

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Híradástechnikai Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

Univerzális Biquad szűrő megvalósítása saját tervezésű panelon

Szerző: Varga Kristóf
Nappali IV. év BSc
Konzulens: Csanádi Bertalan
Műszaki Tanár

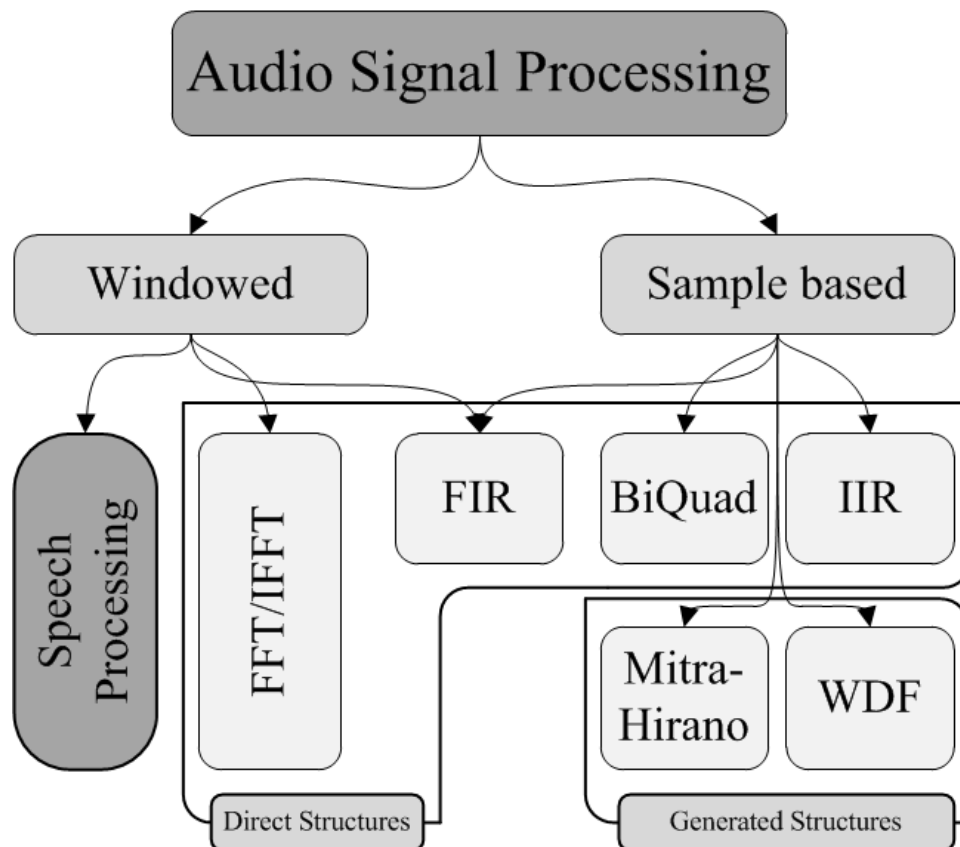
Budapest, 2012.

1, Mi is tartozik a hangjel-feldolgozáshoz

Hangjel-feldolgozással életünk során igen elterjedt területeken találkozhatunk. Mindennapjaink része lett, gyakran észrevétlenül „elsétálunk mellette”. Rádiózásnál, az oceanográfiában, hangok kódolásánál és dekódolásánál, beszéd analízis és szintézisnél, a hétköznapiakban egyre inkább terjedő Voice Over IP (VOIP) alapú kommunikációnál (például Skype), a zeneiparban, valamint pusztán zenehallgatás közben mind találkozunk hangjel-feldolgozással.

1.1 Hangjel-feldolgozó struktúrák

A hangjel-feldolgozó eljárásokat két nagy csoportba oszthatjuk, a mintákat egyesével (sample based) feldolgozó és jeleket csomagokban, úgynevezett ablakokban (windowed) feldolgozó eljárásokra. A mintákat feldolgozó eljárások bemenetén a mintavételezett jelből egymás után várjuk a mintákat, ha úgy tetszik, csak egy bemenete van. Míg a jeleket csomagokban feldolgozó eljárások a minták egy egész csoportját várják, tekinthetünk rájuk úgy, hogy több bemenetünk van, ezekre küldjük a mintáinkat a megfelelő sorrendben, de mindezt egyidejűleg.

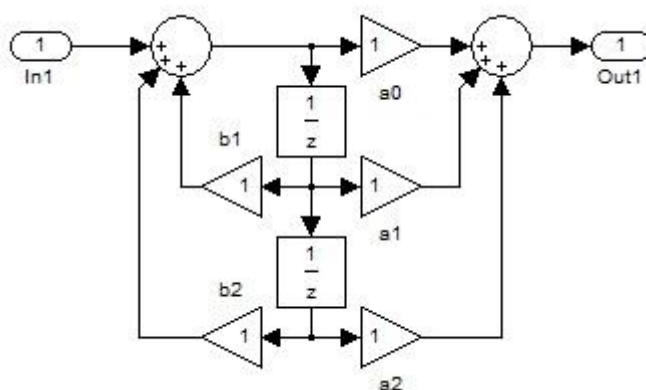


1.1 Ábra. Hangjel-feldolgozó struktúrák néhány példája és csoportosítása

A mintánként feldolgozó eljárások közé sorolhatjuk az IIR és biquad szűrőstruktúrákat, valamint a hullámdigitális szűrőket. A jeleket csomagokban feldolgozó eljárásokra legjobb példa a gyors Fourier transzformáció, illetve az ezen alapuló algoritmusok. Mindkettőbe be tudjuk sorolni a FIR szűrőket, hiszen egy közvetlen bemenetük van, amire magát a mintavételezett jelünket küldjük, magas fokszámon (ami általában szükséges a jel megfelelő minőségű feldolgozásához) viszont úgy is tekinthetünk mintha ablakozott függvényeket dolgozna fel. Ez például egy FIR szűrővel megalkotott zengetőnél igencsak szembevetendő, hiszen többszáz fokszám szükséges a megfelelő hosszúságú zengetés eléréséhez.

Másik csoportosítás szerint meg tudunk különböztetni direkt struktúrákat és indirekt, úgynevezett generált struktúrákat. Direkt struktúrának nevezzük a matematikai leképezés alapján létrejött struktúrákat, ilyen a FIR, IIR, Biquad struktúra és az FFT algoritmus is. A hullámdigitális szűrők nem tartoznak a direkt struktúrák közé, mivel ezek tervezése mindig egy analóg referenciaszűrő megalkotásával kezdődik, abból transzformáljuk a hullámdigitális szűrőnké. A Mitra-Hirano struktúra sem sorolható a direkt struktúrák közé, ez egy speciális struktúra, mely például fáziskorrekcióna is alkalmazható.

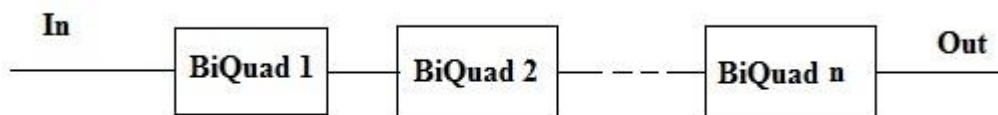
A szűrőstruktúrák közül a biquad struktúra az egyik legegyszerűbben alkalmazható. Tervezése nem olyan bonyolult, mint a hullámdigitális szűrőké, fókuszhatásfok aránya viszont jobb a FIR struktúráknál. A biquad struktúra egy direkt struktúrájú hálózat, ezért roppant egyszerű a megvalósítása illetve a visszafejtése, mivel az ágak szorzóegyütthatói megegyeznek az átviteli függvény együtthatóival. Egy általános másodfokú biquad struktúra átviteli függvénye az 1.1 képlet segítségével számítható.



1.2 Ábra: Általános biquad struktúra felépítése

$$H(z) = \frac{a_0(i) + a_1(i)Z^{-1} + a_2(i)Z^{-2}}{1 - b_1(i)Z^{-1} - b_2(i)Z^{-2}} \quad 1.1 \text{ képlet}$$

Ha a másodfokú struktúra átvitele nem lenne elegendő számunkra, a másodfokú szűrőinket kaszkádba köthetjük, így az átvitelük az 1.2 képlet alapján alakul. Így akár ugyanazt a szűrőt többször alkalmazva javíthatunk az átvitel erősítésén-csillapításán.



$$H(z) = \prod_{i=1}^n \frac{a_0(i) + a_1(i)Z^{-1} + a_2(i)Z^{-2}}{1 - b_1(i)Z^{-1} - b_2(i)Z^{-2}} \quad 1.2 \text{ képlet}$$

1.2 A Mason szabály

A fejlesztés során vizsgált hálózatok átvitelének vizsgálatakor az egyik leghasznosabb eljárás a Mason szabály alkalmazása volt. Ennek segítségével komolyabb számítások nélkül is megállapítható bármilyen hálózat átvitele. A számolások során nem kell transzformálásokat alkalmazni, mindössze összeadások és szorzások segítségével ugyanolyan pontosan, mégis kevesebb idő alatt megállapítható a $H(s)$ és $H(z)$ függvény. A Mason szabály akkor hasznos igazán, ha a hálózat igen bonyolult, vagy sok hurkot tartalmaz. Ezeknek a kiszámítása ezzel a módszerrel sokkal egyszerűbb, mint más eljárásokkal.

Ha egy hálózatnak fel tudjuk rajzolni a jelfolyam diagramját, akkor könnyen meghatározhatjuk az átvitelét ezzel a módszerrel. Első lépésként előre mutató ágakat keresünk a rendszerünkben, majd meghatározzuk az ágak erősítését. Az ágak felírása után hurkokat keresünk a rendszerben, ezeket is felírjuk a hozzájuk tartozó erősítésekkel együtt. A következő feladatunk a $\Delta(z)$ és $\Delta_i(z)$ kiszámolása. A $\Delta(z)$ a következő egyenlet alapján számolható ki.

$$\Delta(z) = 1 - \sum C_i + \sum C_i C_j - \dots + \dots$$

1.2.1, Képlet

A Δ_i ugyanezen képlet segítségével számolható ki, azzal a különbséggel, hogy a hozzá tartozó előremutató ág kitörlése után megmaradt hálózatra alkalmazzuk. Az így megkapott eredményeket felhasználva megadhatjuk a hálózat átviteli függvényét az alábbi összefüggés alapján.

$$H(z) = \frac{\sum_{i=1}^p P_i(z) * \Delta_i(z)}{\Delta(z)} \quad 1.2.2, \text{Képlet}$$

Nézzük meg egy példán keresztül az alkalmazását. Vegyünk egy bonyolultabb struktúrát, aminek az átviteli függvénye nem határozható meg pusztán ránézésből. A hullámdigitális struktúrák egyike tökéletes erre a célra. Az 1.2.1 ábrán láthatjuk a kiszemelt hálózatot. Egy ötödfokú struktúrát nézzünk végig. A fontosabb pontokat célszerű betűkkel elnevezni mielőtt hozzákezdünk, ez nagyban segíti később átlátni a rendszert. Első lépésként vegyük fel az előremutató ágakat az 1.2.1 táblázatba.

Előremutató ágak jele	Előremutató ág útvonala	Előremutató ág erősítése
P1(z)	[abcjghi]	Z^{-1}
P2(z)	[abcjefghi]	$-a_1 * Z^{-1}$
P3(z)	[akemi]	1
P4(z)	[akeoqmi]	$-a_2$

1.2.1 Táblázat: A hullámdigitális szűrő előremutató ágai

$\Delta(z)$	$1-[(2+a1)*Z^{-2}+a2*Z^{-1}]+[Z^{-2}*Z^{-2}+Z^{-2}*(a2*Z^{-2})+(a1*Z^{-2})*(a2*Z^{-2})]$
$\Delta1(z)$	$1-[(2+a1)*Z^{-2}]+[Z^{-2}*Z^{-2}+Z^{-2}*(a2*Z^{-2})]$
$\Delta2(z)$	$1-[Z^{-2}+a2*Z^{-2}]$
$\Delta3(z)$	$1-[(1+a1+a2)*Z^{-2}]+[Z^{-2}*(a2*Z^{-2})+(a1*Z^{-2})*(a2*Z^{-2})]$
$\Delta4(z)$	$1-[(1+a1)Z^{-2}]$

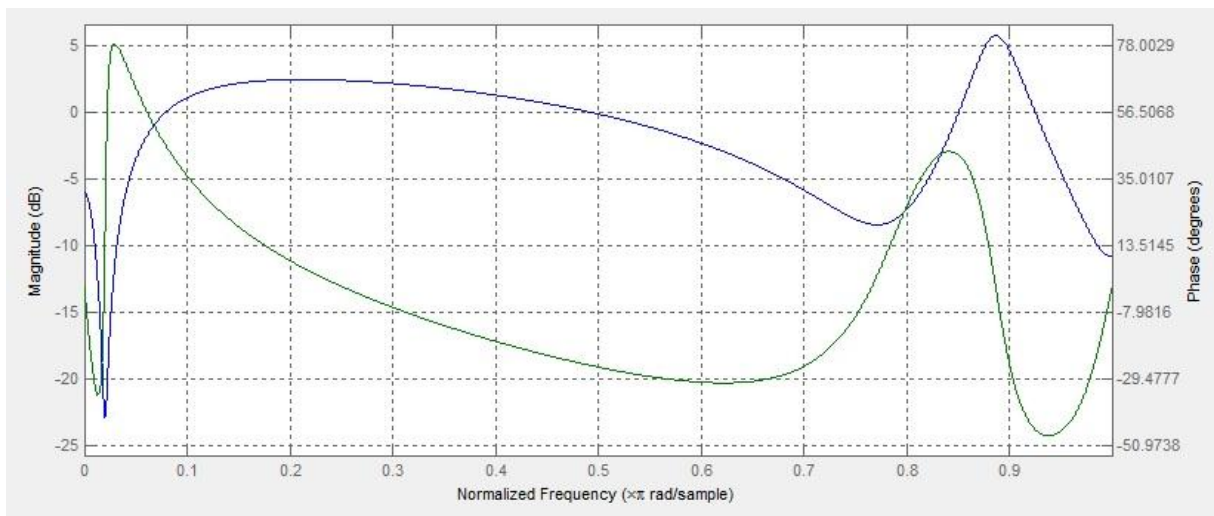
1.2.3. Táblázat: A $\Delta(z)$ értékek

Miután mindent kiszámítottunk, már csak az 1.2.2. képlet segítségével fel kell vennünk a hálózat átviteli függvényét.

$$H(Z) = \frac{1 - a1 + (1 - a1)Z^{-1} + (a1a2 - 1 - a1)Z^{-2} + (a1 + a1a2 - 2 - a2)Z^{-3} + (a2 + a1a2)Z^{-4} + (1 + a2)Z^{-5}}{1 - a2Z^{-1} - (2 + a1)Z^{-2} + (1 + a2 + a1 + a1a2)Z^{-4}}$$

1.2.3 Képlet: A kiszámított átviteli függvény.

Miután kiszámítottuk az átviteli függvényt, a Matlab segítségével szimulálni tudjuk a hálózatot, így könnyedén megkapva a Bode diagram alakját.



1.2.2 Ábra: A hullámdigitális struktúra Bode diagramja

1.3 Software által meghatározott eszköz

A Software Defined Radio és az Universal Software Radio Peripheral elvét alkalmazom. Ezekben az eszközökben meg van valósítva a jelek fogadására alkalmas hálózat (ebben az esetben a rádiófrekvenciás jelek antennával való vétele), majd ebből a fogadott jeltől pusztán software-es feldolgozással nyerik ki a kívánt jelet. Tehát a nyers jelen elvégzik a sávszűrést (csatorna kiválasztása), a demodulálást és esetleg a megfelelő formátumba átalakítását. (hangfrekvenciás jel, vagy digitális adatok USB-n történő továbbításhoz) Hasonló elven én is minden műveletet, minden beavatkozást software-esen oldottam meg. Ennek előnyei, hogy nem szükséges a megfelelő áramkör kialakítása a panelen, valamint ami a legfontosabb számunkra, az eszközön csupán a program frissítésével tudunk alkalmazkodni a kívánt feladatokhoz, akár egy teljesen más funkciót töltve be.

2, Az eszköz tervezése

2.1 – Követelmények

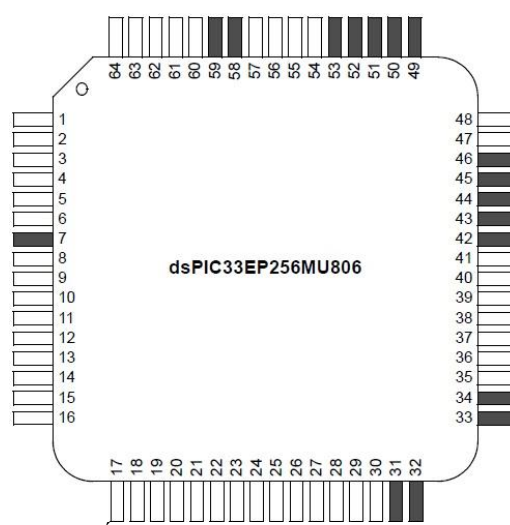
Célom egy olyan eszköz létrehozása volt, amely több audio csatornát képes kezelni egyidejűleg, lehetőséget adva sztereo ki- és bemenet megvalósítására, valamint mellette egyéb analóg jelek feldolgozására. Legyen elég erőforrás a beérkező adatmennyiség feldolgozására, hogy a megfelelő jelfeldolgozó feladatokat valós időben lehessen megvalósítani. Lehetőség legyen nagyobb rendszerbe, vagy hangláncba történő integrálásra és az ottani eszközökkel szinkron működésre. A hang megfelelő minőségű feldolgozásához szükség van legalább 44.1 kHz –es mintavételezési frekvenciára. A processzorban szükségünk van minimum 16 bites szóhosszúságra, a jelátalakító eszközben megfelelő szelektivitású, integrált átlapolódás szűrőre a bemeneten és helyreállító szűrőre a kimeneten. A processzor és a jelátalakító között szükség van szabványos kommunikációra. Kezelése legyen egyszerű, így bővítve a lehetséges felhasználók sorát.

2.2 – Kiválasztás

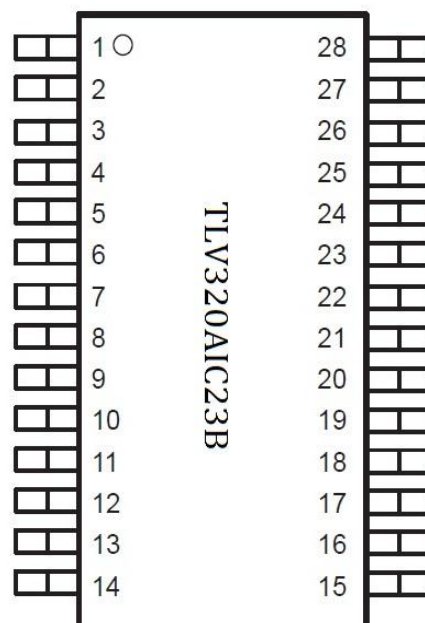
Egy új eszköz fejlesztésénél a legfontosabb munkafázis a megfelelő alkatrészek, illetve IC-k kiválasztása. A követelmények figyelembe vételével kell megtalálni azt az alkatrészt, amely kielégíti az igényeinket, egyúttal megtartva a költséghatékonyságot.

Az általam kiválasztott processzor a Microchip által gyártott dsPIC33EP256MU806. A processzor 24 analóg lábbal rendelkezik, így alkalmassá téve hangjelek feldolgozására. A processzor viszonylag kis tápfeszültségről, 3.3 V-ról működtethető. 85 °C-ig 70MIPS, 85 °C-tól 125 °C-ig 60 MIPS sebességre képes. Tartalmaz beépített dsp(Digital Signal Processing) engine-t, amely dsp jellegű utasítások végrehajtását teszi lehetővé. Előny univerzális eszköz tervezése közben ennél a szériánál a láb-multiplexálás lehetősége, amivel mi magunk rendeljük hozzá az adott lábhoz, a számunkra kívánt funkciót. Hátránya, hogy csak fixpontos aritmetikájú, ezt figyelembe kell venni a későbbiekben a dsp feladatok tervezésénél.

A jelátalakítás feladatára a Texas által gyártott TLV320AIC23B típusú kodeket szemeltem ki. Számomra ideális az eszköz által támogatott SPI, illetve I2C kommunikáció. Az általam kiválasztott kodek nagy előnye, hogy tartalmaz megfelelő szelektivitású beépített



2.2.1. Ábra: A processzor tokozása [8.]



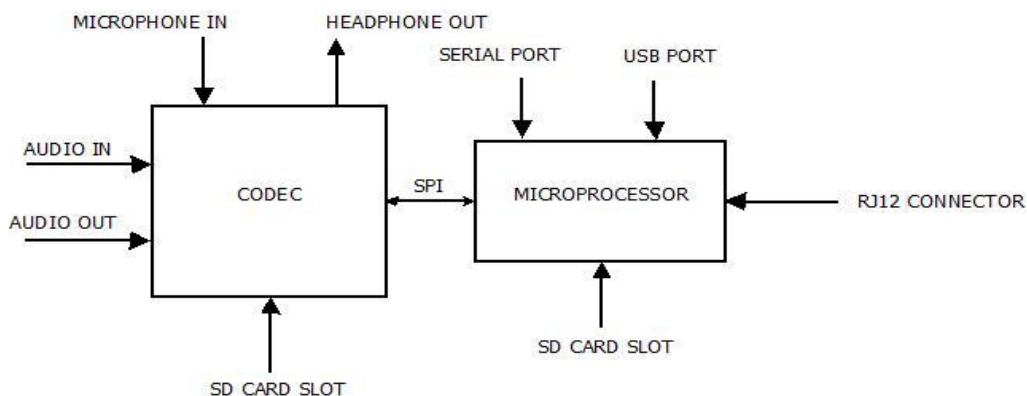
2.2.2. Ábra: A kodek tokozása [6.]

anti-aliasing szűrőt és helyreállító szűrőt a kimeneten, valamint beépített erősítőt a mikrofon bemeneten és a fülhallgató kimeneten. A tápellátásához használható 3.3V, amely megegyezik a processzor feszültségigényeivel, így egyszerűsíthetünk a tápellátásért felelős áramkörön. Az analóg-digitális átalakító 90 dB-es jel-zaj aránnyal rendelkezik, míg a digitális-analóg átalakító 100 dB-es jel-zaj viszony mellett üzemel. Mintavételezési sebesség szempontjából 8kHz-től a 96kHz-es tartományt támogatja. Analóg bypass funkciót is találhatunk az eszközben, így akár a kodekben összeköthetjük a be- és kimeneti csatornát. Fogyasztása is előnyös, hiszen Playback Mode használatkor 23 mW-ot vesz fel.

3, Az elkészült Audio Signal Processing Board

3.1 – A jelfeldolgozó eszköz tervezése

A hardware megvalósításánál az alap koncepció az volt, hogy három modulból álljon össze az eszköz, modularitás szerint választva el őket, így mindig csak azt kell használni, amire szükség van. Három panelt tartalmazott a kezdeti elképzelés, kettőt ajánlott használni, a harmadik csupán opcionális. A legfelső a mikroprocesszort, tápellátást és az USB kommunikációhoz szükséges áramköröket tartalmazza. A középső panelre került a kodek, az analóg illesztések, a world-clock csatlakozás, valamint a soros port áramköre. A legalsó panelre a középsőn keresztül egy molex csatlakozó segítségével csatlakozhatunk, ahol számunkra megfelelő analóg illesztés érhető el. A középső (kodek) panel nem csak ezzel a processzoros panellel használható, hanem csatlakoztathatók hozzá az Arduino család paneljei is. Ezzel a megoldással akár maradhatunk a már jól ismert rendszerünkönél.



3.1. Ábra. Az eszköz blokkvázlata

3.1.1 – A Mikroprocesszort tartalmazó panel

A panel lelkét a már előbb bemutatott dsPIC33EP256MU806 adja. A processzor működéséhez 3.3 V-os tápfeszültség szükséges, ezt a feszültséget külső 5 V-os táplálásról, vagy más eszközünkről USB csatlakozó segítségével tudjuk kinyerni. A kapott 5 V-ot a TS1117CW-3.3-as regulátor IC segítségével állítjuk elő. Az így előállított 3.3 V szolgál a továbbiakban az egész eszközünk táplálásához. Két ledet helyeztem el a regulátor mellett, amelyek jelzik a tápfeszültség állapotát. Az egyik led a várt 5 V meglétét jelzi, a másik led egy zener diódás kapcsolásban található, amely jelzi, ha a tápfeszültség meghaladja az 5 V-ot, míg a zener dióda az esetleges túlfeszültséget gátolja meg.

Erre a panelre került az USB csatlakozó, mely lehetővé teszi a kommunikációt számítógépünkkel. A processzor fullspeed USB kommunikációt támogat host és device módban. Ezen kívül lowspeed kommunikációra is képes, de ezt csak host módban. Az eszköz hardware-sen képes kezelni az USB kommunikáció handshake funkcióját, így ezt software-sen

nem kell megvalósítani, valamint támogatja mind a négy adatátviteli módot (controll, interrupt, bulkdata, isochronous).

Az RJ12-es csatlakozó is ezen a panelen található, ennek segítségével tudok csatlakozni például az ICD eszközhöz, mely a programozáshoz szükséges. Ez a megoldás kikerülhető egy bootloader használatával.

Erre a panelre került még 3 led felhasználói célra, 2 nyomógomb, valamint a reset gombja. A panel szélén helyeztem el a csatlakozósort, mellyel csatlakozunk a kodekes panelhez, illetve ennek segítségével illeszthetjük a processzoros panel helyére az Arduino eszközünket. Az alsó rétegen helyet kapott még egy SD kártya foglalat, melyet szabadon használhatunk, akár SPI kommunikációval memória bővítésére, akár a feldolgozott jelek valamely értékeinek tárolására.

3.1.2 - A Kodeket tartalmazó panel

A Kodek panel központi alkatrésze az analóg-digitális, illetve digitális-analóg átalakításokért felelős TLV320AIC23B kodek, melyet a Texas Instruments Inc. gyárt. A szükséges 3.3 V-os tápfeszültséget a már előbb bemutatott csatlakozósor szolgáltatja ennek a panelnek.

Az audio csatornák számára itt találhatók az analóg illesztések.

Ezen a panelen kapott helyet a soros port csatlakozója. A soros port használatához szükség volt a MAX3232-es IC-re ami az RS232 és TTL szintek közötti konvertálást valósítja meg. Utána a soros port jelét a csatlakozósoron keresztül a felső panelre vezetem, ahol a processzorban kerül feldolgozásra.

A panelen található még a world-clock fogadására a koaxiális csatlakozó, illetve a hozzá tartozó áramkör. A célra a 4N33-as optocsatolót használom, és a kapott jel interruptot generál a processzorunkban, így érve el a szinkron működést a rendszer többi eszközével.

Ennek a panelnek az alsó rétegére is került egy SD kártya foglalat, mely további adatterület bővítésre alkalmas.

3.2 – DSPIC Software

3.2.1 – Forrásállományok

A software-t a pic számára a Michrohip MPLab IDE környezete segítségével írtam és az ICD3 –as programozóval töltöttem fel az eszközre. A program több forrásállományból áll, a main függvényből hívom meg az ezekben megtalálható függvényeket. Így sokkal áttekinthetőbb lesz a programkód, saját magunk illetve mások számára is. A saját programkódunk kommentálása nagyon ajánlott, így később is pontosan emlékszünk a fontos gondolatokra.

A megírt forrásállományok közül a main.c tartalmazza a fő ciklust, innen hívom meg a többi funkció függvényeit, illetve itt történik az interrupt kezelés. A sysinit.c-ben találhatóak a hardware-hez szükséges inicializációk, beállítások. A sysinit.h-ban a sysinit.c függvényeinek a deklarációja található. Ezek a függvények külső függvényként realizálandók, hiszen több forrásállományon keresztül akarjuk meghívni őket. A user.c tartalmazza a felhasználó függvényeit, én itt a hangfeldolgozás számításait realizáltam. A user.h a user.c függvényeit tartalmazza. Az utolsó forrásállomány, az effects.s tartalmazza az IIR illetve egyéb struktúrájú szűrő kódját.

3.2.1.1 - A megfelelő beállításokat tartalmazó forrásállomány

A processzor beállításai miatt először a sysinit.c állományt mutatom be. A sysinit állományban hét függvény van megírva, ezek szolgálnak a processzor egyes funkcióinak a beállítására. Ezek az oszcillátor, a timer 1 és 2, a láb multiplexálás, a kodek, a DCI modul, valamint az UART modul beállításai. Első lépésként az oszcillátort kell elindítani, hisz e nélkül nem lenne órajele a rendszernek. Ehhez minden egyéb előtt szükséges megadni, hogy külső vagy belső forrásból vesszük-e az órajelet, használunk-e PLL-t, illetve egyéb beállítások is megtehetők. Miután megadtuk az órajel forrását, elkezdhetjük az oszcillátor beállítását. A

rendszer frekvenciáját a következő módon számolhatjuk ki. A rendszer figyelésére a watchdog timer nem szükséges, így ezt a működési funkciót kikapcsoljuk. Miután ezek a beállítások megtörténtek, meg kell várnunk, amíg a PLL beáll, utána kezdhetjük el a többi beállítást.

$$F_{osc} = F_{in} * \left(\frac{M}{N1}\right) = F_{in} * \left(\frac{(PLLDVI + 2)}{(PLLPRE + 2) * 2(PLLPOST + 1)}\right) \quad 3.1 \text{ képlet}$$

A Microchip újítása ezen a szérián a multiplexálható lábak tulajdonsága. Itt mi adhatjuk meg (bizonyos kitételekkel), hogy melyik lábhoz milyen funkciót rendelünk hozzá. Egy láb adott funkciójának pontos meghatározásához három dolgot szükséges beállítani. Meg kell határozni az adat irányát egy lábon, azaz, hogy kimenet vagy bemenet, hogy analóg vagy digitális, illetve magát a funkciót, hogy a processzor melyik moduljához csatlakozzon.

A Timerek közül az első kettőt használjuk timer üzemmódban. A timerekkel különböző események bekövetkeztét fogjuk figyelni.

A következő lépés a beállítások során, az SPI, majd utána a kodek beállítása. Azért van szükség előbb az SPI beállítására, mivel SPI kommunikáció zajlik a processzor és a kodek között, valamint így zajlik a beállítása is. Az input channel és output volume control regiszterekben a ki- és bemeneteket tudjuk engedélyezni, illetve a hangerőt állíthatjuk be. Az analog audio path regiszterben a mintavételezés helyét állítjuk be, hogy vonal bemenetről vagy a mikrofonról mintavételezzünk, illetve itt indítjuk el a digitális-analóg átalakítót. Ebben a regiszterben találhatunk még egy bypass funkciót, amivel a kimenetet és bemenetet kötjük össze, kizárva ezzel minden digitális vonalat a rendszerből, így plusz zaj nem kerül a jelre. A digital audio pathban a digitális-analóg átalakító software-es némitását engedélyezhetjük, valamint az ADC felüláteresztő szűrőjét tilthatjuk. A power down control regiszterben az eszköz egyes funkcióit kapcsolhatjuk ki az alacsonyabb fogyasztás érdekében. A digital audio interface format regiszterben a master-slave módokat és az adat formáját, az adatbitek hosszúságát és az adatátviteli módot választjuk ki. A sample rate controll regiszterben a mintavételi frekvenciát tudjuk beállítani. Én 48 kHz-es mintavételezéssel dolgozok.

Ezután a soros port beállítása következik. A soros portot 115200-as baud rate-el használom. Az átvitelben 8 adatbitet használok, paritásbit nélkül és két stopbittel működik. A beérkező adatokra beállíthatjuk, hogy egy, három, vagy négy beérkezett bit után generáljon interruptot. Ezt alapértelmezett esetben három bitre állítom.

A data converter interface felelős a kommunikáció lebonyolításáért a kodek és a mikroprocesszor között. Az órajelet a kommunikációhoz a kodek szolgáltatja, esetünkben ő a master a rendszerben. Ehhez a frameszinkronizáció lábat bemenetnek kell állítani, hiszen a mikroprocesszor fogadja az órajelet, nem ő szolgáltatja. Lehetőség van a data converter interface-n belül arra, hogy bypasst valósítsunk meg, ezt a beállítások alatt, illetve a tesztelések közben használtam ki. Az átvitel során egy keretben két szó kerül átküldésre, melyek 16 bitnyi adatot tartalmaznak.

3.2.1.2 – A fő ciklus

A main.c a fő forrásállomány, innen kerül meghívásra az összes függvény, valamint itt fut a program fő ciklusa. Az állomány a header file-ok beágyazásával kezdődik, majd a programozást segítő definíciók meghatározása után deklarálni kell a szükséges változókat. A main függvényben meghívom az éppen szükséges funkciók beállításainak a függvényeit. Az oszcillátor és a lábmultiplexálás beállításai mindig szükségesek. Ezek után számomra szükséges még a soros port, a timer egy és kettő, a data converter interface, valamint az SPI és a kodek beállításai. A függvények meghívása után a soros porton kiküldök egy tetszőleges, de általam ismert adatot, hogy lássam, a beállítások hiba nélkül megtörténtek, az eszköz készen áll a

használatra. Utána egy végtelen ciklust rakok be, hogy a program futása ne álljon le, a további műveleteket interrupt kezelésén belül fogom megoldani.

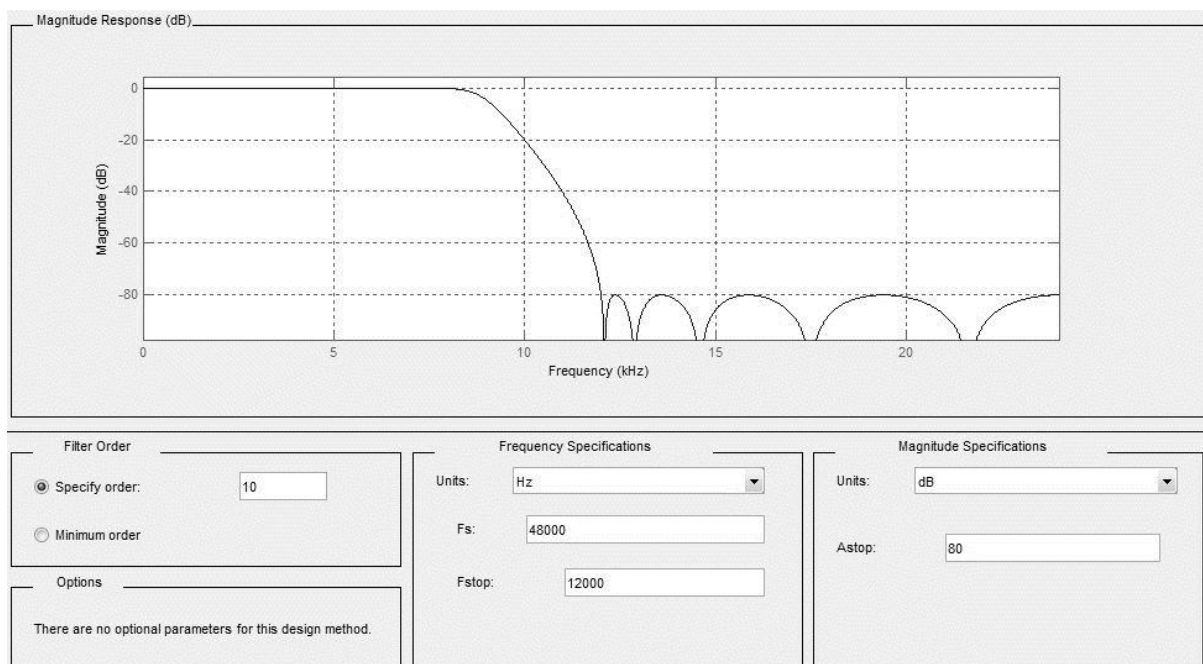
A timereket időzített események kezelésére használom. A timer eggyel a gombok állapotát figyelem, így nem kell a processzornak mindig figyelnie a gombokat, csak a timer által generált interruptokkor vizsgálom az állapotukat. A timer kettő a zero crossing-nál oltja ki a ledet a meghatározott időtartam után. A soros port beérkező oldali interruptjában történik a szűrőparaméterek átvétele és ezek kezelése. A data converter interface interruptjában hívom meg a szűrő függvényét, ha a megfelelő gomb lenyomásra került. Ellenkező esetben egy software-s bypassot valósítok meg, tehát a beérkezett jelet visszaküldöm a kimenetre. A függvénynek egy tömböt adok át, ami tartalmazza a szűrő szorzókonstansait.

3.1.2.3 - A felhasználó függvényei

A user.c-ben a felhasználó által megírt hangjelfeldolgozás történik. Itt történnek a szükséges előszámítások, illetve itt valósítok meg egy zero-crossing elven történő ritmusdetektálást. Ehhez a nullaátmeneteknél bekapcsolok egy ledet, majd egy rövid idő után (timer kettő) kioltom. A beérkezett adatokat és a fogadott szorzókonstansokat átadom egy Assembly függvénynek, melyben maga a Biquad szűrő van megvalósítva.

4. – Pc oldali software

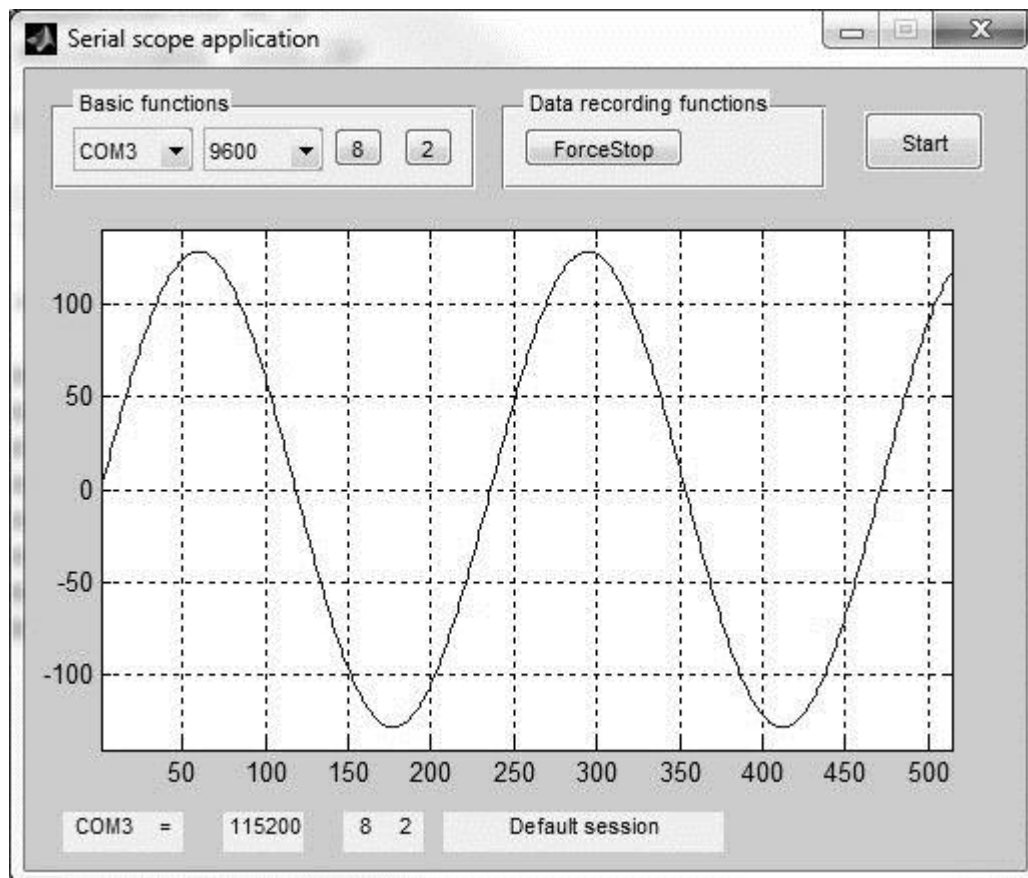
A számítógép oldalán is sok program van, amely segít a hangjelfeldolgozás megvalósításában. A legtöbb segítséget nyújtó program számomra a Matlab. Matlab segítségével tudom előre tervezni a szűrőt. A Filter Design and Analysis Tool (FDA tool) segítségével tudom szimulálni a kiszámolt, megtervezett szűrőket, így tudom, hogy milyen válaszokra várhatok. Ezt használhatjuk egyfajta ellenőrzésképpen, hogy lássuk jól működik-e az eszközön megvalósított szűrő.



4.1. Ábra: FDA tool – a tervezés menete

A quantizer object segítségével a kiszámolt szorzókonstansokat tudom átalakítani, a PIC által igényelt (Q 0.15) tört, fixpontos számábrázolásba. Nagy figyelmet kell fordítani tervezéskor, hogy ha ezen az eszközön akarjuk megvalósítani a szűrőt, akkor a szorzókonstansok nem lehetnek nagyobbak egynél, illetve kisebbek mínusz egynél, mivel ezt a fixpontos számábrázolásnál nem tudjuk ábrázolni. Ennek megfelelően a quantizer object az átalakítás során a megengedett határt átlépő értékeket levágja a megengedett maximumig.

Matlab-ban megvalósítható a soros port kezelése is, melynek számos előnye van. Elsőre nézzük az adatok fogadásának lehetőségét. Azáltal, hogy fogadjuk a soros porton küldött adatokat, például magukat a mintavételezett értékeket, a számítógépünkön is fel tudjuk ezeket dolgozni. Szélsőséges esetben maga a jelfeldolgozás is megtörténhet itt, majd a lejátszani kívánt jelet visszaküldhetjük az eszköznek, amire így már csak a digitál-analóg átalakítás marad. Látjuk, hogy ez az eszköz szempontjából nem túl gazdaságos üzemmód, mindenesetre a nagyobb számításokat igénylő feldolgozás ezzel a módszerrel megvalósítható a számítógépünk segítségével hívásával. Egy másik lehetséges alkalmazása a soros port kezelésének, hogy a kívánt digitális jelünket elküldjük soros porton, ez Matlab alatt megjeleníthető, így egy oszcilloszkópot valósítva meg a számítógépünkön.



4.2. Ábra: A Matlab alatt futó oszcilloszkóp működés közben

Végző megoldásként használhatjuk a Matlabot a szűrőtervezés automatizálására. Az FDA tool-ban megtervezzük a kívánt szűrőnk, majd az eredményként kapott szorzókonstansokat a megfelelő alakban a soros porton továbbítjuk az eszközünk felé. Így gyakorlatilag egy FDA tool-os tervezés alatt kiszámoljuk a paramétereket, átadjuk az eszközünknek és fogadás után már az új szűrő szerint dolgozzuk fel a fogadott jelet.

5. Összefoglalás

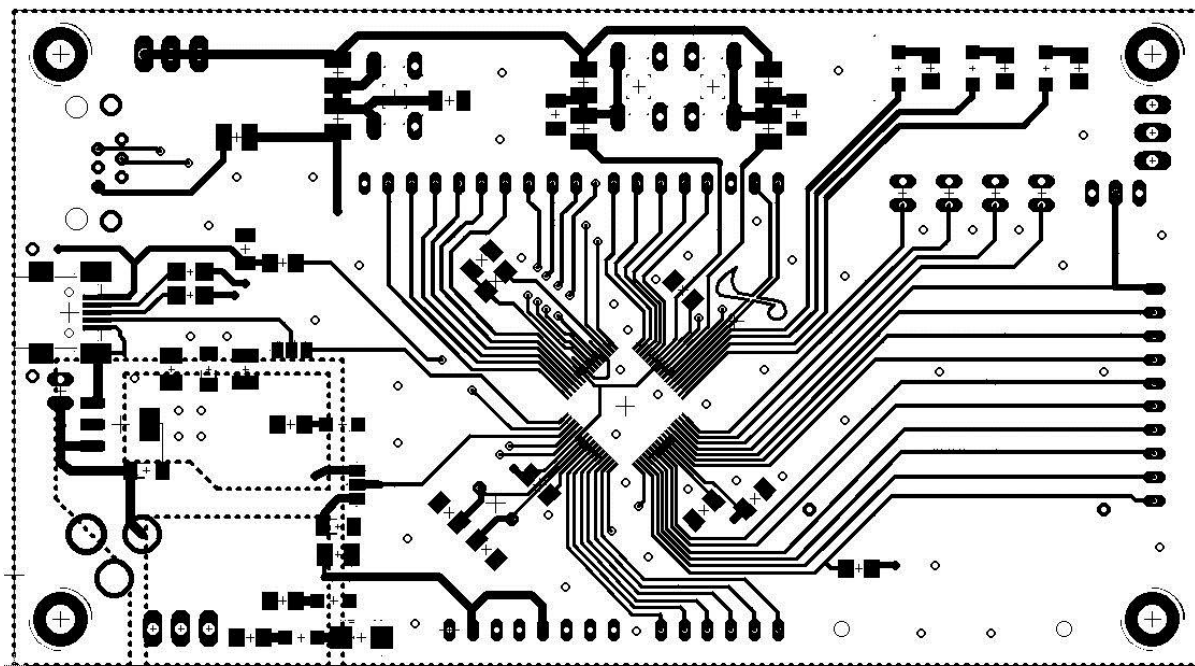
Végeredményben megvalósítottam egy olyan univerzális eszközt, amely a hangfeldolgozás területén software-esen valósít meg alkalmazásokat. A dsPIC33-as család processzorával 70 MIPS-es sebességet lehet kihasználni, így könnyedén megvalósíthatóak két csatornán (sztereó jel) hangjel-feldolgozó alkalmazások. Nagy sebességű USB kapcsolatának és a soros portnak köszönhetően külső eszközökkel is tudunk kommunikálni. Az SD kártya foglalatokkal memóriabővítést lehet elérni, vagy akár külső file-okat lehet feldolgozásra, illetve lejátszásra csatlakoztatni. A world clock csatlakozási lehetőséggel egy nagyobb rendszerbe vagy hangláncba is beépíthető, valamint ezekkel képes szinkron futásra.

A Mason szabály alkalmazásával könnyedén számolható egy megvalósítandó rendszer átvitele, majd a megfelelő struktúra kiválasztásával meg is valósíthatjuk azt. A számítógéppel történő összekapcsolás esetén át lehet adni a feldolgozási feladatok egy részét az eszköznek. A megfelelő kezeléssel elérhető, hogy futás közben alakítsuk át a szűrő átvitelét, akár a struktúra fokszáma is felülírható. Megvalósítottam tehát egy olyan hangjel-feldolgozó rendszert, mely rugalmasan átalakítható az igényeknek megfelelően, valamint futás közbeni átalakítással megspórolható az eszköz cseréjével járó kellemetlenségek.

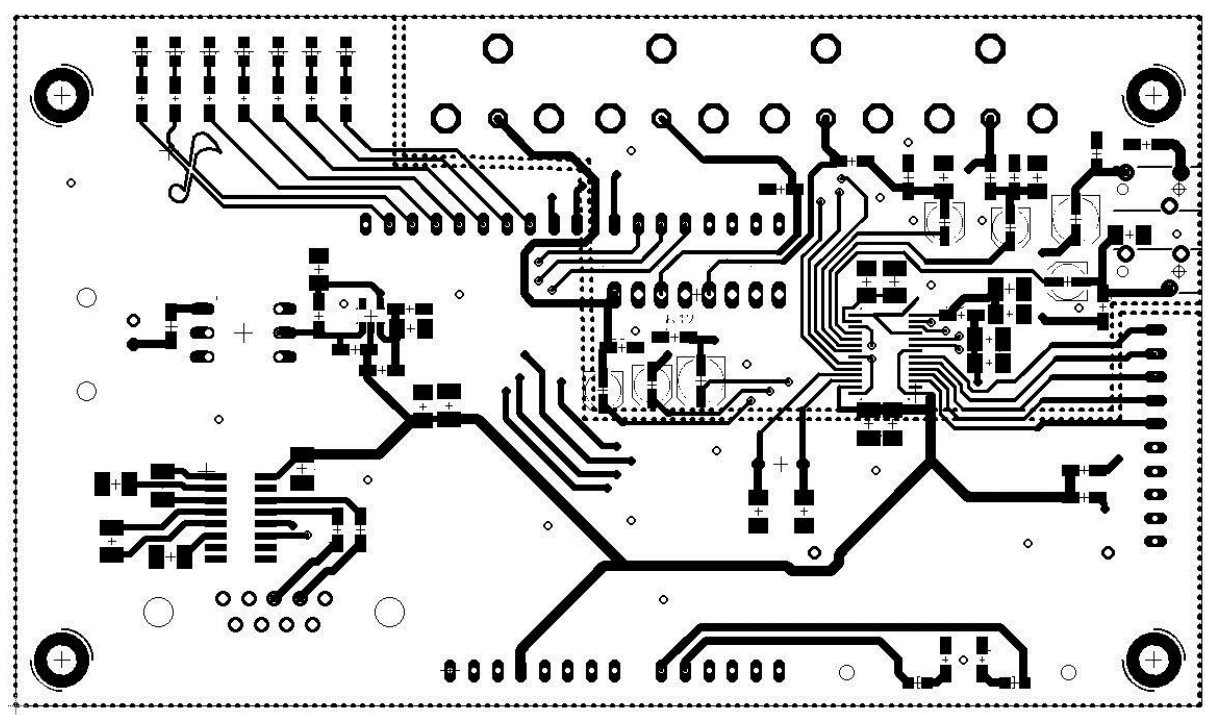
8. Felhasznált Irodalom:

- [1.] Gyányi Sándor, Wühl Tibor, *Digitális jelfeldolgozó hálózatok gyakorlati megvalósítása*, 2009, BMF-KVK 2072, Budapest
- [2.] F. Shipley, F. Arnold, M. Tsecouros, R. King (2002, february) *Evaluation Platform for the TLV320AIC23 Stereo Audio CODEC and TLV320DAC23 Stereo DAC*. Available: <http://www.ti.com/lit/ug/sleu016/sleu016.pdf>
- [3.] Horváth Elek, *Méréstechnika változatlan utánnyomás* 2010., Budapest
- [4.] *Eagle users manual* 2010., Available: <http://www.cadsoftusa.com/>
- [5.] *MAX3232 datasheet*, Texas Instruments Inc (2000 January) Available: <http://www.ti.com/lit/ds/symlink/max3232.pdf>
- [6.] *TLV230AIC23B datasheet*, Texas Instruments Inc. (2004) Available: <http://www.ti.com/lit/ds/symlink/tlv320aic23b.pdf>
- [7.] *SD Socket datasheet*, Attend Technology Inc., Available: http://www.hestore.hu/files/mcc-sd_2.pdf
- [8.] *DsPIC33EPXXX (GP/MC/MU)806/810/814 Datasheet*, Microchip Technology Inc., Available: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en554301> (2012. 03. 21.)
- [9.] *MPLAB ICD In-Circuit Debugger*, Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en537580&redirects=icd3
- [10.] R. Lyons, *Using Mason's Rule to Analyze DSP Networks*, Available: <http://www.dsprelated.com/showarticle/76.php>
- [11.] Tae Hong Park, *Introduction to Digital Signal processing Computer Musically Speaking*, 2010, Singapore
- [12.] J. Goette, *An IIR-Filter Example: A Butterworth Filter*, 2012, Biel
- [13.] Mitra, Sanjit K., *Digital Signal Processing: A Computer Based Approach, Second Edition*, McGraw Hill, 2001.
- [14.] Julius O. Smith *Physical Audio Signal Processing: For Virtual Musical Instruments and Audio Effects* August 2007 Edition, 2007
- [15.] D. Schlichthärle, *Digital Filters*, Basics and Design second edition, 2011
- [16.] Dr. Simán István, *Digitális Jelfeldolgozás*, 2008, Budapest
- [17.] Dr. Simonyi Ernő, *Digitális Szűrők*, 1984, Budapest
- [18.] Microchip Technology Inc., *Filter Design for dsPICTM DSC Digital Filter Design and Analysis System*, 2008, Available: <http://ww1.microchip.com/downloads/en/devicedoc/sw300001%20manual.pdf>
- [19.] J. O. Pliam, E. Bruce Lee, *On the Global Properties of Interconnected Systems*, 1995

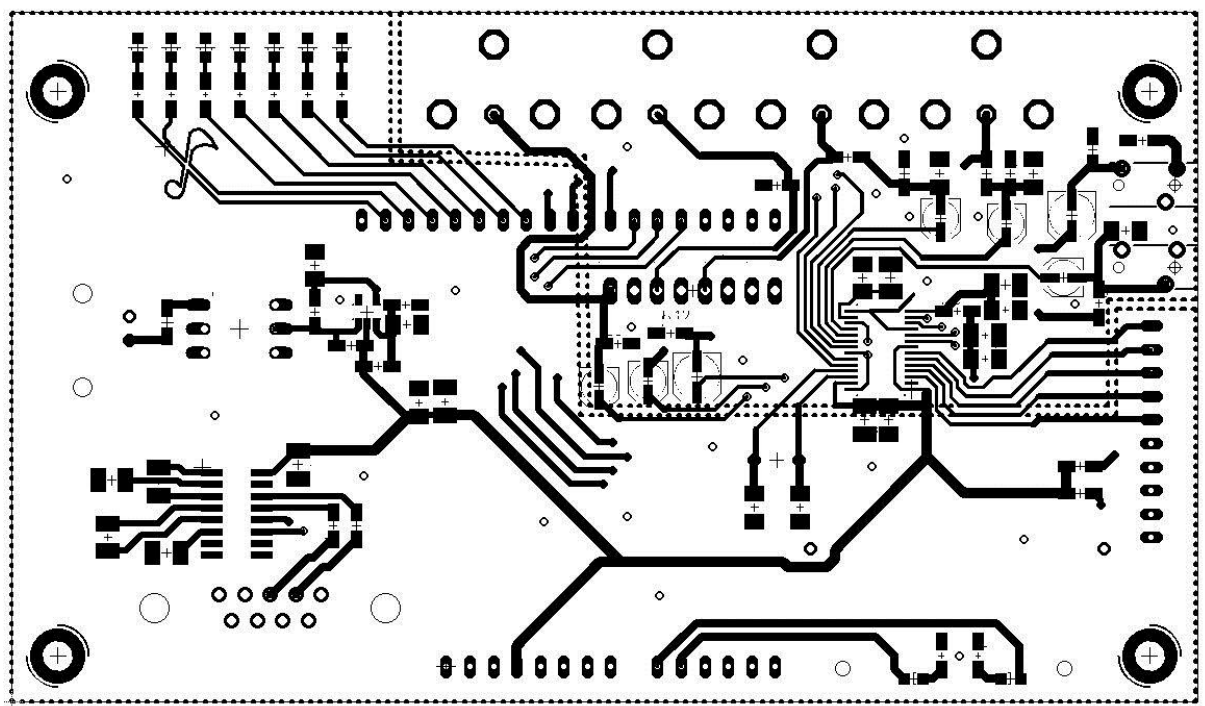
Mellékletek



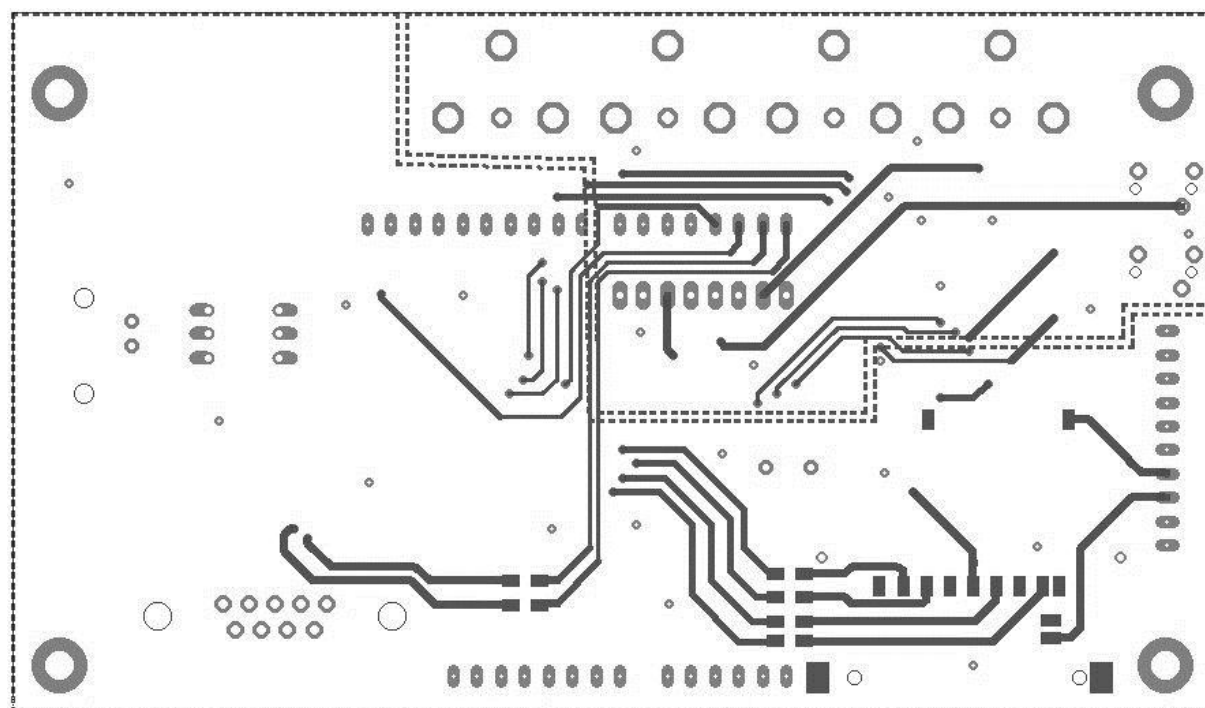
1. ábra. A mikroprocesszort tartalmazó panel felső rétege



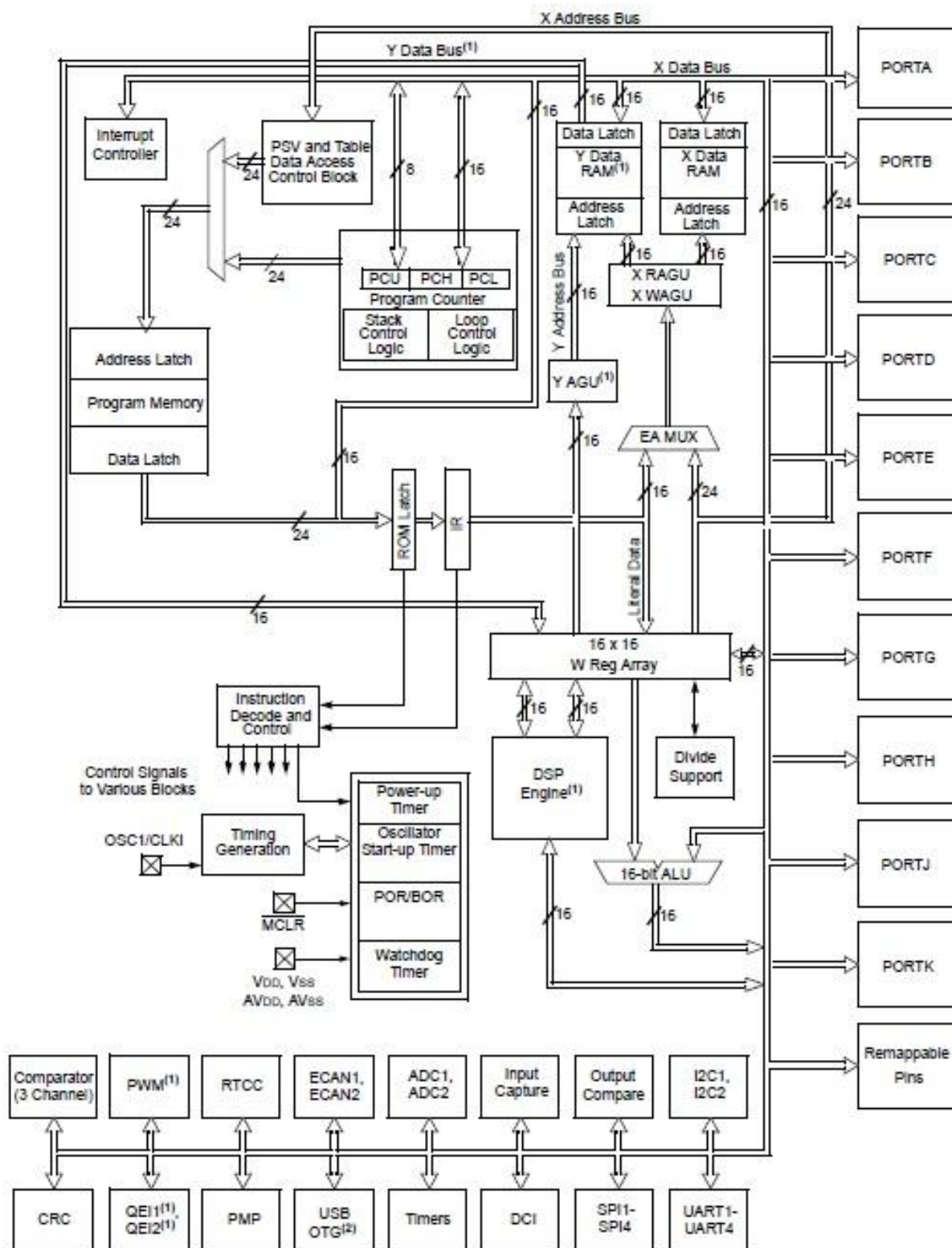
2. ábra. A mikroprocesszort tartalmazó panel alsó rétege



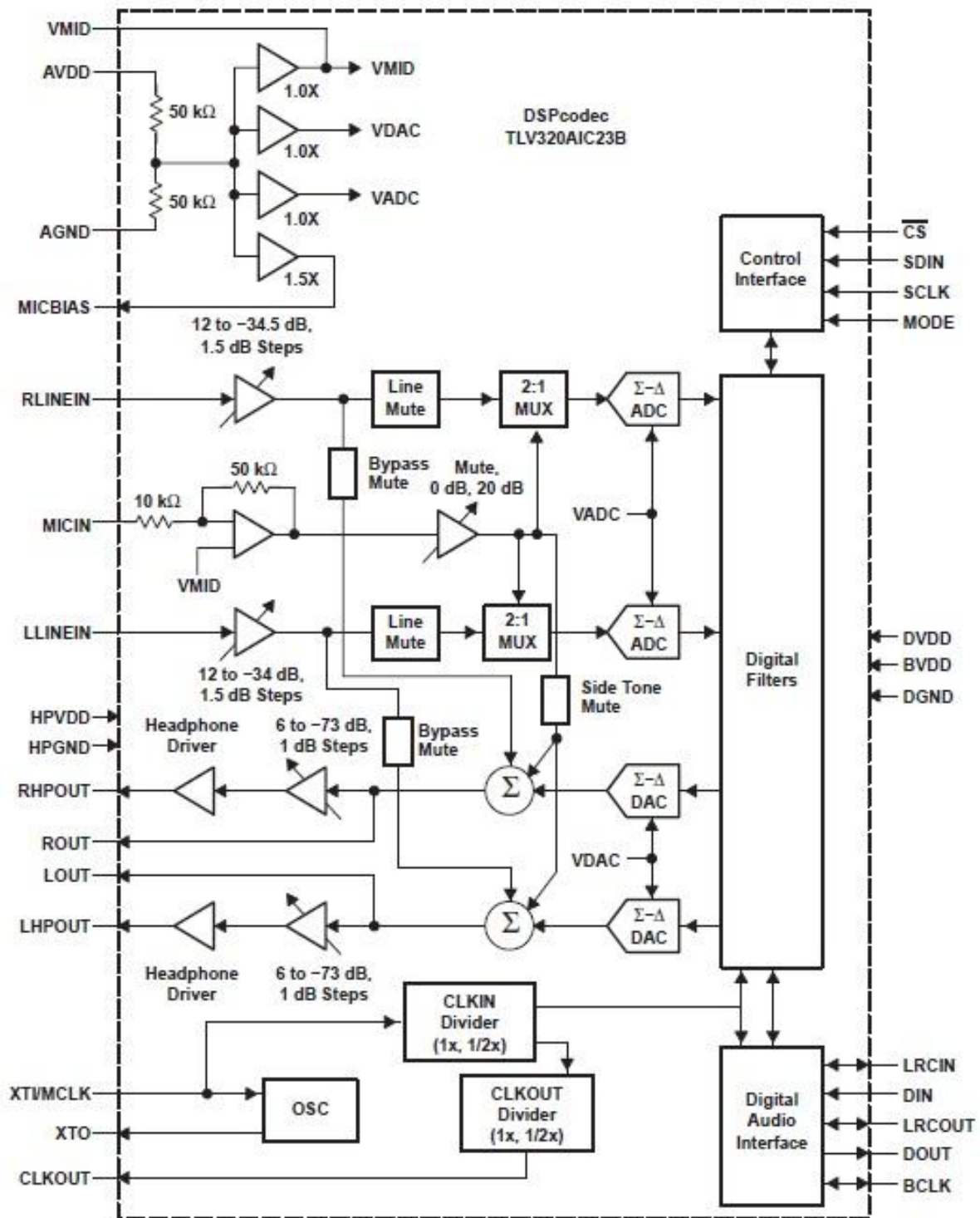
3. ábra. A Kodeket tartalmazó panel felső rétege



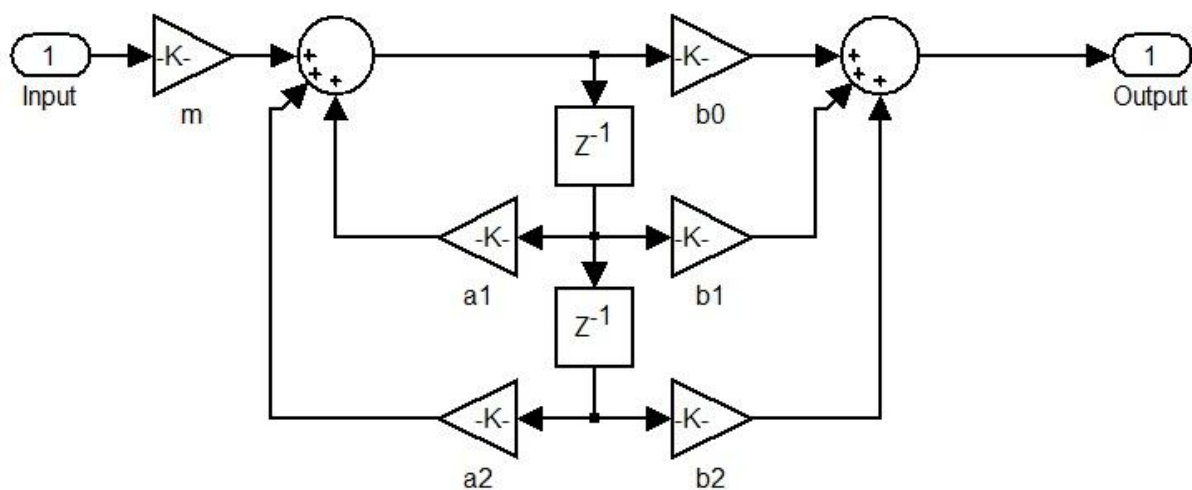
4. ábra. A Kodeket tartalmazó panel alsó rétege



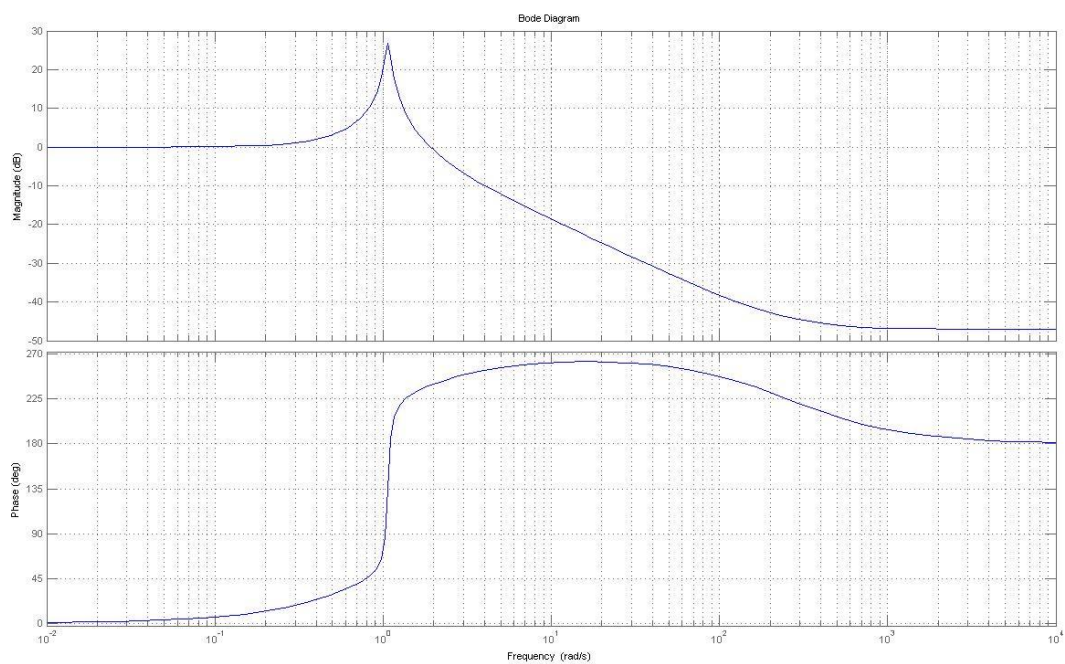
5. Ábra: A processzor funkcionális blokkvázlata



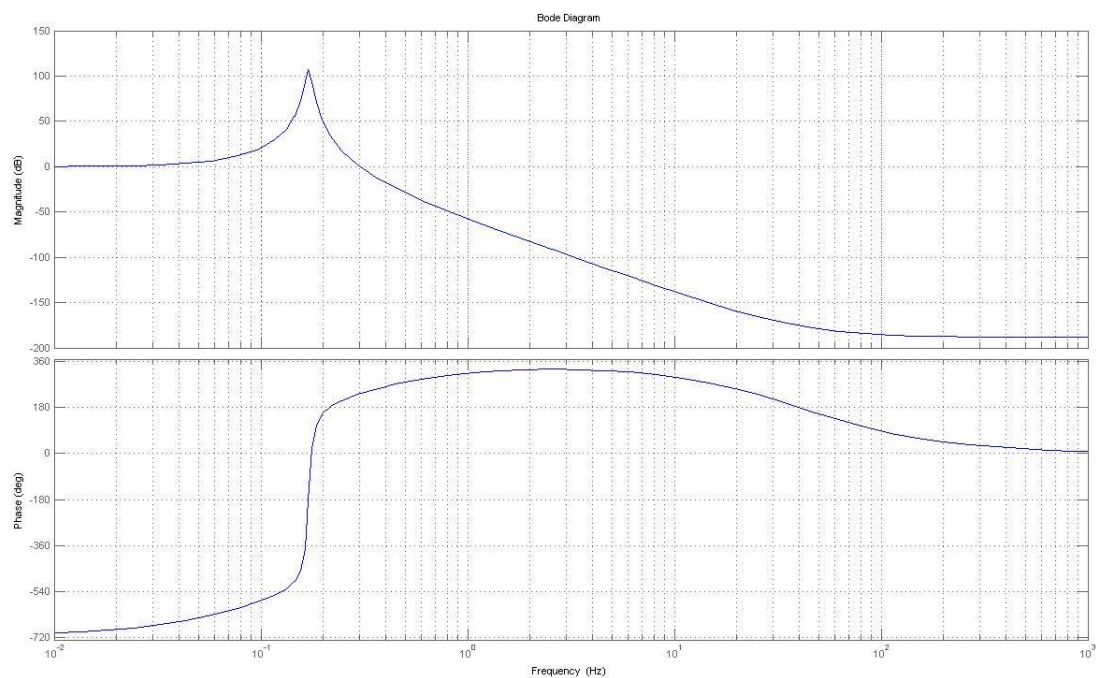
6. Ábra: A jelátalakító eszköz funkcionális blokkvázlata



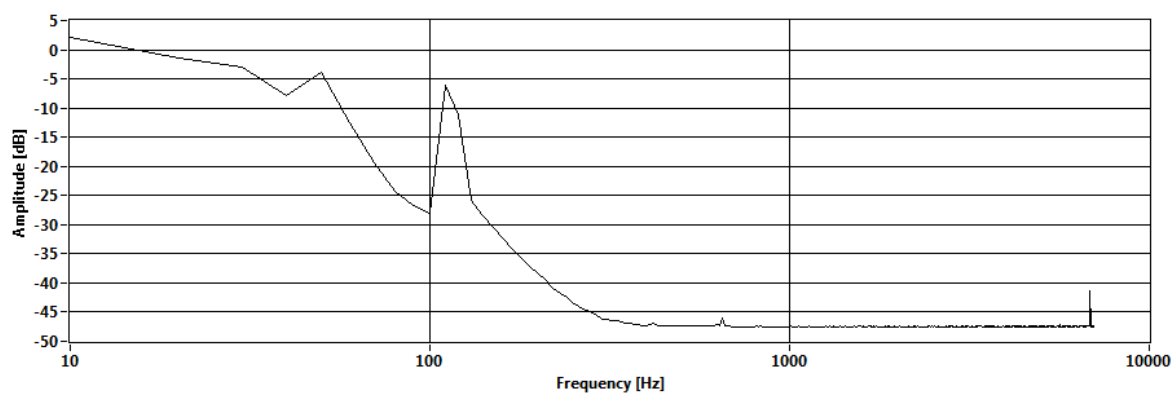
5. Ábra: Egy megvalósított szűrő struktúrája



6. Ábra: A megvalósított másodfokú szűrő Bode diagramja



7. Ábra: A megvalósított nyolcadfokú szűrő Bode digramja



8. Ábra: Egy megvalósított szűrő mért átvitele