
                                movff     RxData, RxPuff
                                return

; 2 bájtos üzenet
RxStatus2b       movlw     1
                                movwf    RxCount
                                movlw     2
                                movwf    RxSize                ; aktuális üzenet hossza = 2
                                movff     RxData, RxPuff
                                return

; 1 bájtos üzenet
RxStatus1b       movf       RxData, W
                                fsrs      2, Fsr2Tmp            ;FSR2 mentése
                                rcall     UsartTx                ;rögtön a kimenetre küldeni
                                fsrl      2, Fsr2Tmp            ;FSR2 visszaállítása
                                return

; nem csatorna üzenet (F0..FF)
RxStatusNcs      btfsc     STATUS, Z
                                bra       RxStatusSeBe          ;F0
                                addlw     -1
                                btfsc     STATUS, Z
                                bra       RxStatus2b             ;F1
                                addlw     -1
                                btfsc     STATUS, Z
                                bra       RxStatus3b             ;F2
                                addlw     -12
                                btfsc     STATUS, Z
                                return                             ;FE
                                bra       RxStatus1b

; Sysex be
RxStatusSeBe     bcf       SysExMode

                                fsrs      2, Fsr2Tmp
                                tblptrs   TblPtrTmp
                                movff     CurPos, CurPosTmp
                                clrf      CurPos                ; kurzorpozíció mentése
                                ltblptr   SysExOnText
                                call      WriteTableStr          ; Lcd-re kiírni
                                movff     CurPosTmp, CurPos        ; kurzorpozíció visszaállítása
                                fsrl      2, Fsr2Tmp
                                tblptrl   TblPtrTmp

                                bra       RxStatus1b

SysExOnText      db        " SyEx Mode On  ", 0
SysExOffText     db        " SyEx Mode Off ", 0

END

```

Fifo.asm

```
LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
```

```
GLOBAL FifoInit           ; Fifo inicialiálása
GLOBAL FifoWrite          ; Fifo-ba egy byte beírása W-ből, ha tele van C=1 el jelzi
GLOBAL FifoRead           ; Fifo-ból egy byte olvasása W-be, ha üres C=1 el jelzi
GLOBAL FifoLength         ; ennyi elem van éppen benne
GLOBAL FifoFree           ; ennyi szabad hely van épp benne
```

```
UDATA_ACS
    FifoStart      RES 1      ; Ez mondja meg hogy a tömbben hol található a cső eleje
    FifoLength     RES 1      ; Ennyi elem van most benne
    FifoFree       RES 1      ; Ennyi hely van még benne
    FifoTemp       RES 1      ; átmeneti tár
```

```
FifoBank      UDATA 0x0300
    FifoArray     RES .128    ; 128 bájt hosszú a kimeneti puffer
    FifoSize      EQU .128    ; Puffer mérete
```

```
CODE
```

```
;-----
; Fifo inicialiálása
FifoInit      clrf    FifoStart          ; eleje = 0
              clrf    FifoLength        ; hossza = 0
              movlw   FifoSize
              movwf   FifoFree          ; szabad hely = teljes méret
              return
```

```
;-----
; egy bájt írása FIFO tárolóba
; ha tele van C = 1 visszatérő értékkel jelzi
FifoWrite     movwf   FifoTemp
              movf    FifoFree, W
              bsf     STATUS, C          ; megtelet ?
              btfsc   STATUS, Z
              return                    ; igen megtelt -> C = 1

              lfsr    2, FifoArray
              movf    FifoStart, W
              addwf   FSR2L, f
              movff   FifoTemp, INDF2    ; adat a tömbbe
              incf    FifoLength, f      ; csőhossz növelése
              decf    FifoFree, f        ; szabad hely csökkentése
              movf    FifoStart, f
              addlw   -1
              andlw   FifoSize - 1
              movwf   FifoStart          ; cső eleje hátra 1-el
              bcf     STATUS, C          ; rendben jelzés
              return
```

```
;-----
; egy bájt olvasása Fifo tárolóból
; olvasott bájt -> W
; ha üres -> C = 1
FifoRead      movf    FifoLength, W
              bsf     STATUS, C
              btfsc   STATUS, Z          ; üres ?
              return                    ; igen üres -> C = 1

              lfsr    2, FifoArray
              movf    FifoStart, W
              addwf   FifoLength, W      ; eleje + hossz
              andlw   FifoSize - 1      ; körbejárás miatt
              addwf   FSR2L, f
              decf    FifoLength, f      ; rövidíteni
              incf    FifoFree, f        ; szabad hely számát növelni
              movf    INDF2, W          ; kivett adat
              bcf     STATUS, C          ; rendben
              return
```

```
;-----
END
```

Lcd.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>

#define      LcdData PORTB
#define      LcdDtTr TRISB
#define      LcdRs    PORTB,4
#define      LcdRsTr  TRISB,4
#define      LcdRw    PORTB,5
#define      LcdRwTr  TRISB,5
#define      LcdE     PORTA,4
#define      LcdETr   TRISA,4

#define      LcdRefresh PrgStatus,0 ; ha 1, akkor a kijelzőt frissíteni kell

GLOBAL      LcdInit, StringToLcd
EXTERN      CurPos, WriteChar, WriteTableStr
EXTERN      Delayusec, Delaymsec
EXTERN      LcdString, PrgStatus

; LCD Modul parancsok
DISP_ON     EQU    0x00C    ; kijelzés be
DISP_ON_C   EQU    0x00E    ; kijelzés be, kurzor be
DISP_ON_B   EQU    0x00F    ; kijelzés be, kurzor be, villogó kurzor
DISP_OFF    EQU    0x008    ; kijelzés ki
CLR_DISP    EQU    0x001    ; képernyő törlés
ENTRY_INC   EQU    0x006    ;
ENTRY_INC_S EQU    0x007    ;
ENTRY_DEC   EQU    0x004    ;
ENTRY_DEC_S EQU    0x005    ;
DD_RAM_ADDR EQU    0x080    ; alsó 7 bit adja meg a címet
DD_RAM_UL   EQU    0x080    ; bal felső sarla az LCD-nek

UDATA_ACS
Count       RES     1        ; számláló
Char        RES     1        ; aktuális karakter tárolására
Temp        RES     1        ; átmeneti tároláshoz

CODE

;=====
; adatport irányát Lcd-re történő írásra állítjuk
LcdTrisWrite macro
    movlw    0xF0
    andwf    LcdDtTr, f
endm

;=====
; adatport irányát Lcd-ről való olvasásra állítjuk
LcdTrisRead  macro
    movlw    0x0F
    iorwf    LcdDtTr, f
endm

;=====
; 4 bites adat írása LCD-re (bekapcsoláskori inicializáláshoz)
LcdDataWrite4
    movlw    0xF0
    andwf    LcdData, f
    movf     Char, W
    andlw    0x0F
    iorwf    LcdData, f
    bsf      LcdE
    nop
    nop
    nop
    bcf      LcdE
    return

```

```

;=====
; 8 bites adat írása LCD-re
LcdDataWrite8
    movlw    0xF0
    andwf   LcdData, f
    swapf   Char, W
    andlw   0x0F
    iorwf   LcdData, f ;felső 4 bit
    nop
    nop
    bsf     LcdE
    nop
    nop
    nop
    bcf     LcdE

    movlw    0xF0
    andwf   LcdData, f
    movf     Char, W
    andlw   0x0F
    iorwf   LcdData, f ;alsó 4 bit
    nop
    nop
    bsf     LcdE
    nop
    nop
    nop
    bcf     LcdE

    return

;=====
; várakozás arra hogy az LCD kijelző adatot tudjon fogadni
LcdWaitBusy
    LcdTrisRead
Loop    bcf     LcdRs
        bsf     LcdRw
        nop
        nop
        nop
        bsf     LcdE
        nop
        nop
        nop
        swapf   LcdData, W
        bcf     LcdE
        andlw   0xF0
        movwf   Temp
        nop
        nop
        nop
        bsf     LcdE
        nop
        nop
        nop
        movf     LcdData, W
        bcf     LcdE
        andlw   0x0F
        iorwf   Temp, f
        bcf     LcdRw
        btfsc   Temp, 7
        bra     Loop
    LcdTrisWrite
    return

```

```

;=====
; LCD inicializálása
LcdInit      bcf      LcdRsTr      ; RS = 0
              bcf      LcdRwTr      ; R/W = 0
              bcf      LcdETr      ; E = 0
              LcdTrisWrite      ; Lcd adatlabakra menő lábak kimenetek
;----- 1
              movlw    .100
              call     Delaymsec
              bcf      LcdRs
              bcf      LcdRw
              movlw    B'0011'
              movwf    Char
              rcall    LcdDataWrite4

;----- 2
              movlw    .5
              call     Delaymsec

              movlw    B'0011'
              movwf    Char
              rcall    LcdDataWrite4

;----- 3
              movlw    .100
              call     Delayusec

              movlw    B'0011'
              movwf    Char
              rcall    LcdDataWrite4

;----- 4
              movlw    .100
              call     Delayusec

              movlw    B'0010'
              movwf    Char
              rcall    LcdDataWrite4

;-----
              movlw    B'00101011' ; 4 bites, többsoros
              rcall    LcdSendCmd   ; parancs küldés
              movlw    DISP_ON
              rcall    LcdSendCmd   ; kijelzés be
              movlw    CLR_DISP
              rcall    LcdSendCmd   ; képernyő törlés
              movlw    ENTRY_INC
              rcall    LcdSendCmd   ; kurzor automatikus léptetése
              movlw    DD_RAM_ADDR
              rcall    LcdSendCmd   ; bal felső sarokba menni
              return

;=====
; egy karakter kiküldése az LCD kijelzőre
LcdSendChar   movwf    Char
              rcall    LcdWaitBusy
              bcf      LcdRw
              bsf      LcdRs
              rcall    LcdDataWrite8
              return

;=====
; egy parancs kiküldése az LCD kijelzőre
LcdSendCmd    movwf    Char
              rcall    LcdWaitBusy
              bcf      LcdRw
              bcf      LcdRs
              rcall    LcdDataWrite8
              return

```

```
=====
; 32 bájtos LcdString-ben található szöveg átvitele az LCD kijelzőre
StringToLcd    bcf      LcdRefresh
               lfsr     0, LcdString
               movlw    DD_RAM_UL
               rcall    LcdSendCmd
               movlw    .16
               movwf    Count
Sor1Loop       movf     POSTINC0, W
               rcall    LcdSendChar
               decfsz    Count
               bra      Sor1Loop

               movlw    DD_RAM_ADDR + 0x40
               rcall    LcdSendCmd

               movlw    .16
               movwf    Count
Sor2Loop       movf     POSTINC0, W
               rcall    LcdSendChar
               decfsz    Count
               bra      Sor2Loop

               return

=====
END
```

String.asm

```

LIST P=18F4550, F=INHX32
#include <P18F4550.INC>
#include "macro.inc"

GLOBAL      CurPos, LcdString, Num
GLOBAL      ClrScr, WriteChar
GLOBAL      WriteByte1, WriteByte2, WriteByte3      ; 1 bájtos előjel nélkül
GLOBAL      WriteShort1, WriteShort2, WriteShort3    ; 1 bájtos előjeles
GLOBAL      WriteWord3, WriteWord4, WriteWord5       ; 2 bájtos előjel nélkül
GLOBAL      WriteInt3, WriteInt4, WriteInt5          ; 2 bájtos előjeles
GLOBAL      WriteTableStr, ClearLine

UDATA_ACS
    CurPos      RES 1      ; kurzor pozíciója
    Char        RES 1      ; aktuális karakter
    Ptr         RES 1      ; karakter mutató

    Num         RES 2      ; szám -> karakter átalakításhoz
    WTemp       RES 1
    StrStatus   RES 1

#define        ZeroSw  StrStatus, 0
#define        NegSw   StrStatus, 1

LcdArrays     UDATA 0x0200
    LcdString   RES .32    ;
    LcdLenght   EQU .32    ; Puffer merete
    LcdPtrMask  EQU LcdLenght - 1

CODE

;=====
ClrScr         lfsr      2, LcdString
              movlw     .32
              movwf     CurPos
              movlw     ' '
ClrScrLoop    movwf     POSTINC2
              decfsz    CurPos, f
              bra       ClrScrLoop
              return

;=====
WriteChar      movwf     Char
              lfsrb     2, LcdString, CurPos
              movff     Char, INDF2
              incf      CurPos, f
              bcf       CurPos, 5                ; kurzorpoz körbejár
              return

```

```

;=====
WriteByte1;    WriteByte 1, '0', ' ', WriteChar
               return
;=====
WriteByte2     WriteByte 2, '0', ' ', WriteChar
               return
;=====
WriteByte3     WriteByte 3, ' ', ' ', WriteChar
               return
;=====
WriteShort1;   WriteByte 1, ' ', '-', WriteChar
               return
;=====
WriteShort2    WriteByte 2, ' ', '-', WriteChar
               return
;=====
WriteShort3    WriteByte 3, ' ', '-', WriteChar
               return
;=====
WriteWord3;    WriteWord 3, ' ', ' ', WriteChar
               return
;=====
WriteWord4     WriteWord 4, ' ', ' ', WriteChar
               return
;=====
WriteWord5     WriteWord 5, ' ', ' ', WriteChar
               return
;=====
WriteInt3;     WriteWord 3, ' ', '-', WriteChar
               return
;=====
WriteInt4      WriteWord 4, ' ', '-', WriteChar
               return
;=====
WriteInt5      WriteWord 5, ' ', '-', WriteChar
               return
;=====
; program területen található szöveget ír LcdString-be
; szöveg kezdete a programmemóriában : TBLPTR
; szöveg vége jelzés : #0 karakter
WriteTableStr  lfsrb 2, LcdString, CurPos
WTSckl        tblrd  *+
               movf  TABLAT, W
               btfsc STATUS, Z
               return
               incf  CurPos, f
               movwf POSTINC2
               bra   WTSckl

;=====
; egy sornyi szóköz kiírása
ClearLine     movlw  .16
               movwf WTemp
               movlw  ' '
               call  WriteChar
               decfsz WTemp, f
               bra   ClearLine + 4
               return

;=====
END

```

Macro.inc

```

=====
; programmemória mutató feltöltése konstans értékkel
; paraméterek :
; - Addr : adatmemória címe (konstans)
ltblptr macro Addr
    movlw    LOW Addr
    movwf    TBLPTRL
    movlw    HIGH Addr
    movwf    TBLPTRH
endm

=====
; programmemória mutató mentése egy 16 bites változóba
; paraméterek :
; - tw : 16 bites változó, ahová a mentés történjen
tblptrs macro tw
    movff    TBLPTRL, tw
    movff    TBLPTRH, tw + 1
endm

=====
; programmemória mutató betöltése egy 16 bites változóból
; paraméterek :
; - tw : 16 bites változó, ahol a betöltendő mutató található
tblptrl macro tw
    movff    tw, TBLPTRL
    movff    tw + 1, TBLPTRH
endm

=====
; adatmemória mutató beállítása egy bájtokat tartalmazó tömb, index-edik elemére
; paraméterek :
; - FsrNum = {0, 1, 2} : melyik adatmutatót állítsa be
; - ArrayStart : tömb kezdőcíme (konstans)
; - index : 8 bites változó, amelyik az indexet tartalmazza
; megj: a tömbnek egy lapon belül kell lennie
lfsrb macro FsrNum, ArrayStart, index
if FsrNum == 0
    lfsr     0, ArrayStart
    movf     index, W
    addwf    FSR0L, f
endif
if FsrNum == 1
    lfsr     1, ArrayStart
    movf     index, W
    addwf    FSR1L, f
endif
if FsrNum == 2
    lfsr     2, ArrayStart
    movf     index, W
    addwf    FSR2L, f
endif
endm

```

```

;=====
; adatmemória mutató beállítása egy szavakat tartalmazó tömb, index-edik elemére
; paraméterek :
; - FszNum = {0, 1, 2} : melyik adatmutatót állítsa be
; - ArrayStart : tömb kezdőcíme (konstans)
; - index : 8 bites változó, amelyik az indexet tartalmazza
; megj: a tömbnek egy lapon belül kell lennie
lfsrw macro FsrNum, ArrayStart, index
if FsrNum == 0
    lfsr    0, ArrayStart
    bcf     STATUS, C
    rlcfc   index, W
    addwff  FSR0L, f
endif
if FsrNum == 1
    lfsr    1, ArrayStart
    bcf     STATUS, C
    rlcfc   index, W
    addwff  FSR1L, f
endif
if FsrNum == 2
    lfsr    2, ArrayStart
    bcf     STATUS, C
    rlcfc   index, W
    addwff  FSR2L, f
endif
endm

;=====
; adatmemória mutató mentése egy 16 bites változóba
; paraméterek :
; - FszNum = {0, 1, 2} : melyik adatmutatót állítsa be
; - fw : 16 bites változó, ahová a mentés történjen
fsrs macro FsrNum, fw
if FsrNum == 0
    movff   FSR0L, fw
    movff   FSR0H, fw + 1
endif
if FsrNum == 1
    movff   FSR1L, fw
    movff   FSR1H, fw + 1
endif
if FsrNum == 2
    movff   FSR2L, fw
    movff   FSR2H, fw + 1
endif
endm

;=====
; adatmemória mutató betöltése egy 16 bites változóból
; paraméterek :
; - FszNum = {0, 1, 2} : melyik adatmutatót állítsa be
; - fw : 16 bites változó, ahol a betöltendő mutató található
fsrl macro FsrNum, fw
if FsrNum == 0
    movff   fw, FSR0L
    movff   fw + 1, FSR0H
endif
if FsrNum == 1
    movff   fw, FSR1L
    movff   fw + 1, FSR1H
endif
if FsrNum == 2
    movff   fw, FSR2L
    movff   fw + 1, FSR2H
endif
endm

```

```

;=====
; W-ben található 8 bites bináris számot karakteressé alakítja
; képes előjel nélküli és előjeles számok átalakítására is
; ha a szám nem fér el 'he' karakterhelyen, az elejét csonkolja
; a makro hivatkozás helyén szükséges változók:
; - Num : átmeneti tár a szám tárolásához
; - WTemp : átmeneti tár W regiszter tárolásához
; - ZeroSw : egy bit tárolóhely a kezdő nullák eltüntetéséhez (csak ha ZeroChar != '0')
; - NegSw : egy bit tárolóhely az előjel tárolásához (csak ha NegChar == '-')
; paraméterek :
; - he = {1, 2, 3} : ennyi számjeggyel írja ki
; - ZeroChar : szám elején előforduló nullákat mivel helyettesítse
; - NegChar : ha '-' akkor előjeles számként írja ki
; - WriteProc : karakterkiíró eljárás címe

WriteByte macro he, ZeroChar, NegChar, WriteProc
LOCAL l100loop, l100end, l10loop, l10end
LOCAL n100, n10, n1

                movwf    Num
; előjeles szám esetén
if NegChar == '-'
LOCAL Plus
                bcf      NegSw                ; pozitív szám
                btfss   Num, 7                ; pozitív ?
                bra     Plus                  ; igen
                negf    Num
                bsf     NegSw                ; negatív szám

Plus
endif

if ZeroChar != '0'
                bcf      ZeroSw
endif

if NegChar == '-' & ZeroChar == '0'
                movlw   NegChar
                btfss   NegSw
                movlw   ' '
                rcall   WriteProc
endif

; százaskok
if he >= 3
                movlw   '0'
                movwf   WTemp
endif
l100loop
                movlw   .100
                subwf   Num, f
                bnc     l100end
if he >= 3
                incf    WTemp, f
                if ZeroChar != '0'
                    if NegChar == '-'
                        btfsc   ZeroSw
                        bra     n100
                        movlw   ' '
                        btfsc   NegSw
                        movlw   NegChar
                        rcall   WriteProc
                    n100
                endif
                bsf      ZeroSw
            endif
        endif
                bra     l100loop

l100end
                movlw   .100
                addwf   Num, f

if he >= 3
    if ZeroChar != '0'
        movf    WTemp, W
        btfss   ZeroSw
        movlw   ZeroChar
        rcall   WriteProc
    else

```

```

        movf    WTemp, W
        rcall   WriteProc
    endif
endif

; tizesek
if he >= 2
        movlw   '0'
        movwf   WTemp
endif
l10loop    movlw   .10
           subwf   Num, f
           bnc     l10end
if he >= 2
        incf     WTemp, f
        if ZeroChar != '0'
            if NegChar == '-'
                btfsc ZeroSw
                bra    n10
                movlw   ' '
                btfsc NegSw
                movlw   NegChar
                rcall   WriteProc
            n10
        endif
        bsf      ZeroSw
    endif
endif
        bra     l10loop
l10end    movlw   .10
        addwf   Num, f
if he >= 1
    if ZeroChar != '0'
        movf    WTemp, W
        btfss   ZeroSw
        movlw   ZeroChar
        rcall   WriteProc
    else
        movf    WTemp, W
        rcall   WriteProc
    endif
endif
if NegChar == '-'
        btfsc   ZeroSw
        bra     n1
        movlw   ' '
        btfsc   NegSw
        movlw   NegChar
        rcall   WriteProc
n1
endif
; egyesek
        movlw   '0'
        addwf   Num, W
        rcall   WriteProc
endm

```

```

;=====
; Num változóban levő 16 bites bináris számot karakteressé alakítja
; képes előjel nélküli és előjeles számok átalakítására is
; ha a szám nem fér el 'he' karakterhelyen, az elejét csonkolja
; a makro hivatkozás helyén szükséges változók:
; - Num : változó a 16 bites szám tárolásához, az átalakítandó számot is ide kell tenni
; - WTemp : átmeneti tár W regiszter tárolásához
; - ZeroSw : egy bit tárolóhely a kezdő nullák eltüntetéséhez (csak ha ZeroChar != '0')
; - NegSw : egy bit tárolóhely az előjel tárolásához (csak ha NegChar == '-')
; paraméterek :
; - he = {3, 4, 5} : ennyi számjeggyel írja ki
; - ZeroChar : szám elején előforduló nullákat mivel helyettesítse
; - NegChar : ha '-' akkor előjeles számként írja ki
; - WriteProc : karakterkiíró eljárás

WriteWord macro he, ZeroChar, NegChar, WriteProc
LOCAL l10000loop, l10000end, l1000loop, l1000end, l100loop, l100end, l10loop, l10end
LOCAL n10000, n1000, n100, n10, n1

; előjeles szám esetén
if NegChar == '-'
LOCAL Plus
        bcf      NegSw                ; pozitív szám
        btfss    Num + 1, 7           ; pozitív ?
        bra      Plus                ; igen

        comf     Num + 1, f
        negf     Num
        btfsc    STATUS, C
        incf     Num + 1, f
        bsf      NegSw                ; negatív szám

Plus
endif

if ZeroChar != '0'
        bcf      ZeroSw
endif

; ha előjeles és kezdőnullák is vannak, akkor előjel kiírása
if NegChar == '-' & ZeroChar == '0'
        movlw    NegChar                ; negatív számnál ezt
        btfss    NegSw
        movlw    ' '                    ; pozitív számnál ezt
        rcall    WriteProc
endif

; tízezresek
if he >= 5
        movlw    '0'
        movwf    WTemp
endif
l10000loop    movlw    LOW .10000
               subwf    Num, f
               movlw    HIGH .10000
               subwfb   Num + 1, f
               bnc      l10000end
if he >= 5
        incf     WTemp, f
        if ZeroChar != '0'
                if NegChar == '-'
                        btfsc    ZeroSw
                        bra      n10000
                        movlw    ' '
                        btfsc    NegSw
                        movlw    NegChar
                        rcall    WriteProc
                n10000
                endif
                bsf      ZeroSw
        endif
endif
        bra      l10000loop

l10000end    movlw    LOW .10000
             addwf    Num, f
             movlw    HIGH .10000
             addwfc   Num + 1, f

```

```

if he >= 5
    if ZeroChar != '0'
        movf    WTemp, W
        btfss   ZeroSw
        movlw   ZeroChar
        rcall   WriteProc

    else
        movf    WTemp, W
        rcall   WriteProc
    endif
endif

; ezresek
if he >= 4
    movlw      '0'
    movwf      WTemp

endif
l1000loop
    movlw      LOW .1000
    subwf      Num, f
    movlw      HIGH .1000
    subwfb     Num + 1, f
    bnc        l1000end

if he >= 4
    incf       WTemp, f
    if ZeroChar != '0'
        if NegChar == '-'
            btfsc   ZeroSw
            bra     n1000
            movlw   ' '
            btfsc   NegSw
            movlw   NegChar
            rcall   WriteProc

        n1000
        endif
    endif
    bsf         ZeroSw

endif
endif
    bra        l1000loop

l1000end
    movlw      LOW .1000
    addwf      Num, f
    movlw      HIGH .1000
    addwfc     Num + 1, f

if he >= 4
    if ZeroChar != '0'
        movf    WTemp, W
        btfss   ZeroSw
        movlw   ZeroChar
        rcall   WriteProc

    else
        movf    WTemp, W
        rcall   WriteProc
    endif
endif

; százaskok
if he >= 3
    movlw      '0'
    movwf      WTemp

endif
l100loop
    movlw      LOW .100
    subwf      Num, f
    movlw      HIGH .100
    subwfb     Num + 1, f
    bnc        l100end

if he >= 3
    incf       WTemp, f
    if ZeroChar != '0'
        if NegChar == '-'
            btfsc   ZeroSw
            bra     n100
            movlw   ' '
            btfsc   NegSw
            movlw   NegChar
            rcall   WriteProc

        n100
        endif
    endif

```

```

        n100
    endif
        bsf        ZeroSw
    endif
endif
        bra        l100loop

l100end        movlw    LOW .100
                addwf    Num, f
                movlw    HIGH .100
                addwfc    Num + 1, f

if he >= 3
    if ZeroChar != '0'
        movf        WTemp, W
        btfss       ZeroSw
        movlw       ZeroChar
        rcall       WriteProc
    else
        movf        WTemp, W
        rcall       WriteProc
    endif
endif

; tizesek
if he >= 2
        movlw       '0'
        movwf       WTemp
endif
l10loop        movlw       .10
                subwf     Num, f
                bnc       l10end
if he >= 2
        incf        WTemp, f
        if ZeroChar != '0'
            if NegChar == '-'
                btfsc   ZeroSw
                bra     n10
                movlw   ' '
                btfsc   NegSw
                movlw   NegChar
                rcall   WriteProc
            n10
        endif
        bsf        ZeroSw
    endif
endif
        bra        l10loop

l10end        movlw       .10
                addwf     Num, f

if he >= 1
    if ZeroChar != '0'
        movf        WTemp, W
        btfss       ZeroSw
        movlw       ZeroChar
        rcall       WriteProc
    else
        movf        WTemp, W
        rcall       WriteProc
    endif
endif

if NegChar == '-'
        btfsc       ZeroSw
        bra         n1
        movlw       ' '
        btfsc       NegSw
        movlw       NegChar
        rcall       WriteProc
n1
endif
        movlw       '0'
        addwf       Num, W
        rcall       WriteProc
endm

```