Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Forró Zsigmond Attila

MIKROPROCESSZOR ALAPÚ MIDI SZINTETIZÁTOR FEJLESZTÉSE

KONZULENS
Dr. Firtha Gergely

BUDAPEST, 2021

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Forró Zsigmond Attila**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2021. 12. 10.

.....
Forró Zsigmond Attila

Összefoglaló

Célom egy olyan polifonikus szintetizátor elkészítése, mely a szokásosnál nagyobb szabadságot kínál a hangok előállításában, különböző hanghatások létrehozásával kapcsolatos kísérletekhez. A bemenetre MIDI eszközt csatlakoztatva hajtható meg, a kimeneti jelforma pontos alakja a felhasználó által valósidőben, játék közben is állítható. A kimeneti jelalakot a szinuszhullám felharmonikusainak összegeként adható meg, a hangleütés pillanatában induló amplitúdó burkoló alakja a zeneiparban elterjedt ADSR paraméterekkel állítható. A szintetizátorban megtalálható egy vibrato és egy egyszerűbb chorus effekt, melyek segítségével tovább manipulálható a hangzás.

A hanggenerálás digitálisan történik, a számításokat egy mikroprocesszor végzi. A működés során elvégzendő feladatok az alábbiak: a MIDI szabványnak megfelelő üzenetfogadó áramkör és dekódoló program megvalósítása, a felhasználói felületről érkező információk feldolgozása, valamint az ezek által meghatározott hangzás és a leütött hangok alapján, az analóg kimeneti jel előállítása. Ezen feladatokat lefutási gyakoriságuk szerint különböző időzítők által vezérelt megszakítási rutinokban implementáltam. A szakdolgozat során ezen programblokkok megtervezésével és hardveres feladatok megoldásával kapcsolatos megfontolásokat és a pontos működést tárgyalom.

Az elkészült eszközről elmondható, hogy nem csak hanghatásokkal kapcsolatos kísérletekre, de zenei előadásra is teljes mértékben alkalmas, a MIDI szabványnak megfelelő eszközökkel meghajtható digitális hangszer.

Abstract

My goal is to develope a polyphonic digital synthesizer that offers more freedom than usual in the production of sound shapes, for experiments with different sound effects. It can be driven by a connected MIDI device on the input, the exact sound can be modified in real time by the user using potentiometers. The output waveform can be specified by the sum of harmonics of the sine wave and the amplitude envelope can be set by ADSR parameters that are commonly used in the music industry. The synthesizer also includes a vibrato and a simple chorus effect to manipulate the sound further more.

The sound is generated digitally and the calculations are performed by a microprocessor. The operational tasks are the following: creating a reciever circuit, and implementing a decoder program for recieving standard MIDI messages, processing the information from the user interface and generating the analogue output signal based on the specified tone and the notes that are currently pressed on the connected MIDI keyboard. These tasks are implemented in interrupt routines called by different timers according to their frequency of execution and priority. Throughout this thesis, I discuss the design considerations for these program blocks and their exact operation.

The resulting instrument can be said to be a fully capable digital instrument not only for experimenting with sound effects, but also for musical performance, that can be driven by any standard-compliant MIDI device.

1. Bevezetés

1.1. Tervek, lehetőségek

A szintetizátor tervezése során céljaim a következők:

A hangok előállítása történhessen

- alap és felharmonikus komponensek kombinálásával, vagy
- *tárolt, kívülről importált minták alapján, vagy*
- *időtartománybeli megadással.*

A különböző hanghatások előállításához álljanak rendelkezésre az alábbi funkciók:

- polifónikus működés,
- ADSR automatikus amplitúdóvezérlés,
- vibrato effekt (frekvenciamoduláció),
- lehetőség mesterségesen megtöbbszörözött hangok elhangolására egymáshoz képest (egyszerű chorus effekt),
- a hanghatások módosítására játék közben is legyen lehetőség.
- *sztereó kimeneti jel előállítása*

(A dőlt betűvel szedett funkcióknak a szakdolgozat elkészültekor csupán az alapjait teremtettem meg, hogy a továbbfejlesztés ebben az irányban lehetséges legyen.)

1.2. Tervezési megfontolások

A valósidőben történő hangszintézishez, hogy a kimeneti érték számolásakor időt spóroljunk, a kiadni kívánt jel egy periódusát előre legeneráljuk és a memóriában eltároljuk. Ennek a hullámformának a frissítését csak a mintavételi frekvenciánál jóval ritkábban tesszük meg. A hangszín változtatásának sebességére vonatkozó elvárásainkat másodpercenként 10-szeres hullámforma generálással is kielégíthetjük, ugyanakkor a mintavételi frekvencia sűrűségével elvégzendő feladatokat jóval lerövidíthetjük, ha csak ebből az eltárolt mintából nézzük ki az értékeket, valamint a minták között interpolációt végzünk. Egy generált periódust 128 mintában tárolunk. Magát a fázist 16 biten ábrázoljuk, tehát az interpoláció után egy periódus $2^{16} = 65536$ mintából áll. A kimeneti analóg jel felbontását legalább 11 bites felbontással konvertáljuk a kimeneten, ez később igény és lehetőség szerint növelhető.

A polifonikus működés megvalósításához egyszerre legfeljebb 10 leütött hangot tartunk számon, ez a legtöbb alkalmazás során elegendő. Ezeket a hangokat a program során úgynevezett hangrekeszekben tároljuk, melyek erre a célra kialakított a hangok paramétereit tároló struktúrák.

A feladatkiírásban szerepelt sztereó kimeneti jel előállítása, azonban konzulensemmel egyeztetve, a választott hardver adottságai és egyéb funkciók nagyobb prioritása miatt arra jutottunk, hogy mégis egycsatornás kimenettel látom el a szintetizátort.

2. Fejlesztői környezet

2.1. Processzor és fejlesztőkörnyezet kiválasztása

A processzor kiválasztásakor szempontjaim a következők voltak:

- A processzor legyen elérhető fejlesztőkártyán.
- A fejlesztőkártya megfizethető árú legyen (néhány ezer forint).
- A fejlesztőkörnyezet lehetőleg legyen ingyenes, vagy a fejlesztőkártyával együtt megfizethető árú.
- A kiválasztott környezetben a fejlesztés a rendelkezésre álló idő alatt kivitelezhető legyen (nagyjából 1 hónap munka teljes munkaidőben számítva egy ember számára).
- A kiválasztott processzor alkalmas legyen az elvégzendő feladatokra (megfelelő méretű memória, gyors aritmetikai műveletvégzés legalább fixpontos 32 bites adatokon, megfelelő felbontású és sebességű DA és AD konverzió támogatása).

A kiválasztás során az alábbi lehetőségeket vizsgáltam meg:

- *Valamilyen kifejezetten jelfeldolgozási célokra fejlesztett processzor (DSP):* Műszaki szempontokat figyelembe véve kétségtelenül ez lenne a legmegfelelőbb választás, azonban magas ára miatt ezt a kategóriát elvetettem.
- *Arduino fejlesztőkártya*: Ezek a kártyák megfizethetők és különösen az Atmel SAM3X8E ARM Cortex-M3 CPU-val felszerelt Arduino Due alkalmas lenne a feladatra, az ARDUINO IDE fejlesztőrendszer ingyenes, nagyon sokak által használt. Éppen ezért rengeteg ingyenesen letölthető könyvtárban szinte mindenre található funkció, melyek felhasználásával rendkívül gyorsan létrehozható egy működő rendszer. Ugyanakkor a hibakeresési eszközök szinte teljes hiánya bonyolultabb fejlesztések során komoly nehézségeket okozhat.
- *C2000 Delfino MCU F28379D LaunchPad™ development kit*: Ez a fejlesztőkártya az általános célú processzorok és a speciális célú drága jelfeldolgozó processzorok között jó választásnak tűnik. Bár ára felülről súrolja az általam elfogadhatónak tartott határt, nagy számítási teljesítménye, a két processzormag, a hardveres lebegőpontos műveletvégzés nagyon vonzóvá teszik a feladatra. Programozásához az ingyenes Code Composer Studio fejlesztőrendszer használható. Be is szereztem egy ilyen kártyát, azonban hamar kiderült, hogy ennek a rendszernek a tanulási görbéje igen lapos, így arra a következtetésre kellett jutnom, hogy a tervezett fejlesztés a rendelkezésre álló idő alatt nem megvalósítható.
- *CY8CKIT-059 PSoC® 5LP Prototyping Kit*: A Cypress CY8C5888LTI-LP097 chipjét tartalmazó fejlesztőkártya. Ez egy úgynevezett Pragrammable System On a Chip rendszer, ami programozható digitális és analóg blokkokat, perifériákat és egy 32 bites Arm Cortex-M3 processzort tartalmaz. A fejlesztést az ingyenes Psoc Creator fejlesztőkörnyezet teszi lehetővé, melynek grafikus felületén nagyon gyorsan kialakítható a kívánt architektúra, és az így megadott rendszer alapján automatikusan generálódó könyvtárak nagyban megkönnyítik a programozást is. Az adatlap szerint szinte minden tekintetben kielégíti a követelményeinket, nehézséget a hardveres

lebegőpontos műveletvégzés hiánya, valamint a D/A konverterek 8 bites volta jelent, azonban ezek nem okoznak áthidalhatatlan problémát. A vonzó fejlesztési környezet, ár és a feladatra való alkalmassága miatt ezt a rendszert választottam.

2.2. PSoC és a PSoC Creator

Az egychipes programozható rendszer (Programmable System on a Chip, továbbiakban PSoC) eszköz kategóriát a Cypress Semiconductor vezette be 2002-ben, valószínűleg az általa korábban is nagy mennyiségben gyártott programozható logikai tömb (FPGA) termékek továbbfejlesztéseként. Egy PSoC egy vagy több processzorból, különböző memória tömbökből, konfigurálható digitális és analóg blokkokból, és programozható összekötő hálózatból áll. A digitális blokkok részben előre meghatározott, bár konfigurálható funkciókat látnak el, részben univerzálisak, melyekből a legváltozatosabb alkalmazások hozhatók létre.

A rendszer legfőbb vonzerejét számomra ezek a programozható logikai és analóg blokkok, valamint ezeknek PSoC Creator által megvalósított, a függvénykönyvtárakkal összehangolt interaktív kezelése alkotja. Ha létre szeretnénk hozni például egy D/A konvertert, akkor egyszerűen behúzhatjuk azt a munkaterületre, ha szükséges, összekötjük a már ott lévő komponensekkel, majd arra kétszer rákattintva beállítjuk annak paramétereit. Ezek után elindítva a *build* funkciót, generálódnak a szükséges függvénykönyvtárak és konfigurálódnak az összeköttetések. Ezután beleírhatjuk a saját kódunkat a megfelelő rutinokba és a főprogramba. A komponensek behúzása nem csak a hardver konfigurációját végzi el, hanem, ha szükséges, a szoftver modulokat is generálja. Ha például egy UART esetében a hardverben rendelkezésre állónál nagyobb puffert definiálunk, automatikusan generálódik a megszakítási rutin, ami a nagyobb szoftver puffert megvalósítja. Egyes esetekben kiválaszthatjuk, hogy az adott elemet előre definiált digitális funkciók, vagy univerzális digitális blokkok segítségével hozzuk létre. Utóbbi nagyobb konfigurációs lehetőséget biztosít.

A programozható összekötő hálózat nagyon nagy rugalmasságot biztosít az I/O lábak és a belső funkciók közötti összeköttetés megvalósítására. Szinte minden funkció kivezethető minden lábra, ami nagyban leegyszerűsítette a panel huzalozását. (Kivételt képeznek ez alól a kommunikációs perifériák, melyek csak kommunikációs lábakra vezethetők ki.) Az összeköttetések nem csak programozhatók, hanem működés közben is átkonfigurálhatók, így például nagy multiplexerek alakíthatók ki. A meglévő lábak jó kihasználását segíti, hogy a belső építőelemeket anélkül is összeköthetjük, hogy kivezetnénk őket egy lábra.

Ami igazán megdöbbentő, és választásomat döntően befolyásolta, az a chipre és az azzal kapcsolatos függvénykönyvtárakkal kapcsolatos dokumentáció mennyisége, minősége és a fejlesztő rendszerrel való integrációja. A programfejlesztés vagy hardware konfigurálás során egy kattintással azonos formátumban hozzáférhetünk minden, a keresett funkcióval kapcsolatos dokumentumhoz, legyen az akár valamilyen hardver részlet, vagy függvénykönyvtár. Ez az integrált fejlesztői környezet tette lehetővé, hogy a kívánt funkcionalitást viszonylag rövid idő alatt kifejlesszem annak ellenére, hogy semmilyen tapasztalatom nem volt korábban ezzel az eszközzel kapcsolatban.

3. MIDI kommunikáció

3.1. A MIDI protokollról

Az 1980-as évek elejéig, habár az első digitális hangszerek már megjelentek, nem volt egységes mód a különböző hangszerek, stúdióeszközök összeköttetésére. Különböző gyártóknak voltak a saját eszközeik közötti kommunikációt lehetővé tevő megoldásaik, de ezek használata az egységesítés hiánya miatt igencsak körülményes volt a zeneipar számára. Erre kínált megoldást a MIDI (**M**usical **I**nstrument **D**igital **I**nterface) szabvány. Az első MIDI-kompatibilis eszközök közé tartoztak a sokak által ismert *Roland Jupiter-6* és a *Prophet 600* szintetizátorok.

A MIDI kommunikáció során maga a zenei analóg jel helyett a lejátszandó hangoknak a zenész játékára vonatkozó tulajdonságait küldjük el, pl. leütötte a C4 billentyűt 100-as erősséggel. Vagy éppen felengedte ezt a billentyűt, vagy lenyomta a sustain pedált. A hangot magát az ezeket az üzeneteket fogadó másik MIDI eszköz állítja elő. Felmerülhet bennünk az igény, hogy habár klarinéton játszani nem tudunk, mégis jó lenne, ha amit a digitális zongoránkon játszunk, az nem zongora, hanem klarinét hangon szólalna meg. Ezt könnyedén megvalósíthatjuk, ha van egy klarinét hangot kiadni képes szintetizátorunk és annak MIDI bemenetét összekötjük a zongora MIDI kimenetével. Minden billentyűleütéskor illetve felengedéskor a billentyűzet üzenetet küld a szintetizátornak, melyben kódolja a hang magasságát, illetve billentyűlenyomás esetén a leütés erősségét, ezek alapján pedig a szintetizátor a megfelelő klarinét hangot generálja a billentyű lenyomásától kezdve a felengedésig. A MIDI széleskörű elterjedésével, illetve a személyi számítógépek MIDI támogatásával rengeteg új lehetőség nyílt meg a digitális zenekészítés számára, a hangnak ilyen módon való szétválasztása játék-információkra és hangszínre nagyon hasznosnak bizonyult, hasonlóan a kottához.

Az eszközök felhasználási módjuknak megfelelően MIDI IN, MIDI OUT, vagy MIDI THRU csatlakozókkal lehetnek ellátva. Egy zongorabillentyűzet (ami nem tud hangot kiadni), általában csak MIDI OUT csatlakozóval rendelkezik, amit összeköthetünk egy szintetizátor MIDI IN bemenetével, ami az azon érkező MIDI üzenetek alapján hangot generál. Ha további eszközöket is meg szeretnénk hajtani ugyanezzel a billentyűzettel, a szintetizátoron lévő MIDI THRU csatlakozón továbbíthatjuk a MIDI IN-en érkező üzeneteket. A küldő eszköz 16 különböző csatornán (channel) küldhet üzeneteket, amiket csak az azokra a csatornákra „figyelő” eszközök dolgoznak fel, így akár egy egész zenekart összeállíthatunk, ahol a teljes MIDI hálózaton keresztülhaladó üzenetekből minden tag csak a neki megfelelő csatornán érkezőket hajtja végre (lásd 2-es ábra).

Hagyományosan a MIDI kommunikációhoz használt kábel 180°-os ötvezetékes DIN csatlakozóval van ellátva, de a modernebb eszközök már támogatják az USB-n keresztüli adatátvitelt is. Egy kábelen az információ csak egy irányban haladhat, kétirányú kommunikációhoz újabb vezetékre van szükség. A MIDI szabvány fizikailag egy aszinkron soros kommunikációt valósít meg, melynek sebessége 31250 baud/s. Ez meglepő lehet az ötlábú csatlakozót tekintve, azonban a legtöbb alkalmazás során csupán egy vezetéken halad az adat, a nem szükséges lábak nincsenek bekötve. Kivételt képeznek egyes eszközök, melyek tápfeszültséget is fogadnak a MIDI vezetéken.

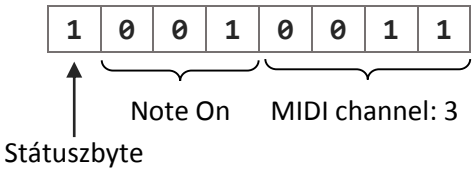
3.2. MIDI üzenetek

A MIDI szabvány számtalan lehetőséget ad a zenei eszközök közötti kommunikációra, hangszínre vonatkozó utasításoktól elkezdve az eszközök óráit szinkronizáló rendszerüzenetekig. Jelen esetben nem célom egy minden lehetséges MIDI üzenetet feldolgozó és azokra megfelelő választ adó eszközt tervezni, csupán egy szűk csoportjukkal foglalkozom. Ezek pedig a zenész játékát leíró, billentyű leütéséről vagy felengedéséről információt hordozó Note On és Note Off üzenetek.

MIDI kommunikáció során, hogy a valós idejű játék lehetséges legyen, egy billentyű leütése és felengedése két külön esemény. Leütéskor a billentyűzet egy Note On üzenetet küld, majd ha felengedték azt a billentyűt, egy Note Off üzenetet.

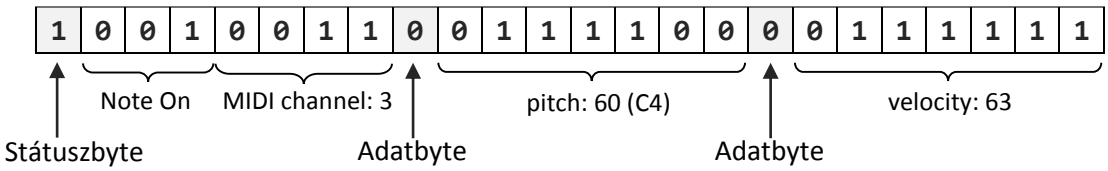
Nem feltétlenül mindig, de a legtöbb esetben ez a két üzenettípus három bájtból áll. Az első az úgynevezett státuszbájt, ezt pedig követi két adatbájt. A státuszbájt első bitje mindig 1, az ezt követő három bit megadja az üzenet típusát pl. Note On (`001`), vagy Note Off (`000`), a maradék négy bit pedig kiválasztja a 16 MIDI csatorna egyikét.

Példa: A hármas számú MIDI csatornán érkező Note On üzenet elejét jelző státuszbájt:



Mindkét üzenet esetén a státuszbyte után két adatbyte következik, melyek közül az első a leütött hang magasságát adja meg, a második pedig a leütés erősségét. A státuszbájtokkal ellentétben az adatbájtok első bitje mindig 0, így az előbb említett adatokat a maradék 7 biten ábrázolhatjuk.

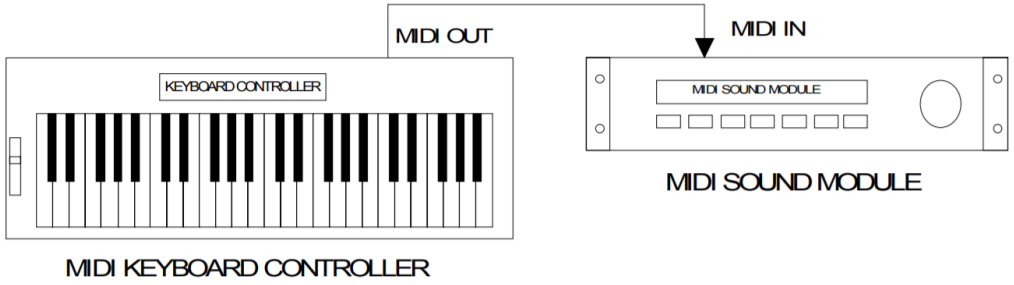
Ha leütjük a C4 hangot közepes erővel (63/127) egy billentyűzeten, amely a hármas MIDI csatornán kommunikál, az alábbi üzenet áll elő:



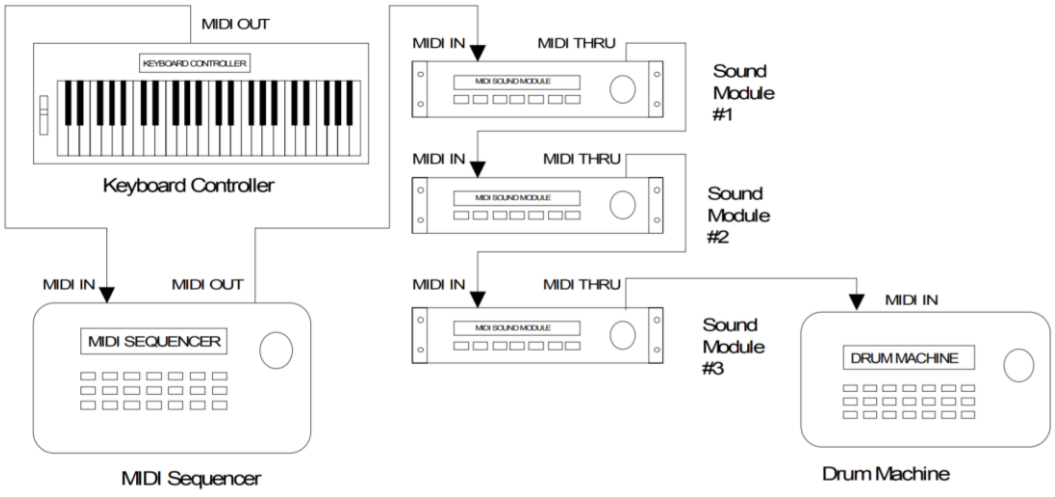
Note Off üzenetnél nagyon hasonló a helyzet, de meg kell jegyezni, hogy habár léteznek eszközök, melyek képesek erre, a legtöbb billentyűzet nem tudja mérni a felengedés sebességét és nem küld értelmezendő adatot a Note Off üzenet második adatbájtjában.

3.3. Lehetséges alternatívák a MIDI kommunikációban

Sok MIDI forrás, sőt a MIDI eszközök többsége a Note Off üzeneteket nem a státuszbájtban jelzi, hanem ugyanúgy Note On üzenetet küld, mintha egy leütés lenne, csak a leütés erőssége ezekben az üzenetekben 0. Emellett az sincsen kőbe vésve, hogy feltétlenül el kell küldeni a státuszbájtot. Amennyiben egymásután sok megegyező státuszbájttal rendelkező üzenetet küld egy billentyűzet, megteheti, hogy csak az elején küld egy státuszbájtot, utána pedig már csak sorban az adatbájtokat. A fogadó eszköz, ha kimarad a státuszbájt és rögtön adat jön, akkor a legutóbbi státuszbájtot veszi figyelembe. Egy szintetizátor megtervezése során tehát fontos, hogy ezeket a különböző lehetőségeket megfelelően kezeljük.



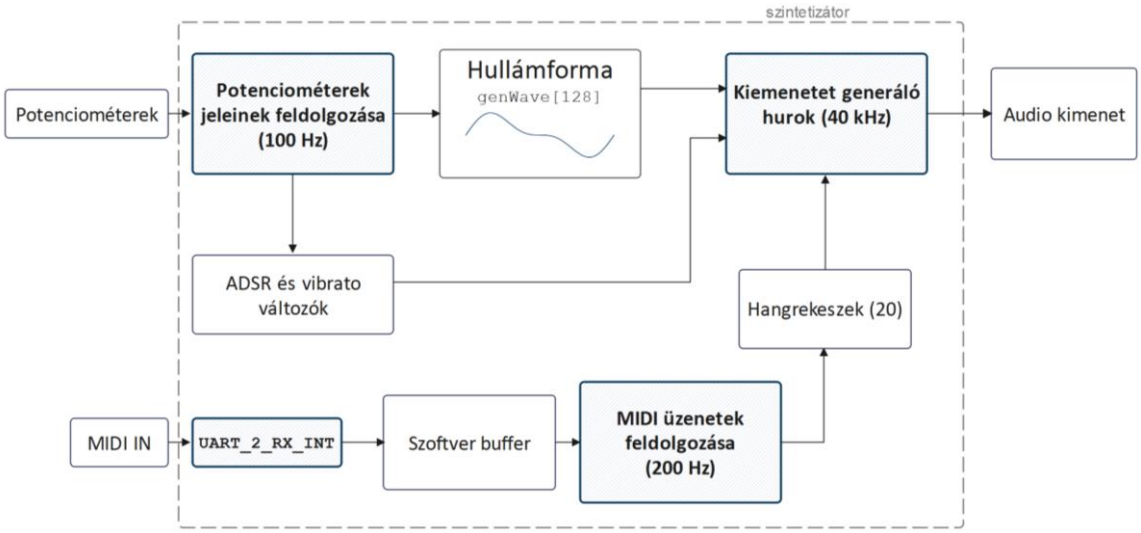
Ábra 1 Egy egyszerű MIDI rendszer (forrás: The Complete MIDI 1.0 Detailed Specification)



Ábra 2 Egy összetett MIDI rendszer (forrás: The Complete MIDI 1.0 Detailed Specification)

4. Működés

A szintetizátor szoftverének működését az alábbi ábra szemlélteti:



ÁBRA 3 A SZINTETIZÁTOR RENDSZERTERVE

4.1. A megszakítási hurkok

A működést megvalósító program időzített megszakítási rutinokból épül fel. A feladatokat lefutási gyakoriságuk és prioritásuk szerint 3 nagyobb blokkra osztottam. Egy 100 Hz frekvenciával végbemenő megszakítási hurokban történik a potenciométerekről származó jelek fogadása és feldolgozása, egy másik, 200 Hz frekvenciával lefutó hurokban zajlik a MIDI üzenetek feldolgozása és egy a mintavételi frekvenciának megfelelő 40 kHz-es hurok végzi a kimeneti jel mintáinak kiszámítását. Ezen kívül egy apró, de annál fontosabb feladatot meg kell említenem itt, ez pedig a MIDI bemenetként használt soros portra érkező üzenetek szoftver pufferbe való eltárolása. Ezekre a megszakítási rutinokra a dolgozat során hurokként fogok hivatkozni.

A megszakításoknak és így a rutinok által végzett feladatoknak is különböző prioritásuk lehet, egy nagyobb prioritású megszakíthat egy kisebb prioritásút, de fordított esetben a kisebb prioritásúnak meg kell várnia, míg végez a nagyobb prioritású. Érdemes tehát alaposan meggondolni, hogy a 4 megszakítási rutinnak és a bennük lévő feladatoknak milyen prioritást választunk, vegyük sorra a szempontokat.

Digitális hangszerről lévén szó, értelemszerűen a legfontosabb, hogy mintavételi időnként ki kell számolni a mindenkori kimeneti értéket, nem fordulhat elő például az, hogy letelt a mintavételi idő és a processzor nem generálja a kimenetet, mert éppen egy MIDI üzenet feldolgozásával van elfoglalva. Ennek a blokknak pontosan tartania kell magát a mintavételi frekvenciához, nem lehetnek kimaradások, vagy késések. Tehát a 40kHz-es hangjel generáló megszakítási rutint választottam a legnagyobb prioritásúnak. Ezt követi a hasonlóan kritikus fontosságú, de szerencsére igen rövid feladat, a soros porton érkező MIDI üzenetek hardver pufferből szoftveres pufferbe helyezése. Mivel a szintetizátor egy hangszer kiszolgálására készült, arra kell számítani, hogy rövid idejű nagy adatsebességű átvitelek lehetnek, az átlagos

forgalom azonban jóval a maximális sávszéleség alatt marad. Másodpercenként 40 hang leütésével és felengedésével számolva az adatforgalom *40*3*2=240* bájt másodpercenként. Arra azonban számítani kell, hogy tíz hangot közel egyidőben ütnek le, ami egy 30 bájtos adat csomót jelent. Bár a MIDI UART-ot kiszolgáló megszakítás csak a második legmagasabb prioritású, a nála egyedül magasabb szintű kimenetet generáló hurok nagyságrendileg nagyobb gyakorisága miatt mégis időben végbe tud menni a bejövő MIDI üzenetek eltárolása a szoftver pufferbe a kimenetgenerálási blokkok között. Ha ez nem történne meg időben és elvesznének MIDI üzenetek, akkor kimaradhatnak hangok, vagy kimaradó NOTE OFF üzenet esetén, egy bekapcsolódott hang annak újbóli leütéséig szólhat, ez azonban még a kisebb gond. A hardver puffer túlírása az üzenetek összekeveredését is okozhatja, ami miatt egész más frekvenciájú hangok is megszólalhatnak, mint amilyen hangot leütöttek. Ilyenkor ugyanis a leütés erősségét kódoló bájtokat a hang magasságára vonatkozó üzenetként, vagy éppen fordítva értelmezhetjük.

A megmaradt feladatok, mint a MIDI jelek feldolgozása, potenciométerek mintavételezése, a hullámforma és az ADSR görbe generálása a fentiekhez képest nagyságrendileg ritkábban zajlanak le, valamint a halasztást is sokkal jobban tűrik. A bejövő MIDI jelek feldolgozását egy másodpercenként 200-szor, a potenciométerek mintavételezését illetve a hullámforma és az ADSR görbe frissítését egy másodpercenként 100-szor lefutó hurokban implementáltam. A bejövő jeleket analóg multiplexerek segítségével osztom szét két A/D konverterre, egy ciklusban csak 2 potenciométert olvasunk ki. Így a hullámforma csak minden nyolcadik lefutáskor fog frissülni, azaz 12,5 Hz gyakorisággal. Ez a két legkisebb prioritású feladat, a MIDI jelek feldolgozása a nagyobb, a potenciométerek kiolvasása a kisebb. A főprogram a legkisebb prioritású, ebben jelenleg az inicializáláson kívül nincs feladat.

A tervezett fejlesztések egy része (pl. grafikus kijelző kezelése) a főprogramba, a nyomógombok kezelése és az ezzel összefüggő LED kezelés a 100 Hz-es hurokba kerülhetne.

4.2. Hangrekeszek

	HANG 0	HANG 1	HANG 2	...	HANG 19
Amplitúdó					
MIDI amplitúdó					
Fázislépés					
Fázis					
ADSR státusz					
Decay					
Sustain					
Release					
Decay felezése					
ADSR számlálók					

ÁBRA 4 HANGREKESZEK FELÉPÍTÉSE

A szintetizátor a kiadandó hangokat a működés során úgynevezett hangrekeszekben tárolja, melyeknek felépítése a 4-es ábrán látható. Ezekből a hangrekeszekből 20 darab van, 10 szolgál a zenész játékából származó hangok tárolására, 10 pedig mesterségesen van feltöltve az előbbi 10 hang alapján. Ezek az utóbbi hangok nem önállók, a leütött hangok megduplázásával, oktávval való eltolásával és finom elhangolással képződnek. Amennyiben leütöttek egy billentyűt és meg kell szólaltatni egy hangot, akkor a leütésből következő hangokat a program elhelyezi az egyik ilyen rekeszpárban. A 20 helyet a hangoknak a programban paraméterenként 20 elemű tömbök valósítják meg, melyek mindegyikében az i-edik indexű elem az i-edik hangra vonatkozik. A kimeneti jelet generáló hurok minden alkalommal sorra veszi mind a 20 hangrekeszt és azok alapján számítja a D/A konverterre vezetendő kimeneti jelet. Pár mondatban vegyük sorra a rekeszek változóinak szerepét a működés során!

Az **amplitúdó** változó a hang amplitúdóját tárolja, ez határozza meg, milyen hangosan szól a hang. Fontos azonban, hogy ennek értéke a billentyű lenyomva tartása alatt változhat az ADSR görbe mentén.

Ugyanez nem igaz a **MIDI amplitúdó** változóra, amit egyszer beállítunk a beérkező Note On üzenet leütési erősséget kódoló bájtja alapján. Ez az érték célul szolgál majd az időben változó amplitúdó értéknek. Az ADSR görbén haladás során az attack szakaszon, az amplitúdó 0-ról indulva felfut a MIDI amplitúdóig. Fontos, hogy a MIDI üzenet feldolgozó hurok csak a *MIDI amplitúdó* változót írhatja, míg a kimenetgeneráló hurok (ami az ADSR szerinti amplitúdó módosítást végzi) csak az *amplitúdó* változót írhatja, a *MIDI amplitúdót* csak olvashatja.

A **fázislépés** a hang frekvenciájával arányos érték, annak magasságát adja meg. A hanggenerálás során egy periódusnyi mintában lépkedve számítjuk a kimeneti értéket, nagyobb lépésenként haladva nagyobb frekvenciájú kimeneti jelet állítunk elő. Mivel a kimenetgeneráló hurok a fázislépés alapján működik, célszerű már eleve ezalapján tárolni a hangokat.

A **fázis** változót a kimenő jel generálását végző hurok használja, ezalapján tudja, hogy éppen hol tart az adott hang a fentebb említett egy periódust tartalmazó hullámforma mintái között.

Az **ADSR státusz** azt adja meg, hogy az adott hangrekesz éppen attack, decay sustain vagy release állapotban van.

Mivel a **decay** és a **release** szakaszokon nem állandó a hangok amplitúdójának csökkenési sebessége, ezért minden rekesznek van külön példánya ezekből a paraméterekből, ami az idő elteltével változhat. Ezek a paraméterek tehát a rekesz amplitúdójának csökkenési sebességét befolyásolják, azonban fordítottan arányosan, valójában azt adják meg, hogy hány kimenetet generáló ciklusnak kell eltelnie egységnyi amplitúdó csökkenés alatt. Mivel az attack szakasz lineáris, az amplitúdó növekedési sebessége nem változik a hangok között, ott erre nincs szükség, elég egy változó az összes rekeszre.

A **sustain** paraméter az a szint, ahova beáll a hang amplitúdója „sokáig” lenyomott billentyű esetén a decay szakasz után. Ez mindig a MIDI amplitúdó és 0 közé normált érték, így a billentyűleütés erősségétől és egy a sustain szintet állító potenciométer állásától függ.

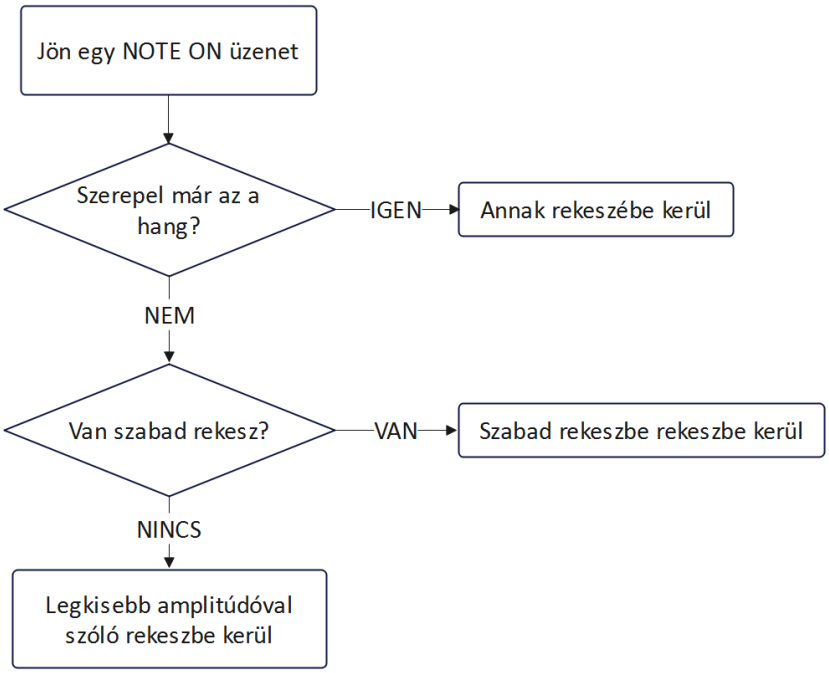
Végül pedig minden rekeszhez van egy **ADSR számláló** paraméter, aminek segítségével biztosított az ADSR görbe aktuális meredeksége az adott hangra nézve.

Bejövő hangok tárolására szolgáló hangrekeszt az alábbiakban leírtak szerint választjuk ki. Ugyanolyan magasságú hangokat nem tárolunk el több rekeszben, így az első lépés a helyválasztás során, hogy megnézzük, nem szól-e már egy ugyanolyan magasságú hang egy aktív rekeszben. Amennyiben találunk ilyet, akkor annak helyére kerül be az új hang. Elsőre talán nem egyértelmű, hogy miért jönne kétszer ugyanaz a hang úgy, hogy még az előző szól, hiszen ugyanannak a billentyűnek két leütése között felengedésnek is kell történnie, akkor pedig ki kellene kapcsolnunk azt a hangot. A magyarázat az, hogy ha nagyon hosszú ideig tartó lecsengést állítunk be a szintetizátoron (nagy release), akkor miután felengedtük a billentyűt, az adott hang nem fog rögtön megszűnni (nem szabadul fel rögtön a rekesze), hanem fokozatosan halkul akár másodpercekig, ez idő alatt pedig újra leüthetjük a billentyűt. Ha ilyenkor egy külön rekeszbe helyeznénk el az új hangot, akkor gyors nyomogatás esetén megtelhet az összes rekesz ugyanolyan magasságú hanggal, közben pedig nem is hallanánk már a hangok release szakaszát, hiszen frekvenciájuk nem különbözik.

Tehát az első lépés új hang számára hangrekesz keresésekor, hogy megvizsgáljuk, van e már olyan magasságú aktív hang. Amennyiben van, akkor annak rekeszét választjuk. Amennyiben nem találunk ilyet, akkor értelemszerűen egy „üres” hangrekeszt próbálunk találni, ami azt jelenti, hogy amplitúdója 0. Ha találunk ilyet, megvan az új rekesze, ám előfordulhat az is, hogy mind a 10 hely foglalt. Amennyiben egyszerre szól 10 hang és a felhasználó leütött egy újat, megkeressük azt a hangrekeszt, ami a legkisebb amplitúdóval szól és annak a helyére helyezzük el az új hangot. Mégiscsak kevésbé kellemetlen, ha játék közben elhallgat egy régóta szóló amúgy is halk hang, minthogy hiába ütjük a billentyűzetet, nem szólalnak meg új hangok. A 10-es polifónia szint természetesen minden további nélkül növelhető, csupán arra kell figyelni, hogy a hangot generáló modul rendelkezzen elegendő kapacitással a hangok egyidejű megszólaltatására. Új hang elhelyezésekor a program mindenképpen végignézi az alsó 10 rekeszt, hogy megtalálja, szerepel e már megegyező frekvenciájú hang. Ahogy halad, folyamatosan frissíti a legkisebb amplitúdójú hang indexét, illetve azt is „feljegyzi”, ha talált egy szabad helyet. Amennyiben megtalálja ugyanazt a hangot az egyik rekeszben, akkor oda rögtön beteszi az újat, nem is keres tovább, azonban ha nem szerepelt még az a hang, akkor a 10 rekeszt átnézve, birtokában lesz egy olyan rekesz indexe, ami vagy üres, vagy ugyan szól, de a legkisebb az amplitúdója.

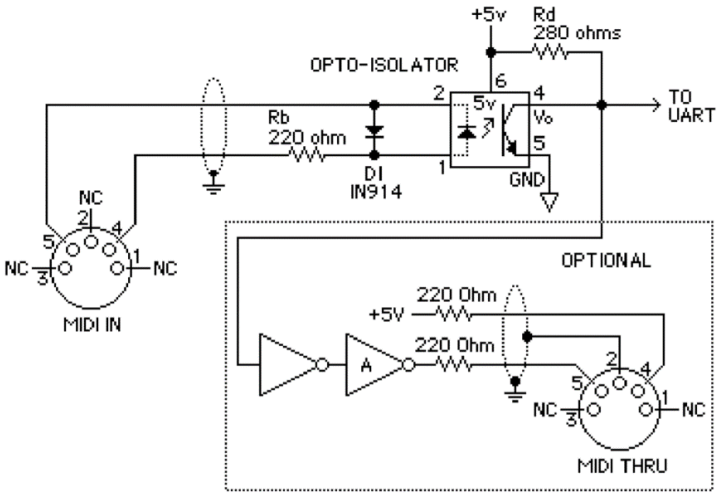
Amennyiben billentyűfelengedésről érkezik MIDI üzenet, úgy csupán a megfelelő hangmagasságú elemet kell megtalálni a rekeszek között, sorban átnézve azokat. Mivel a felső 10 rekesz hangjai az alsó 10 rekeszekben lévő párjaikkal együtt vannak kezelve, ezért Note On és Note Off üzenet esetén is csak az alsó 10 rekeszt szükséges vizsgálni.

Fontos megjegyeznem, hogy habár a 200Hz-es hurokban történik a MIDI üzenetek feldolgozása, illetve a pufferben létük ellenőrzése, amennyiben sűrűn jönnek egymás után a MIDI bájtok, azokat gyorsan dolgozza fel a hurok, ugyanis ha belekezd a szoftver puffer kiolvasásába, akkor azt addig nem hagyja abba, míg az ki nem ürül. Egy billentyűleütés esetén egymásután rögtön 3 MIDI bájt jön, ezek a MIDI protokoll sebességéből adódóan nagyjából 75 µs alatt megérkeznek. Az ennél az időnél sokkal ritkábban lefutó 200 Hz-es hurok nagy valószínűséggel rögtön mind a hármat a pufferben fogja találni, így egy körben ki is veszi mind a hármat és a hangrekeszek egyikében megjelenik a leütött hang, nem kell 3-szor megvárni a 200Hz-es lefutást. A MIDI üzenetek sebessége és azok feldolgozási gyakorisága közötti különbségből is látható, hogy miért van szükség a nagyobb méretű szoftver pufferre.



ÁBRA 5 FOLYAMATÁBRA A HANGREKESZ KIVÁLASZTÁSÁRÓL

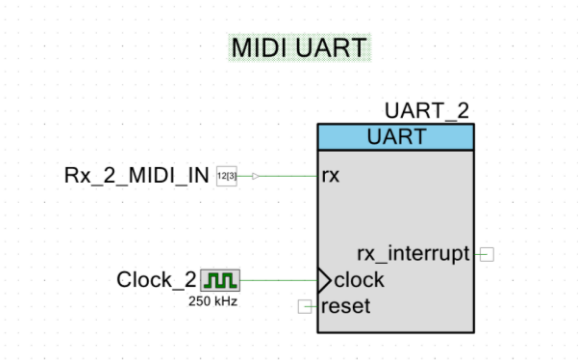
4.3. Beérkező MIDI üzenetek feldolgozása



ÁBRA 6 MIDI FOGADÓ ÁRAMKÖR (FORRÁS: THE COMPLETE MIDI 1.0 DETAILED SPECIFICATION)

MIDI üzenetek fogadására a *Complete MIDI 1.0 Detailed Specification* a 6-os ábrán látható hálózatot adja meg. A bemenet egy optikai csatoló IC-vel galvanikusan el van választva a hálózat többi részétől, ezzel megelőzhető a földhurkok kialakulása, illetve megvédi a MIDI eszközt a bemeneten esetlegesen váratlanul fellépő oda nem illő jelek kártékony hatásaitól.

A specifikációnak megfelelő áramkört készítettem próbapanelen, a MIDI specifikáció által ajánlott 6N138 típusú optikai csatoló felhasználásával. A felhasznált fejlesztőkártya soros portja 4 bájtos hardveres pufferrel rendelkezik, ami nagyon könnyen megtelhet, ha a MIDI szabvány közel 4000 bájt/s –os átviteli sebességét nézzük. Ezt a problémát kezelendő, egy szoftveres puffert hoztam létre. Amennyiben bájt érkezett a soros vonalon, megszakítási kérelem történik, mely ha érvényre jut, elmenti a bájtot a szoftveres pufferbe, ennek méretét 64 bájtra választottam. Mindez a Psoc Creator grafikus felületén egyszerűen beállítható. Az alábbi ábrán a jobb oldali a MIDI üzeneteket feldolgozó UART.

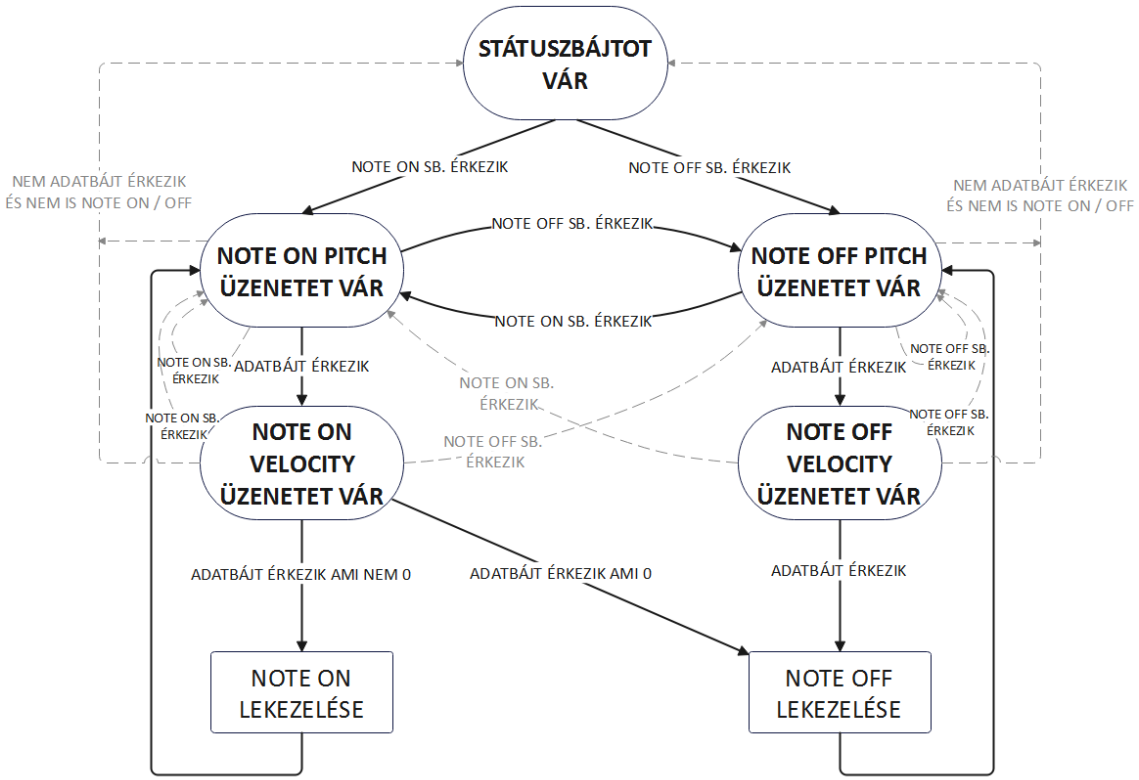


ÁBRA 7 MIDI ÜZENETEK FOGADÁSÁRA KONFIGURÁLT UART BLOKK PSOC CREATOR-BAN

A MIDI üzenetek dekódolása és feldolgozása a 200 Hz gyakorisággal lefutó megszakítási hurkokban történik. Mivel a bejövő üzenetek értelmezése függ a korábban beérkezett üzenetektől, ebben a hurokban egy állapotgépes működés zajlik. A MIDI feldolgozó állapotgépnek összesen 5 állapota van, reakciója a bejövő MIDI üzenetre attól is függ, milyen állapotban kapta azt. Ezek az állapotok az alábbiak:

- Státuszbájtra vár
- Note On pitch üzenetet vár
- Note On velocity üzenetet vár
- Note Off pitch üzenetet vár
- Note Off velocity üzenetet vár

A *3.3 „Lehetséges alternatívák a MIDI kommunikációban”* című fejezetben szó volt róla, hogy amennyiben nem érkezik státuszbájt, úgy a legutóbb érkezett státuszbájtot kell figyelembe venni. Kezdetben *státuszbájtra vár* állapotban van az állapotgép és ebből csak az mozdíthatja ki, ha jön egy NOTE ON vagy NOTE OFF típusú státuszbájt. Az alapján, hogy milyen típusú üzenetet kódolt a státuszbájt, átkerül NOTE ON pitch üzenetet vár, vagy NOTE OFF pitch üzenetet vár állapotba. Ha a következő bájt adatbájt, akkor azt elmenti pitch-ként és átvált *Note nN velocity üzenetet vár*, vagy *Note Off velocity üzenetet vár* állapotba. Amennyiben ezekben az állapotokban érkezik egy újabb adatbájt, azt elkönyveli velocity üzenetként, majd meghívja a megfelelő billentyűleütést, vagy éppen felengedést lekezelő függvények valamelyikét. Utóbbira kétféle esetben kerülhet sor, vagy ha eleve Note Off a státuszbájt, vagy pedig ha a státuszbájt ugyan Note On, a velocity adatbájt értéke azonban 0 volt.



ÁBRA 8 MIDI DEKÓDOLÓ ÁLLAPOTGÉP

Esetleges hibák elkerülése végett az állapotgép lekezeli az összes lehetséges esetet, ami történhet, habár üzemszerű működés során bizonyos esemény kombinációk nem lépnek fel. Ilyen például, hogy jön egy Note On státuszbájt, azt követi egy adatbájt, majd egy Note Off státuszbájt. Azaz kimaradt a leütés erősségét kódoló adatbájt. Ilyenkor a félig megérkezett Note On üzenetet elvetjük és átugrunk *Note Off pitch üzenetet vár* állapotba. Az is előfordulhat, hogy Note On pitch üzenetre várva, jön egy olyan státuszbájt, ami se nem Note On se nem Note Off. Ekkor a legutóbb érkezett státuszbájt már nem a Note On, ezért ha jönnek adatbájtok nem dolgozhatjuk fel őket pitch és velocity üzenetekként, vagyis nem szabad olyan állapotban maradnunk, ami adatbájt érkezése esetén feldolgozza azt. Lehet, hogy például azok már egy modulációs csúszka helyzetére vonatkozó adatbájtok, amiknek státuszbájtját se nem Note On, se nem Note Off lévén ignoráltuk. Ezért ilyen esetben átváltunk *státuszbájtot vár* állapotra.

A hibakezelés röviden összefoglalva: Ha adatbájtot vár az állapotgép és se nem adatbájt jött, se nem Note On vagy Note Off státuszbájt, akkor *státuszbájtot vár* állapotba lép. Ezen kívül pedig bármilyen állapotban igaz az, hogy ha érkezik egy NOTE ON vagy NOTE OFF státuszbájt, akkor annak megfelelő pitch üzenetre váró állapotba lép.

A bejövő MIDI üzenetek függvényében hangokat kell elhelyezni a rekeszekben, vagy éppen el kell távolítani őket onnan. Az előbbi esetben valahogyan választani kell neki egy rekeszt, ahová bekerül majd az adott hang, ennek a választási mechanizmusnak a működését tárgyaljuk a következő fejezetben.

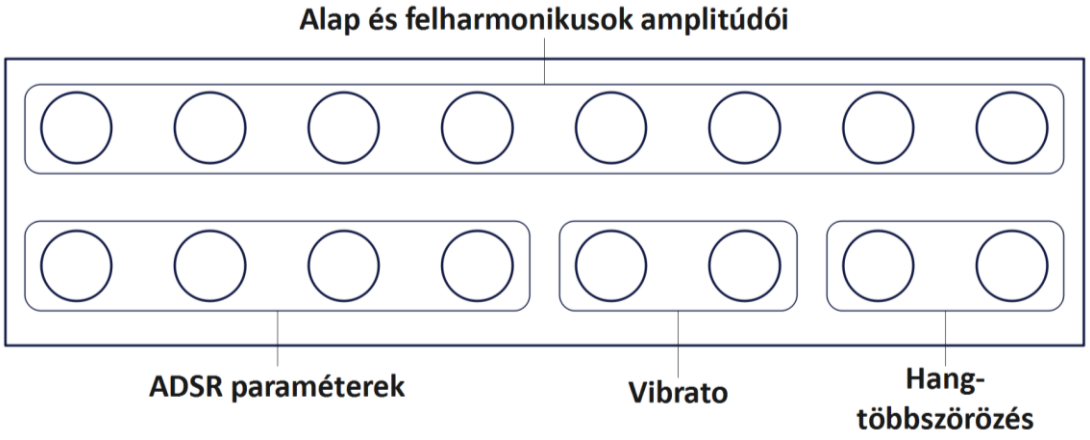
4.4. Funkciók és felhasználói felület

Az szintetizátoron a felhasználó általi információbevitelre a szakdolgozat elkészültekor 16 potenciométer áll rendelkezésre. *(Az egyelőre még nem megvalósított funkciók és jövőbeni továbbfejlesztés során, további nyomógombok, egy grafikus kijelző, egy forgó kvandratúra dekóder hozzáadását, illetve egy menürendszer elkészítését tervezem, ezekkel nagymértékben növelhető lenne a testreszabhatóság.)*

A 16 potenciométer közül

- 8 segítségével állítható be a kiadni kívánt hang hullámformája. A 8 tekerő a jel 1-1 harmonikusának amplitúdóját állítja, 1 az alapharmonikusét és ezen kívül még 7 felharmonikusét.
- 4 potenciométerrel állíthatjuk be az ADSR (attack, decay, sustain, release) paramétereket, melyek a hangok amplitúdójának automatikus lefutását meghatározó görbét határozzák meg.
- 2 további potenciométerrel állíthatjuk a vibrato effekt modulációs mélységét és sebességét,
- 2 pedig a megkettőzött hangpárok közötti amplitúdóban és frekvenciában vett különbségét állítja, ezzel egy egyszerű chorus effektet megvalósítva.

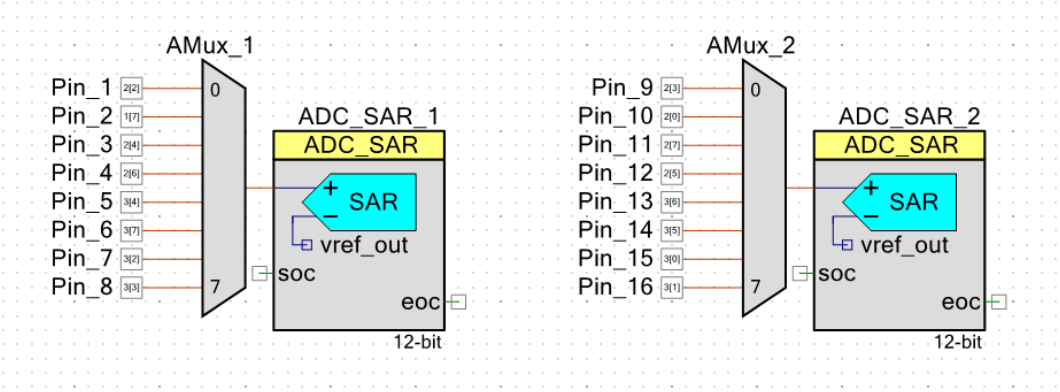
Ezen kívül a szintetizátor el van látva egy MIDI IN, illetve egy 6,35mm-es sztereó jack csatlakozóval a bejövő és a kimeneti jel számára.



ÁBRA 9 KEZELŐI FELÜLET

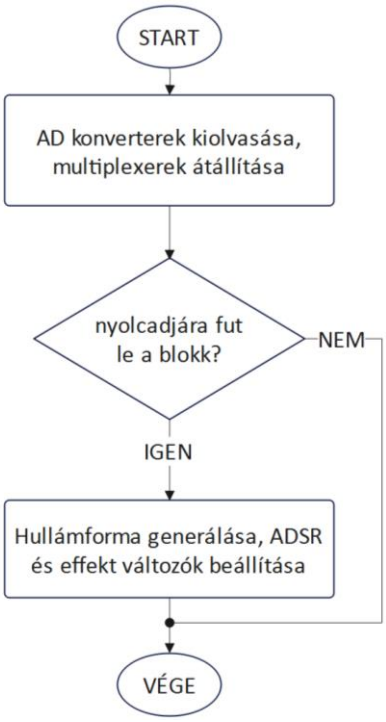
4.5. A felhasználói felületről érkező adatok feldolgozása

A kezelői felületet jelenleg 16 potenciométer képezi, az ezekről származó jelek kiolvasása, illetve feldolgozása a 100 Hz-es hurokban történik. A felhasznált fejlesztőkártyán két SAR (Successive-approximation) A/D konverter található, a 16 potenciométer olvasása során, két analóg multiplexerrel váltunk az A/D konverterekre jutó jelek között.



ÁBRA 10 POTENCIOMÉTEREK OLVASÁSÁHOZ HASZNÁLT HARDVEREK BLOKKJAI PSOC CREATORBAN

Ebben a századmásodpercenként lefutó blokkban egyszerre mindig két potenciométerről származó jelet konvertálunk, majd a multiplexerek segítségével a következő kettő bemenetet választjuk ki. Következő alkalommal mikor lefut ez a blokk, már azokat fogjuk kiolvasni és így tovább. Nyolc lefutás alatt mind a 16 potenciométer értékét kiolvassuk, tehát 100 / 8 = 12,5 Hz frekvenciával frissül a potenciométerek állása. Minden nyolcadik körben ezek alapján az értékek alapján generáljuk a hullámformát, ami a kimenő jel egy periódusát tartalmazza 128 mintában, illetve frissítjük a további változókat, amik az ADSR görbe alakját, valamint a vibrato és chorus effektek paramétereit állítják.



ÁBRA 11 100HZ-ES HUROK FOLYAMATÁBRÁJA

A hullámformát egy olyan 128 elemű tömbben tároljuk, melyben az alapharmonikusnak mindig pontosan egyetlen periódusa szerepel. Az első felharmonikusnak pontosan kettő, a másodiknak három és így tovább, hiszen frekvenciájuk mindig az alapharmonikusénak egész számú többszöröse. Fontos, hogy a lejátszandó hangtól függetlenül generáljuk a hullámformát. Azt, hogy milyen frekvenciával fog egy hang szólni, majd a 40kHz-es hurok számolja, ahhoz pedig ezt a generált hullámformát használja fel oly módon, hogy minden hang pillanatnyi értékét annak fázisa alapján a hullámforma megfelelő pontjai közti lineáris interpolációval határozza meg. Ez az egy periódus, amit itt előállítunk tehát az egy hang által keltett kimeneti jel periódusával egyezik meg. A következő fejezetben ennek a több felharmonikusból álló összetett jelnek a generálását végző függvény működéséről lesz szó.

4.6. A hullámforma előállítása

A hullámformával az alábbi folytonos jelet közelítjük:

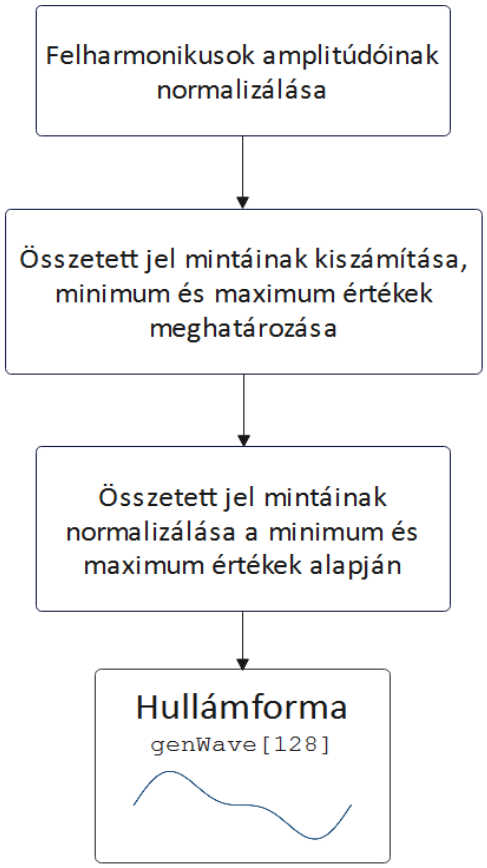
$$U_{hullámforma} = \sum_{i=0}^{7} A_i \cdot \cos(2\pi f_0 i)$$

ahol A_i: az egyes harmonikusok amplitúdó, f_0: az alapharmonikus frekvenciája

Az egyes harmonikusok fázisainak változtatásával ugyan változik a jelalak, mégis az emberi fül számára ez nem igazán észrevehető, így minden komponens azonos fázisban adódik össze, a fázis állíthatóságát elvetettem. A hullámforma mintáit egy függvény számolja ki a potenciométerek által meghatározott harmonikus amplitúdók alapján. Első lépésben megvizsgálja, vannak-e egyáltalán harmonikusok. Ha nincsenek, mert az összes potenciométer le van tekerve, akkor mesterségesen felveszi az alapharmonikust maximumra, igy ekkor az alapharmonikus fog szólni.

A hullámforma kialakításánál, mint ahogy a többi számolási feladatnál is, fontos szempont volt a maximális pontosságra való törekvés. Mivel az alkalmazott processzor nem tartalmaz lebegőpontos aritmetikát, a lebegőpontos számábrázolás megengedhetetlen luxus lett volna. Végeztem erre irányuló kísérleteket, de nem sikerült a szükséges számításokat a kimeneti érték kiszámítására rendelkezésre álló idő alatt elvégezni. Emiatt arra törekedtem, hogy minden számítás megfelelően normált értékekkel történjen, elkerülve a túl kis értékek által okozott pontatlanságot és a túl nagy értékek által okozott túlcsordulást.

Ha vannak értelmezhető felharmonikus amplitúdó értékek, akkor tehát először normáljuk őket, hogy összegük adott érték legyen, így pontosabban generálhatjuk a hullámformát. Ha ezt nem tennénk meg és kis amplitúdókkal számolnánk, amikor a szinuszhullámot tartalmazó táblázatból kinézzük az értékeket, pontosságot veszítenénk és normalizálás után a lehetségesnél „lépcsősebb” jelet kapnánk. Természetesen a komponensek amplitúdóinak normalizálása nem helyettesíti az összetett jel normalizálását ugyanis az összetett jel maximuma kisebb, mint az egyes komponensek amplitúdóinak összege. Emiatt ha megvan az összetett jel, utána azt is mintáról mintára normalizáljuk. Ez tehát egy időigényesebb feladat, mert a program kétszer is végigmegy a hullámforma 128 elemű tömbjén. Első alkalommal több műveletből áll egy lépés, ugyanis ekkor össze kell adni minden körben a 8 harmonikus értékét az amplitúdókkal súlyozva. Második alkalommal (amikor normalizálunk) már csak egy szorzást kell végeznünk minden mintán.



ÁBRA 12 HULLÁMFORMA GENERÁLÁSÁNAK FOLYAMATA

4.7. Kimeneti jel generálása és a 40 kHz-es hurok

A 40 kHz-es hurokban végbemenő események három nagyobb blokkra oszthatók:

- Vibrato effekttel kapcsolatos műveletek,
- a 20 hangrekesz amplitúdóinak manipulálása az ADSR paramétereknek megfelelően, és
- a kimeneti érték kiszámítása a 10 hangrekesz mindenkori állapotai alapján.

4.7.1. Vibrato effekt

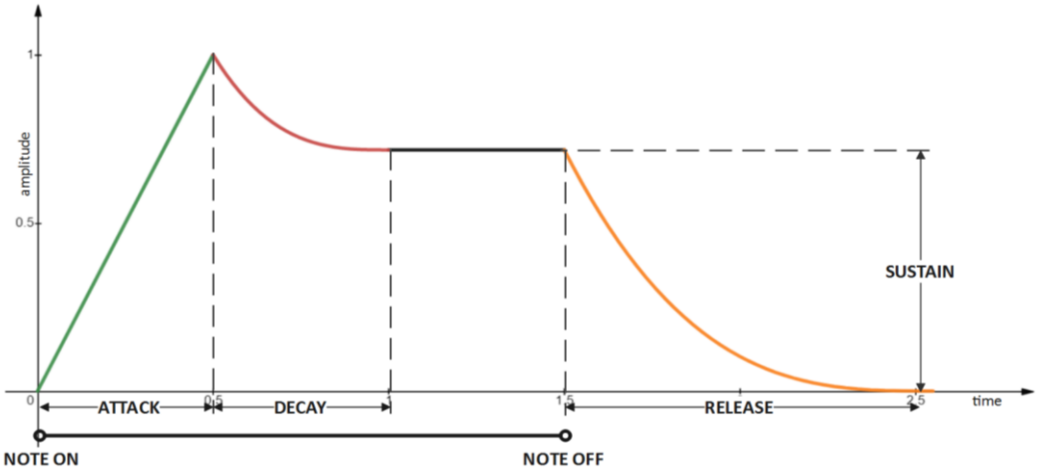
A vibrato effekt tulajdonképpen frekvenciamoduláció. A hang magassága ilyenkor kismértékben időben periodikusan változik, ezzel vibráló hatást keltve. Két változtatható paramétere a moduláció sebessége és mélysége. Mivel 20 hang generálása már eleve igénybe veszi a rendelkezésre álló 25 µs jelentős részét, a vibrato effekt megvalósítására valami nagyon egyszerű módszert kerestem.

Ennek az effektnek a megvalósítása során a kiadott hangok magasságának mintavételi frekvencia függését használtam ki. Amennyiben nagyobb mintavételi frekvenciával adjuk ki ugyanazokat a mintákat, úgy azok nagyobb frekvenciájú jelet fognak eredményezni. A mintavételi frekvencia pedig nem más, mint a 40 kHz-es hurok lefutási gyakorisága, ami

egyszerűen megváltoztatható futás közben is. A kimenet generáló hurok a saját lefutási gyakoriságát meghatározó órának a frekvenciáját állítja be minden körben, egy szinuszhullámnak megfelelően, melynek frekvenciája a moduláló frekvencia, amplitúdója pedig a mélység. Mivel a frekvenciát csak kismértékben kell változtatni, ez nem okoz zavart a működésben.

4.7.2. ADSR görbe

Az ADSR görbe a hang amplitúdójának változását írja le a billentyű leütésétől indulva. A négy betű az angol **A**ttack, **D**ecay, **S**ustain és **R**elease szavak rövidítése. Ezzel a négy paraméterrel adhatjuk meg a görbét, általános alakja a 13-as ábrán látható:



Ábra 13 ADSR görbe

A billentyűleütéstől az amplitúdó maximumáig való felfutás meredekségét az attack paraméterrel lehet szabályozni. A maximum elérése után az amplitúdó hanyatlásba kezd, ennek a csökkenésnek a kezdeti meredekségét állíthatjuk a decay paraméterrel. Ez a csökkenés a sustain paraméter szintjéig tart, ha ezt elérte az amplitúdó, nem változik, egészen addig, míg fel nem engedték a billentyűt, ekkor ugyanis megint csökkenésbe kezd, végül teljesen megszűnik. Ennek a csökkenésnek a kezdeti meredeksége a release paraméterrel szabályozható. Az én megoldásomban az attack szakasz lineáris, a decay és a release pedig közelítőleg exponenciális. Vegyük sorra, mi történik a programban ADSR tekintetben, ha leütünk egy hangot, majd felengedjük!

Mikor megérkezik a Note On üzenet, a korábban részletezett módszerrel keresünk az új hangnak egy rekeszt, és beállítjuk az arra vonatkozó MIDI amplitúdót a leütés erőssége alapján, illetve az ADSR állapotát ATTACK-ra. Az amplitúdót, amivel ténylegesen szólni fog a hang a kimenetet generáló 40kHz-es hurok állítja be folyamatosan.

Amennyiben egy hangrekesz ATTACK állapotban van, a 40kHz-es hurok bizonyos számú lefutásonként 1-gyel megnöveli annak amplitúdóját. Ezt egészen addig csinálja, míg az el nem éri a billentyűleütés erőssége alapján meghatározott MIDI amplitúdót. A felfutás sebessége attól függ, hány ciklusonként növel, ezt pedig az attack változó határozza meg, amit a 100 Hz-es hurok frissít mindig a megfelelő potenciométerről származó jel alapján. Itt jelenik meg az ADSR számlálók szerepe, ugyanis ezeknek értékét a 40kHz-es hurok minden alkalommal, amikor lefut, megnöveli 1-gyel. Ha eléri a növelési, vagy később a decay és release állapotok alatt csökkentési

küszöböt, akkor a számlálót lenullázza, és végrehajtja a növelést, vagy csökkentést. Tehát egy potenciométerrel szabályozható, a hang felfutási ideje.

Felmerülhet az olvasóban, hogy ezzel az eljárással nem valósíthatunk meg akármilyen gyors felfutást, hiszen 25 µs-onként fut le a 40kHz-es hurok, ha minden körben történik is növelés, akkor is mire elérjük a MIDI amplitúdó 400-as nagyságrendben lévő értékét, eltelik valamennyi idő. 400 ciklus 10ms (azaz egy század) másodperc alatt zajlik le, ez az idő mindenképpen szükséges a felfutáshoz. Ez még mindig nagyon rövid idő, teszteléseim során úgy tapasztaltam, hogy ez teljes mértékben alkalmas hirtelen megszólaló, vagy pukkanó hatású hangok kiadására. Ha ennél gyorsabban szólal meg egy jel a kimeneten, apró pattanást hallhatunk, ami mindenképpen elkerülendő valamilyen fokozatos hangosodás segítségével. Amíg nem volt megvalósítva az ADSR görbe követése, ez a pattanás kellemetlen hatást keltett, mióta már mindig van egy minimális felfutási és lefutási idő, ez a probléma már nem jelentkezik.

Amennyiben elérte a hangrekesz amplitúdója a MIDI amplitúdót, a 40 kHz-es hurok átállítja a rekeszt DECAY állapotba. A decay és a release szakaszok az attack-tól eltérő módon több különböző fázisból állnak, és ezekben a fázisokban különböző a csökkenési meredekség, ezeken a szakaszokon töréspontos görbén halad az amplitúdó. Ha egy hangrekesz DECAY állapotban van, akkor a felfutáshoz nagyon hasonló módon először bizonyos ciklusszámonként 1-gyel csökkentjük a hang amplitúdóját. A kezdeti meredekséget ugyanúgy egy potenciométerrel állítható változó határozza meg, a billentyűleütés feldolgozásának pillanatában. Ez a lineáris csökkenés addig tart, míg az amplitúdó el nem éri a sustain szinttől való kezdeti távolság felét. Az első fázisban ezt az amplitúdó szintet (D_f) az alábbi módon kapjuk:

$$D_f = S + \frac{A_{MIDI} - S}{2}$$

ahol S: sustain szint és A_{MIDI}: MIDI amplitúdó.

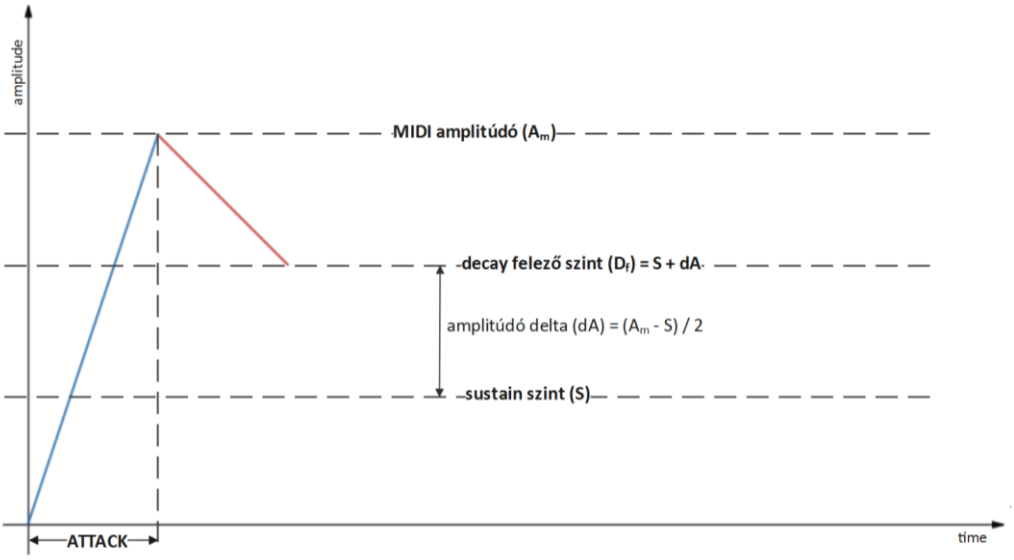
Vezessük be a MIDI amplitúdó és a sustain különbségének felére egy új változót, legyen

$$dA = \frac{A_{MIDI} - S}{2}.$$

Így az előző képletünk az alábbira egyszerűsödik:

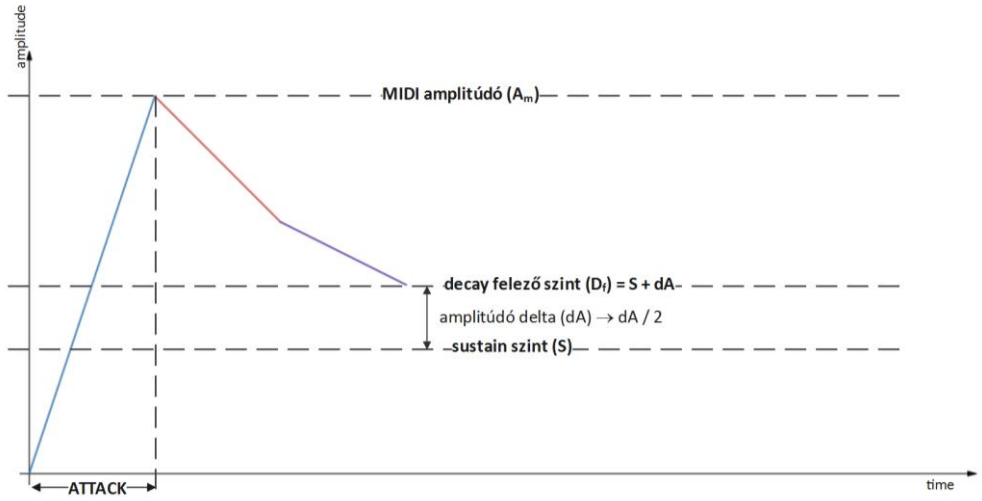
$$D_f = S + dA$$

Fontos, hogy ez a delta amplitúdó érték (dA) és természetesen ezzel együtt a decay meredekség felező szint D_f is csak az első fázisban számolható a fenti képletekkel, később ezek minden fázisváltáskor megváltoznak.



ÁBRA 14 DECAY SZAKASZ AZ ELSŐ TÖRÉSPONT PILLANATÁBAN

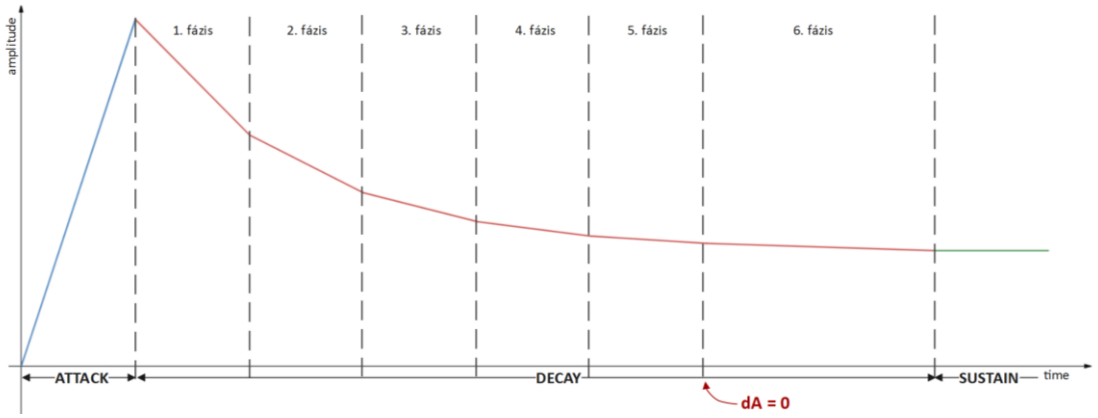
A fenti ábrán látható, hogy eddig a pontig a decay szakaszon ugyanaz történik, mint az attack szakaszon, csak visszafele. Most azonban, hogy a decay felező szinten van az amplitúdó, a 40kHz-es hurok felére csökkenti a csökkenési meredekséget (kétszer annyiadik ciklusonként történik csak csökkentés). Ekkor a csökkentési meredekséget megadó változó elszakad a potenciométerről származó értéktől. Ezzel egyidőben a delta amplitúdó (dA) értéke is felére csökken, vagyis zajlik tovább a csökkenés immáron fele akkora sebességgel, fele akkora „úton”.



ÁBRA 15 DECAY SZAKASZ A MÁSODIK TÖRÉSPONT PILLANATÁBAN

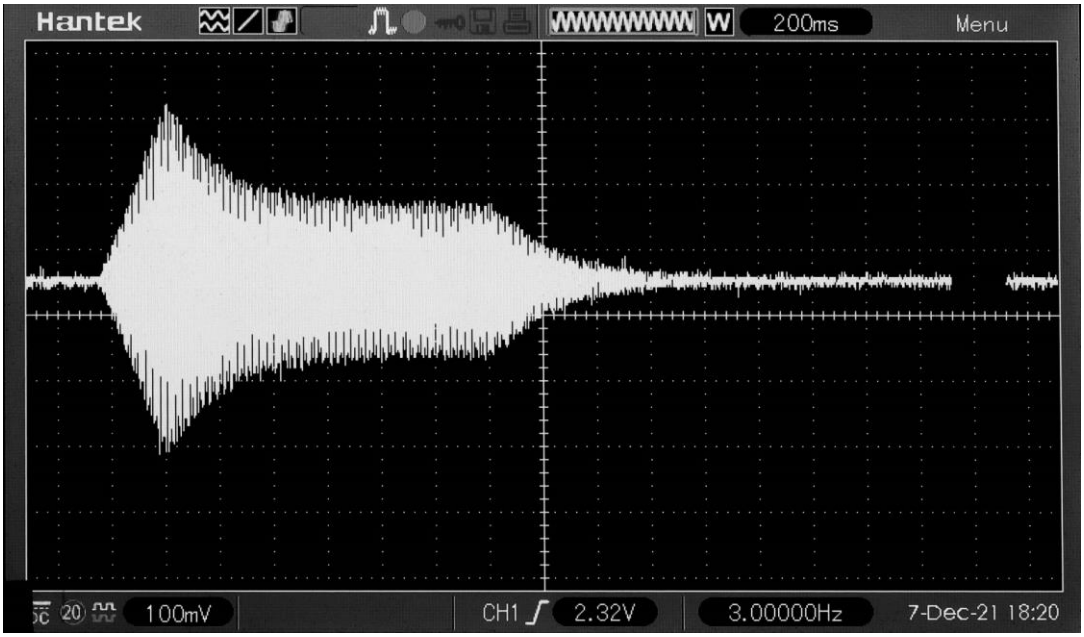
Ez a meredekségváltás újra és újra megtörténik, amikor elérjük a mindig egyre kisebb mértékben előrébb helyezett decay felezési szintet. Eredményül, adott időközönként felére csökken a görbe meredeksége, ami bár töréspontos, mégis így exponenciális jelleget kap. Mivel a dA változó fixpontos adattípusban van eltárolva, a kettővel való osztás jobbra léptetést jelent (Kerekítés nem történik). Amikor az utolsó bitet is kiléptetjük, dA nulla lesz, ami azt jelenti, hogy a következő meredekség felezés a sustain szinten lenne, azonban amikor ezt eléri az amplitúdó, a 40kHz-es hurok átállítja a hangrekesz állapotát SUSTAIN-ra. SUSTAIN állapotban nem változik az amplitúdó, azon a konstans értéken marad, amin éppen van és ami a hangrekeszre

meghatározott sustain szint. (A sustain szintet a hang leütése pillanatában érvényes érték határozza meg.) Az amplitúdók felbontásából fakadóan a dA változó biztosan kevesebb, mint 9 jobbra léptetés után 0 lesz, ami azt jelenti, hogy ennél több töréspont nem lehet. Miközben dA minden fázisváltáskor felére csökken, addig az a határérték ameddig az ADSR számláló számlál két amplitúdó csökkentés között, minden fázisváltáskor kétszeresére nő. Fontos, hogy ezt a változót elegendő méretű adattípusban tároljuk ahhoz, hogy 9 balra léptetés ne okozhasson túlcsordulást.



ÁBRA 16 DECAY SZAKASZ FÁZISAI

Változást az adott hangra vonatkozó Note Off üzenet hozhat, ugyanis ekkor a MIDI üzenet feldolgozó hurok a hangrekeszt RELEASE állapotba állítja, ami szinte egy az egyben úgy működik, mint a decay szakasz, csak itt már a 0 az alsó cél.



ÁBRA 17 KIMENETI JEL ADSR GÖRBÉJÉNEK MÉRÉSE

4.7.3. A kimeneti érték számítása

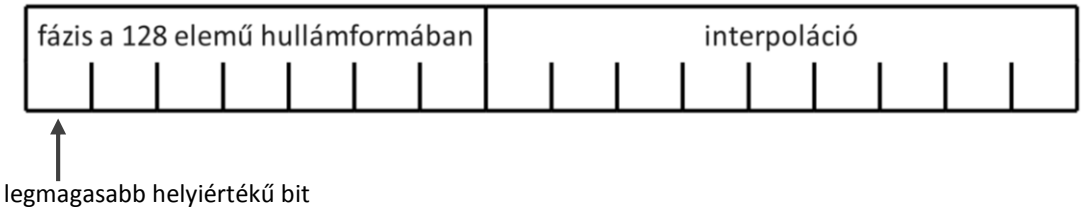
A fázisban mindig a kiadni kívánt hang frekvenciájának megfelelő fázistolással haladunk előre. A fázislépés és a hang frekvenciája közti összefüggés az alábbi:

$$d\varphi = \frac{2^{16} f}{f_s},$$

ahol $d\varphi$: a fázislépés, f: a hang frekvenciája és f_s: a mintavételi frekvencia.

A hangok fázisát 16 biten ábrázoljuk, tehát 65536 mintából áll az a jelperiódus, melyből fázislépésekben haladva vesszük a mintákat és vezetjük a kimenetre. Ennek a periódusnak 128 pontját rögzíti a generált hullámforma, ami azt jelenti, hogy minden 512. mintát. Ezek között lineáris interpolációval számítjuk az értékeket, ezt végzi el ez a programblokk.

A 16 bites fázis felépítése a következő:



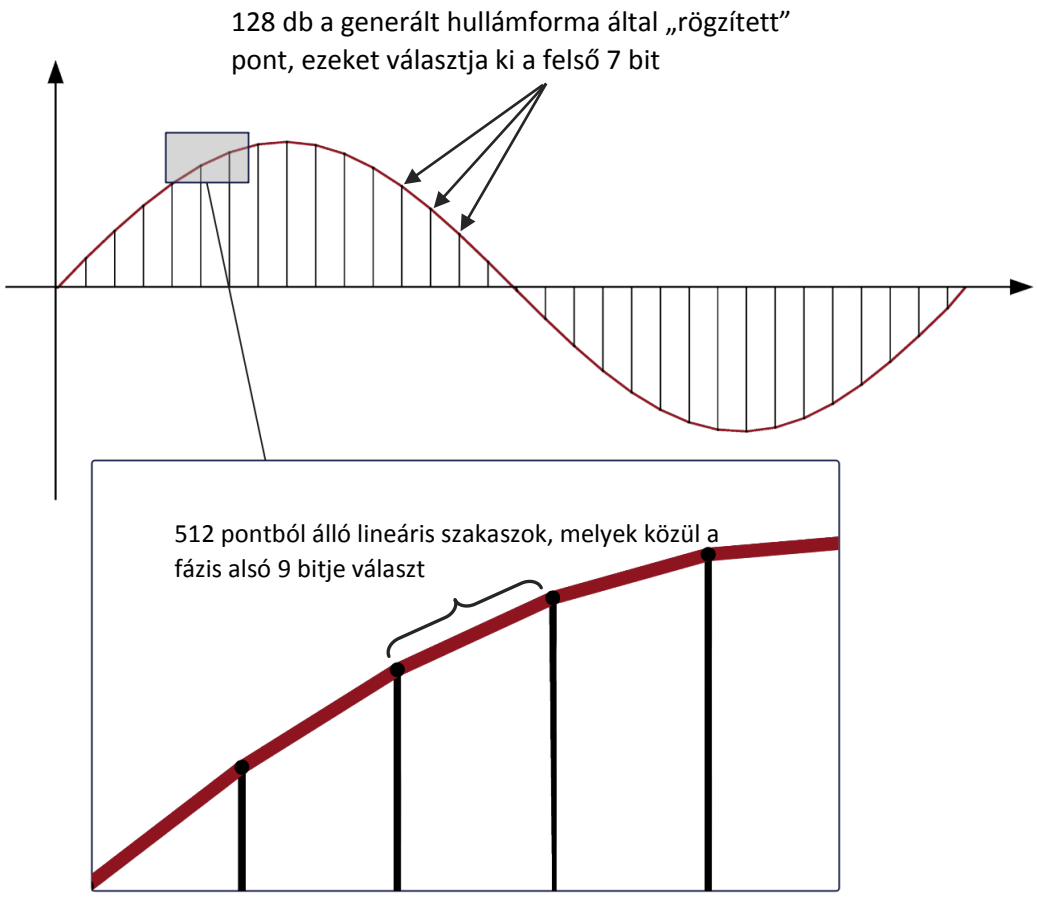
ÁBRA 18 16 BITES FÁZIS FELÉPÍTÉSE

A felső hét bit határozza meg azt, hogy a 128 elemű generált hullámforma tömbben éppen hol tart a 16 bites fázis, az ezen minták közötti lineáris interpolációt pedig az alsó 9 bit alapján számítjuk.

Egy hangrekesz kimeneti értékét úgy kapjuk, hogy a jelenlegi fázisból a felső 7 bit alapján meghatározzuk a lefele szomszédos a generált hullámforma által rögzített pontot, ehhez 1-et adva meghatározzuk a felfele szomszédos ilyen pontot, majd az alsó 9 bit alapján elvégezzük a két pont között a lineáris interpolációt. Az így kapott rekesz jeleket mind a 20 rekeszre kiszámítjuk és összegezzük, így kapjuk a kimeneti jelet.

Mivel ebben a hurokban nagyon fontos a számítási idővel való gazdálkodás, ebben a megszakítási rutinban kizárólag kettő hatványaival történik osztás, ezek az osztások ugyanis bitléptetésekkel elvégezhetők, ezek pedig gyorsan végrehajtható műveletek a processzor számára. A nagyobb pontosság érdekében a léptetés előtt kerekítés történik.

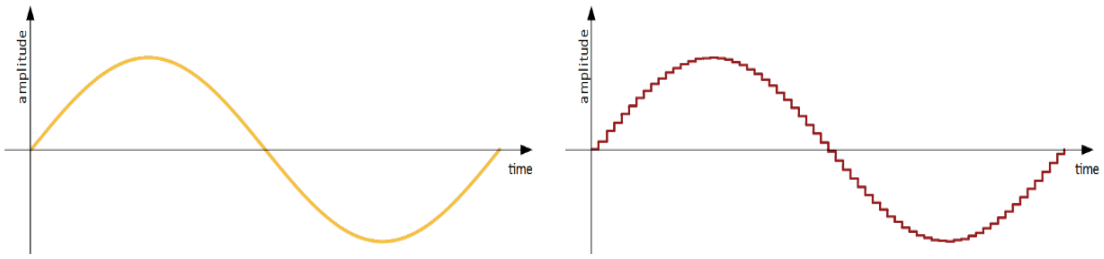
Sajnos igen nehéz olyan ábrát készíteni, melyen egy 128 mintából álló szinuszhullámon egymástól elkülönülnek a minták, ezért a jobb átláthatóság érdekében a következő ábrán, a képen látható mintaszám 128 helyett csupán 32, ez ne tévessze meg az olvasót, amikor a 19-es ábrát tekinti!



ÁBRA 19 A 128 MINTA KÖZÖTT LINEÁRIS INTERPOLÁCIÓVAL KAPOTT 16 BITEN ÁBRÁZOLT JELALAK FELÉPÍTÉSE

4.8. DA konverzió a kimeneten

Egy szintetizátor működése során talán a legfontosabb feladat a később majd erősítőre vezetendő „analóg” jel előállítása. Digitális szintetizátorok esetén, a kimeneten mindenképpen D/A konverziót szükséges végezni. A D/A konverzió során valamilyen mintavételi frekvenciával, valamilyen felbontással állítunk elő egy kvantált kimeneti jelszintet.



ÁBRA 20 ANALÓG ÉS MINTAVÉTELEZETT JEL

A tervezés során eldöntendő paraméterek, egyrészt a mintavételi frekvencia, másrészt a D/A konverter felbontása. Olyan értékeket kell választani ezekre, melyek esetén mondhatjuk, hogy a kimeneten kellő pontossággal megközelíthető a kiadni kívánt a szintetizátor belsejében „megálmodott” analóg jelalak.

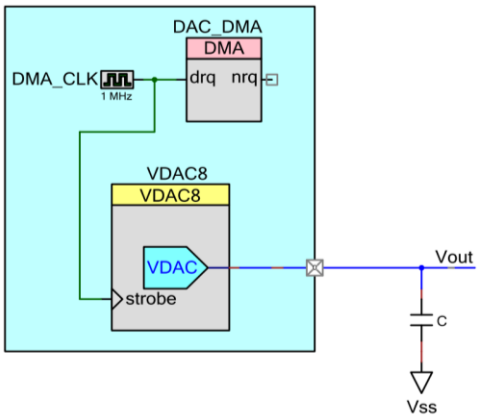
Shannon mintavételi törvénye szerint egy sávkorlátos jel hiba nélkül előállítható mintáiból, amennyiben a mintavételezési frekvencia legalább a jel sávszélességének kétszerese. Mi emberek nagyjából 20 kHz frekvenciáig hallunk hangokat, ennek kétszerese 40 kHz, én ezt választottam mintavételi frekvenciának. Ez valamivel alacsonyabb az iparban használt (pl. CD hangrögzítés) 44,1 kHz-nél azonban így is jóval magasabb, mint a kiadni szándékozott hangok alapfrekvenciáinak kétszerese. Később esetleg megfontolandó a mintavételi frekvencia növelése, azonban a zongorabillentyűzeten előforduló hangok frekvenciáit és a processzor erőforrásait is számításba véve a 40 kHz jó kompromisszumnak tűnt.

Ami a kimeneti jel felbontását illeti, a chipben 8 bites D/A konverterek vannak, ami nem lenne elegendő felbontású a tervezett minőségű hangzáshoz, ezért a D/A felbontás növelésére van szükség. Ennek lehetséges módjaira, és a választott módszer bemutatására a következő fejezetekben térek ki.

Lehetséges módok DAC felbontásának javítására

A D/A konverter felbontásának növelésének lehetőségeit a Cypress AN64275 alapján röviden összefoglalom az alábbiakban. Az alkalmazott proccesszor egyaránt rendelkezik áram, valamint feszültség D/A konverterrel is (továbbiakban IDAC és VDAC). Az IDAC előnye, hogy a kimeneti feszültségtartományt csupán egy ellenállás méretének megválasztásával határozhatjuk meg, míg egy VDAC esetében ehhez feszültségosztóra lenne szükség. Természetesen a kimeneten megjelenő feszültség nem haladhatja meg a *tápfeszültség – maradékfeszültség ($V_{compliance}$)* határt, ami esetünkben 4V, ez megegyezik a VDAC nagyobb feszültségtartományának felső határával..

A *Dithered Output DAC* (DVDAC) eljárás lényege, hogy gyorsan váltogatjuk a kimenetre írt értéket két szomszédos érték közt, a kimenetet megszűrve, így ott a gyorsan váltogatott értékek átlaga jelenik meg. Mivel a váltogatott értékek közti különbség csupán 1 LSB, a kimeneti zaj kicsi lesz, a kvantálási zajhoz hasonló. A gyors kimeneti értékek közötti váltást a processzor számítási kapacitásainak leterhelésének elkerülése végett érdemes DMA segítségével megvalósítani. Kis hátránya, hogy mivel a kimenet két a 8 biten eredetileg ábrázolható érték közötti lehet, az ábrázolási tartomány tetjén elveszik $2^{N-8} - 1$ bit, ahol N: a kibővített DAC felbontása bitekben.



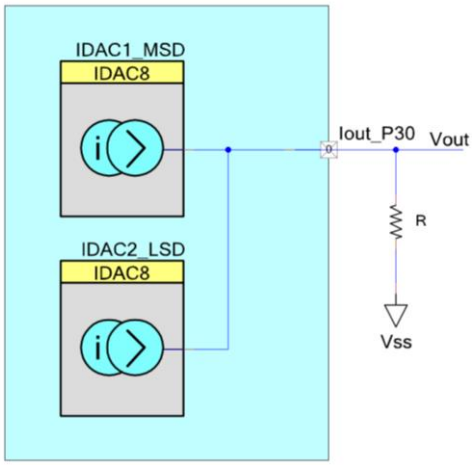
ÁBRA 21 DVDAC BLOKKVÁZLAT (FORRÁS: CYPRESS AN64275)

Egy másik megoldás a parallel IDACs Method. Ennél a megoldásnál két IDAC párhuzamosan vannak kötve, így a kimeneti áram a kettő áramának összege lesz. Az IDAC-ok kimeneti áramtartományait a 21-es ábrán látható táblázat szerint választhatjuk meg. Ha például az egyik IDAC kimeneti tartományának a 0 – 2048 µA tartományt, a másiknak pedig az 0 – 256 µA tartományt választjuk, akkor 11 bitre növelhető a kimeneti felbontás, amennyiben a kimeneti érték felső 8 bitjét az előbbi IDAC-ra vezetjük, alsó 3 bitjét pedig az utóbbira. A módszer alkalmazásának kritikus pontja a két IDAC együttfutása, mivel ennek hiányában az egymást követő bináris értékekhez tartozó kimeneti feszültségkülönbségek egyenletlenné válnak, a differenciális hiba megnövekszik. 11 bit felbontás esetén ez még 1 LSB alatt tartható.

	Ranges		
µA	**2048 µA**		
1024	7		
512	6		
256	5	**256 µA**	
128	4	7	
64	3	6	
32	2	5	**32 µA**
16	1	4	7
8	0	3	6
4		2	5
2		1	4
1		0	3
0.5			2
0.25			1
0.125			0

ÁBRA 22 IDAC-OK VÁLASZTHATÓ KIMENETI TARTOMÁNYAI (FORRÁS: CYPRESS AN64275)

A 0-2048 és a 0-32 µA tartományt kombinálva elvileg 14 bites felbontást kapnánk, azonban ebben az esetben a differenciális hiba nagymértékű megnövekedése miatt gyakorlatilag nem lesz jobb a felbontás.



ÁBRA 23 IDAC BLOKKVÁZLAT (FORRÁS: CYPRESS AN64275)

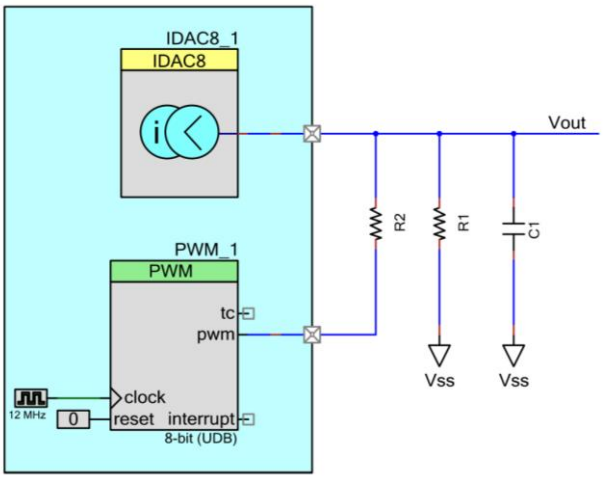
A felbontás növelésére lehetséges PWM jellel modulált IDAC (MIDAC) alkalmazása is. Ekkor a 23-as ábra szerinti kapcsolásban egy IDAC és egy impulzus szélesség modulátor (PWM modul) van összekapcsolva. A kimenet és a föld közti R1 ellenálláson az IDAC és a PWM modul által generált áram összege fog átfolyni. A PWM modul és az IDAC kimenet közt lévő ellenállás jóval nagyobb, mint az IDAC kimenet és a föld közötti, így a PWM modul csak kismértékben, a legkisebb helyiértékű biteknek megfelelően befolyásolja R1 áramát. Először R1 értékét határozzuk meg, annak függvényében, hogy milyen nagyságú kimeneti jelet akarunk előállítani.

$$R1 = \frac{V_{max}}{I}$$

Ezután meghatározzuk R2 értékét.

$$R2 = R1\frac{V_{dd} + (V_{max} \,/\, 256)}{V_{max} \,/\, 256}$$

A kis helyiértékű biteket a PWM modulra, a nagyobb helyiértékűeket az IDAC-ra vezetve, hasonló megoldást kapunk, mint a DVDAC esetében, azzal a különbséggel, hogy a PWM frekvencia nagyobb lehet, így a keletkezett zajt könnyebb kiszűrni. Ugyanakkor hátrányos, hogy R1 és R2 viszonya kritikus az előállított jel szempontjából, így R1 és R2 beállítása nagy pontosságot igényel.



ÁBRA 24 MIDAC BLOKKVÁZLAT (FORRÁS: CYPRESS AN64275)

A negyedik lehetőség a *visszacsatolt PIDAC* eljárás. Mint a PIDAC (visszacsatolás nélküli) megoldás leírásában említettem, ha a párhuzamosan kapcsolt IDAC-ok kimeneti tartományában nagyobb az eltérés, akkor lehet ugyan nagyobb kimeneti felbontást elérni, de a nagy differenciális hiba miatt ez nem igazán ér semmit. Ha azonban egy A/D konverterrel visszacsatoljuk a jelet, akkor a kimeneti hiba korrigálható, és akár 14 bit pontosság is elérhető. A visszacsatolás miatt azonban jelentősen megnő a konverzióhoz szükséges idő, így ez a megoldás a mi esetünkben nem használható.

A fenti lehetőségek közül a DVDAC és a PIDAC megoldásokat vizsgáltam meg. Mindkét esetben hasonló eredményeket kaptam, a PIDAC megoldást választottam, mert ebben az esetben a kimeneti jel amplitúdójának beállítása egyszerűbb és a tesztrendszer zaja is kisebb volt, mint a DVDAC esetében.

5. Eredmények és tesztelés

5.1. Eredmények

A szakdolgozat elején a terveimről beszélve sok dolgot megemlítettem lehetőségként, ezek közül az alábbiak valósultak meg:

- A szintetizátor meghajtható a MIDI szabványnak megfelelő billentyűzettel,
- Potenciométerek segítségével beállítható
    - a hangszín az alapharmonikus és 7 felharmonikus erősségének szabályozásával,
    - egyes hangok esetében az amplitúdó automatikus alakulása ADSR paraméterekkel,
    - a vibrato effekt modulációs sebessége és mélysége
    - Egyszerű chorus effektet megvalósító megkettőzött hangok finom elhangolása és hangossága
- A kimeneten 11 bites felbontásban, 40kHz mintavételi frekvenciával jelenik meg az analóg jel.

Ezen állapotában a szintetizátor leginkább elektromos orgonák hangszínéhez hasonló hangot tud kiadni (ADSR funkcióval kibővítve), azonban a szinuszjel, melyet felhasználunk a felharmonikusok összegzésével a hullámforma kialakításához, bármikor nagyobb nehézségek nélkül lecserélhető a programban valamilyen „izgalmasabb” jelre.

A továbbfejlesztés során mindenképpen abba az irányba indulnék el, hogy a hullámforma generálás során több lehetőségünk legyen, akár lehetnének az egyes felharmonikusok különböző jelalakúak. Mindenképpen szeretném a hangokat magasságuktól függően eltérő hullámformákból létrehozni. Ez a későbbiekben tárgyalt frekvenciatartománybéli átlapolódás megszüntetése és nemkívánt „fantom” hangok megjelenése szempontjából is, ugyanakkor az általános zenei változatosság és kellemetlen magas hangok elkerülése szempontjából is hasznos lenne. Emellett érdemes lehet kedvenc hangszíneink elmentését lehetővé tenni, hogy azokat bármikor egy gombnyomással beállíthassuk. Ehhez a szintetizátor működéséből fakadóan elég lenne nagyon kevés adatot elmenteni az EEPROM-ba, hiszen a kimenő jelet egyértelműen meghatároznak a felhasználó által állítható paraméterek értékei. Ez jelen esetben a 16 potenciométerről származó 2 bájtos adattípusokban tárolt adat, vagyis összesen 32 bájt, de a későbbiekben tervezett, időtartománybeli töréspontos jel megadás esetében sem sokkal nagyobb érték. A funkciók bővítését nyomógombok, menürendszer, egy kisebb grafikus LCD kijelző és egy forgó kvadratúradekóder által megvalósított univerzális digitális tekerő segítségével tervezem.

5.2. Tesztelés

A szintetizátor működése során kulcsfontosságú a beérkező MIDI üzenetek hibátlan feldolgozása. Ezen funkció megfelelő működését extrém mértékű MIDI üzenet terheléssel vizsgáltam. Az adatfolyamot számítógépes program segítségével állítottam elő, zenészek játékát bőven felülmúlva az időegység alatti üzenetküldésben. A fejlesztés során hosszabb ideig fennakadást okozott, hogy ilyenkor nem a várt működést tapasztaltam, néha beragadtak

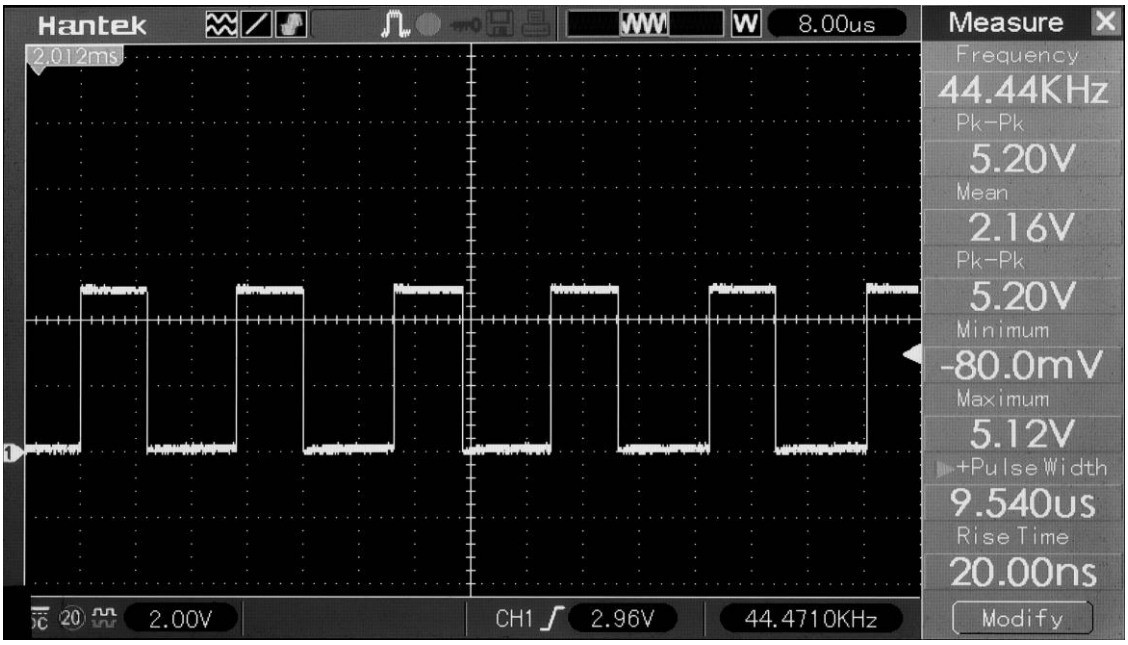
hangok, nem is leütött mély vagy éppen magas, a zongora billentyűinek tartományába nem is tartozó hangok szólaltak meg. Ahogy korábban erről már írtam, a megszakítások prioritásainak megfelelő megválasztásával ez a probléma megszűnt.

A különböző MIDI eszközök kommunikációjában való eltérések miatt megpróbáltam a szintetizátort minél több eszközzel kipróbálni. Kipróbáltam a két különböző MIDI billentyűzettel itthon, melyekről tudom, hogy az egyik mindig küld státuszbájtot, a másik viszont csak ha az nem egyezik meg az előzővel. Az viszont közös bennük, hogy a Note Off üzeneteket mindkettő Note On státuszbájttal és 0 leütési erősséggel jelzi. Ebben azonban eltér tőlük a számítógéppel küldött MIDI jel, ugyanis ezesetben billentyű felengedésekor valóban Note Off státuszbájt érkezik. Ez a három eszköz tehát a számunkra lényeges lehetséges kommunikációs alternatívák közül mindegyikre ad példát, a fent tárgyalt állapotgép minden esetben megfelelően kezeli a bejövő adatokat. Ezeken kívül kipróbáltam egy a tanszéken található MIDI billentyűzettel is, azzal is megfelelően működött, azonban annak MIDI üzenet küldési stratégiáit nem vizsgáltam.

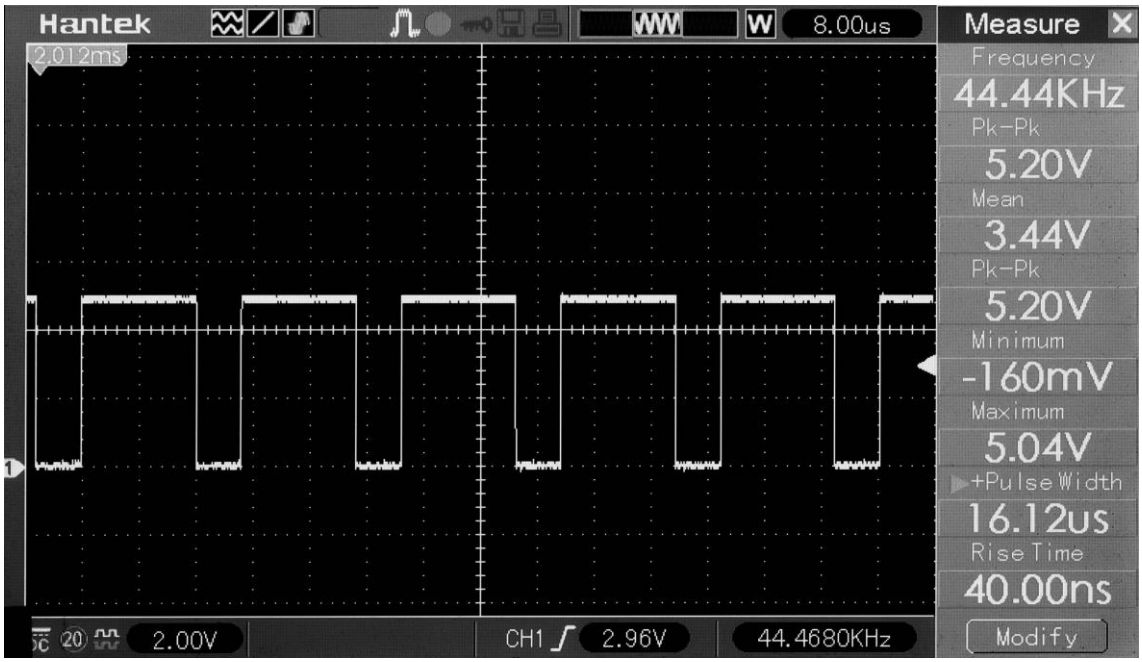
Jelenlegi működés során számítógéppel hatalmas mértékű üzenetmennyiség küldése esetén megtörténik, hogy nem sikerül időben megszólaltatni és elhallgattatni a hangokat, azonban a szoftveres pufferbe mindig átmentődnek a bejövő üzenetek. Ilyenkor azt tapasztalhatjuk, hogy az üzenetáradat hirtelen megszűntetését követően rövid ideig még jelennek meg hangok és szűnnek meg, ahogy érvényüket kifejtik a szoftver pufferben feltorlódott üzenetek. Ez azonban egyáltalán nem életszerű állapot, így azt gondolom bátran kijelenthetem, hogy ha akár maga Liszt Ferenc saját magával négykezesezne is a szintetizátoron, a leütött hangok kivétel nélkül megjelennének a kimeneten. (Persze a 10-es polifónia szám miatt lehet, hogy ráncolnák a szemöldöküket…)

Egy másik kritikus metszete a működésnek a 40 kHz-es kimeneti jelet generáló hurok futási ideje. Ez a megszakítási rutin 25 μs periódusidőnként fut le, ezalatt az idő alatt kell a vibrato effektet, és az ADSR görbe mentén való amplitúdókorrekciókat elvégezni, majd a 20 hangrekesz mindegyikéből kiszámítani a kimeneti értéket. Az ehhez szükséges idő mérése céljából egy lábra kivezettem egy digitális jelet, melyet a hurok elején kiadunk, a végén pedig visszaveszünk. Ezt a jelet oszcilloszkópon figyelve mérhető a hurok lefutásának hossza.

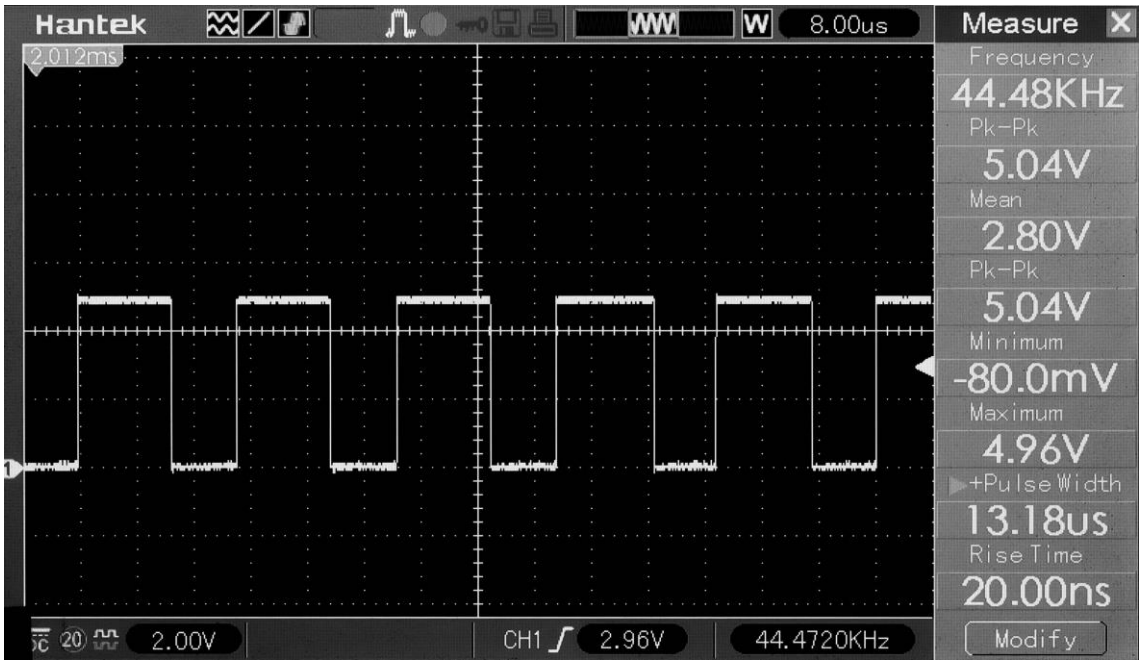
Az alábbi ábrákon látható ennek a mérőjelnek a képe először 0 hang leütése esetén (25-ös ábra), másodszor maximum terhelés esetén (10 hang le van nyomva)(26-os ábra), végül pedig átlagos játék során (27-es ábra):



ÁBRA 25 40KHZ-ES HUROK KITÖLTÉSI TÉNYEZŐJE HA NEM SZÓL HANG



ÁBRA 26 40KHZ-ES HUROK KITÖLTÉSI TÉNYEZŐJE MAXIMUM TERHELÉS ESETÉN



ÁBRA 27 40KHZ-ES HUROK KITÖLTÉSI TÉNYEZŐJE ÁTLAGOS JÁTÉK KÖZBEN

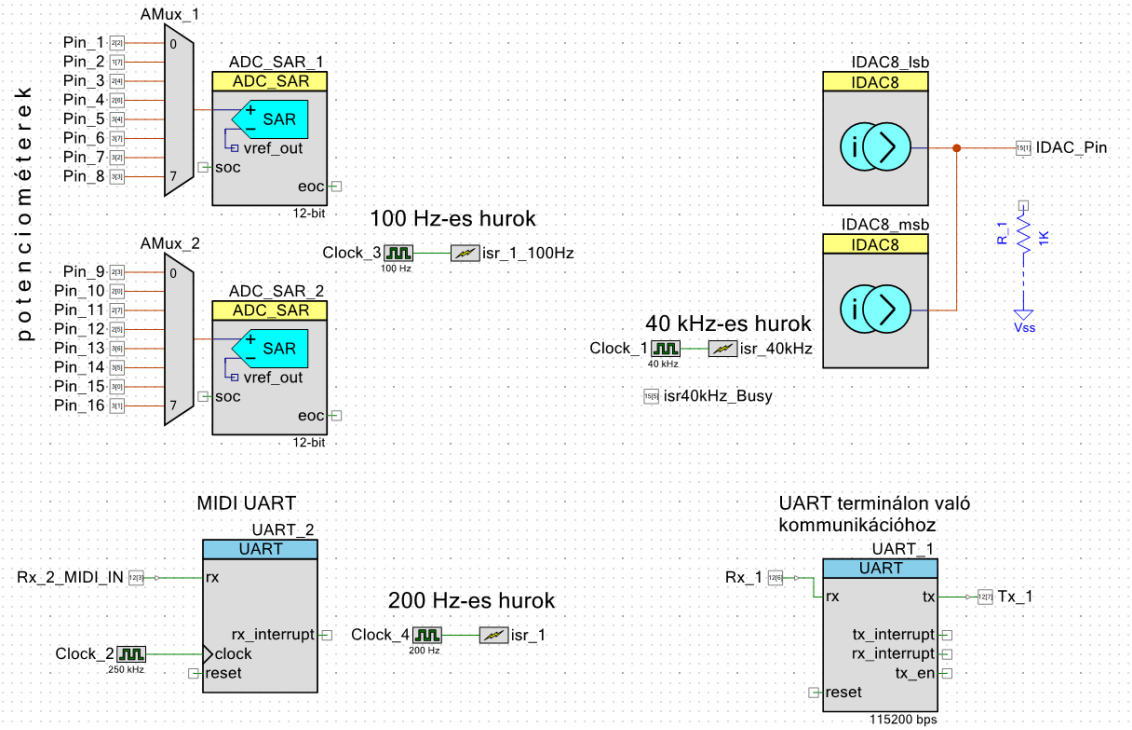
6. Összefoglalás

Összefoglalva a működést tehát a program 3 nagy blokkra bontható. Ezek a blokkok 100, 200, illetve 40.000 Hz frekvenciával futnak le.

A 100Hz-es blokkban történik a felhasználói felületről érkező információk feldolgozása és bevitele a rendszerbe. Ez a blokk generálja a potenciométerekről bejövő adatok alapján a kimeneti hullámforma egyetlen periódusát, melyet 128 mintában tárol el.

A 200Hz-es blokkban dekódoljuk a bejövő MIDI üzeneteket, illetve azoknak megfelelően módosítjuk a hangrekeszeket, melyekben a mindenkor szóló hangokat tároljuk. Ezekből a hangrekeszekből összesen 20 darab van, melyből azonban csak az első 10 helyre helyezhetünk a MIDI csatornán érkező hangot, a második 10-et automatikusan töltjük fel úgy, hogy minden hangot megkettőzünk. A „klón” hangokat egy oktávval eltoljuk és kismértékben elhangolhatjuk, így valósítva meg egy egyszerű chorus effektet a kimeneten.

A 40 kHz-es hurokban történik az ADSR paramétereknek megfelelő amplitúdóváltoztatás az egyes hangrekeszeken, a vibrato effekt megvalósítása magának a 40kHz-es hurok frekvenciájának kismértékben való periodikus módosításával, végül pedig a kimeneti jelérték kiszámítása, majd két párhuzamosan kapcsolt IDAC segítségével 11 bites felbontásúra bővített áramértékké konvertálása. A PSoC Creator grafikus felületén a projektben felhasznált blokkok vázlata a 24-es ábrán látható.



ÁBRA 28 PROJEKT MUNKATERÉBEN SZEREPLŐ MODULOK PSOC CREATORBAN

6.1. Tanulságok

6.1.1. 40 kHz-es minta generálási frekvencia következményei

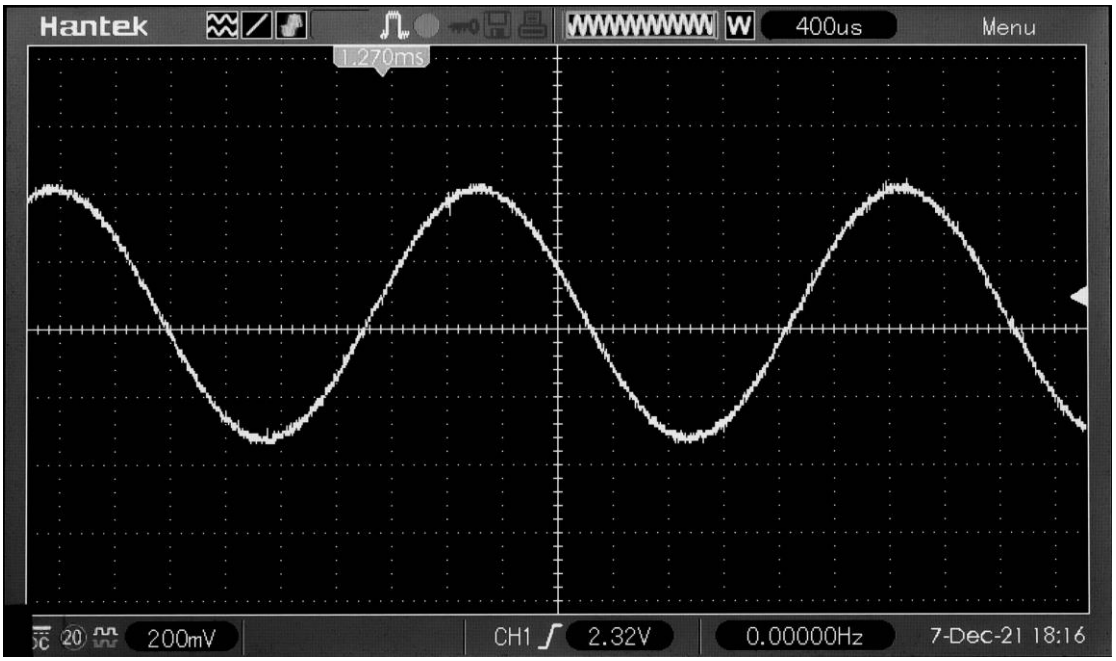
40 kHz-es mintavételi frekvencia mellett a „bemenő jel” frekvenciájának maximális értéke 20 kHz. A legmagasabb előállítani kívánt hang alapfrekvenciája 4185 Hz. Látható, hogy ennek a hangnak az 5. felharmonikusa már nem fér bele ebbe a sávkorlátba. Amennyiben az 5-7. felharmonikusok amplitúdója nagy, a kisebbeké pedig kicsi, akkor a zongora legmagasabb billentyűinek leütésekor megszólalnak „fantom” hangok is, melyeknek frekvenciája a billentyűkkel ellentétes irányban halad. Ez azzal magyarázható, hogy a 20 kHz feletti hangok esetén a spektrumban megjelennek a mintavételi frekvenciáról „visszatükrözött” komponensek, melyek minél magasabb hangot adunk ki, annál alacsonyabb frekvenciákra tolódnak. Érdekes élmény volt ily módon megtapasztalni a mintavételi tétel gyakorlati megnyilvánulását. Meg kell jegyezni, hogy ez a jelenség csak olyan beállítások esetén hallható, melyeknél az 5-7. felharmonikusok kivételével minden más harmonikus és az alapharmonikus is 0. Ezek pedig nem életszerű beállítások, kellemetlen visító hangszínt eredményeznek. Emiatt ezt a problémát nem tekintettem azonnal megoldandónak, azonban két lehetséges megoldásom van a jövőre nézve, ha ezt ki szeretném küszöbölni.

Az egyik, hogy növelünk a mintavételi frekvencián. Ezzel az a fő probléma, hogy nagyon sokkal kellene növelni azt, hogy az alapharmonikus frekvenciájának 8-szorosa is a mintavételi frekvencia felénél alacsonyabb legyen. A másik, sokkal kézenfekvőbb megoldás, hogy több kimeneti hullámformát is kialakítunk és más lenne a jelalak magasabb, valamint mélyebb hangokon. Ekkor természetesen a magas hangok hullámformájában nem szerepelnének a magas felharmonikusok, amik már kilóghatnak a mintavételi frekvencia fele által állított sávkorlátból. Ez egyéb szempontok szerint is egy jó megoldás lenne, változatosabbá tenné a hangszínt.

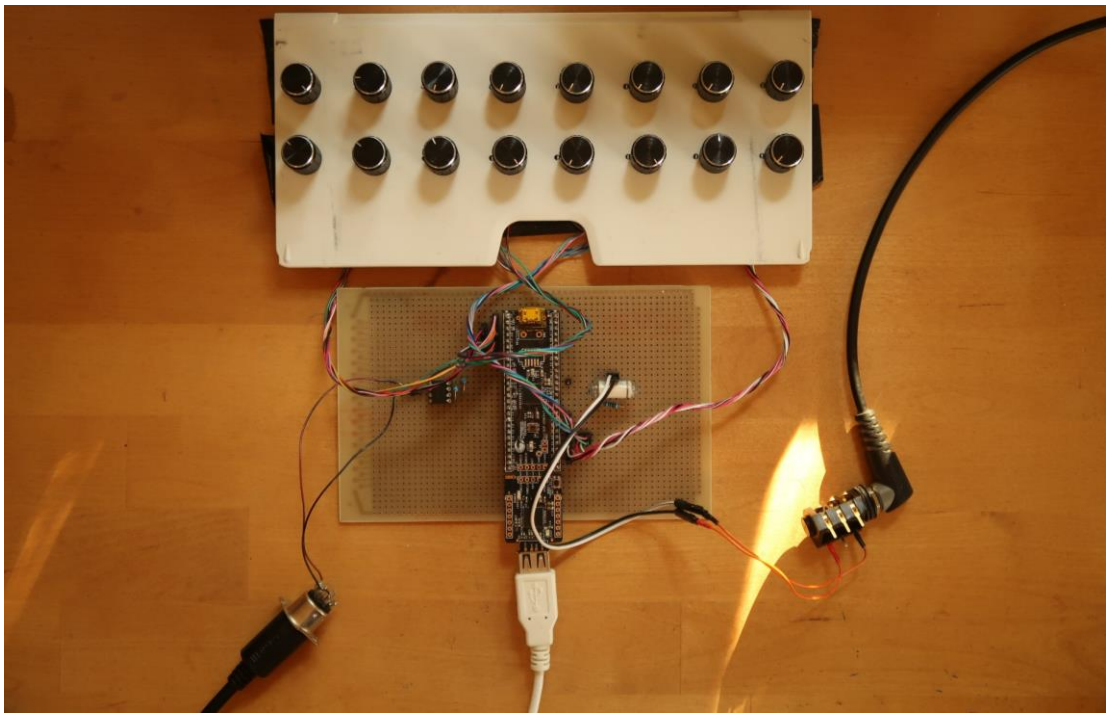
6.1.2. A 16 biten ábrázolt fázis következményei

Mivel a fázislépést egész számmal közelítjük, a lejátszott hangok frekvenciájának pontossága sérül. Ez a pontatlanság inkább mély hangok esetén jön jobban elő, ugyanis ekkor $d\varphi$ kicsi, így az egész számra kerekítéssel nagyobb relatív hibát okozunk. Az ily módon létrehozott hangok frekvenciájának eltérését az ideális temperált skála hangjaitól a 28-29-es ábrákon látható táblázatban ábrázoltam, mely a függelékben található. *(A táblázat alapjául a Wikipédia „Piano key frequencies” oldalán található táblázatot használtam fel.)* A zongora billentyűzetén megtalálható hangok esetében a pontosság kielégítő, a nagyobb hibák az annál mélyebb hangok esetén jelentkeznek. Fontos látni, hogy habár a 40 kHz mintavételi frekvenciával így nem szorulunk korrekcióra, nagyobb mintavételi frekvencia választása kisebb $d\varphi$ értékeket eredményezne, ami rontaná a hangmagasságok pontosságát. Amennyiben tehát a mintavételi frekvencia növelése mellett döntünk, szükséges lehet a fázis felbontásának növelése is.

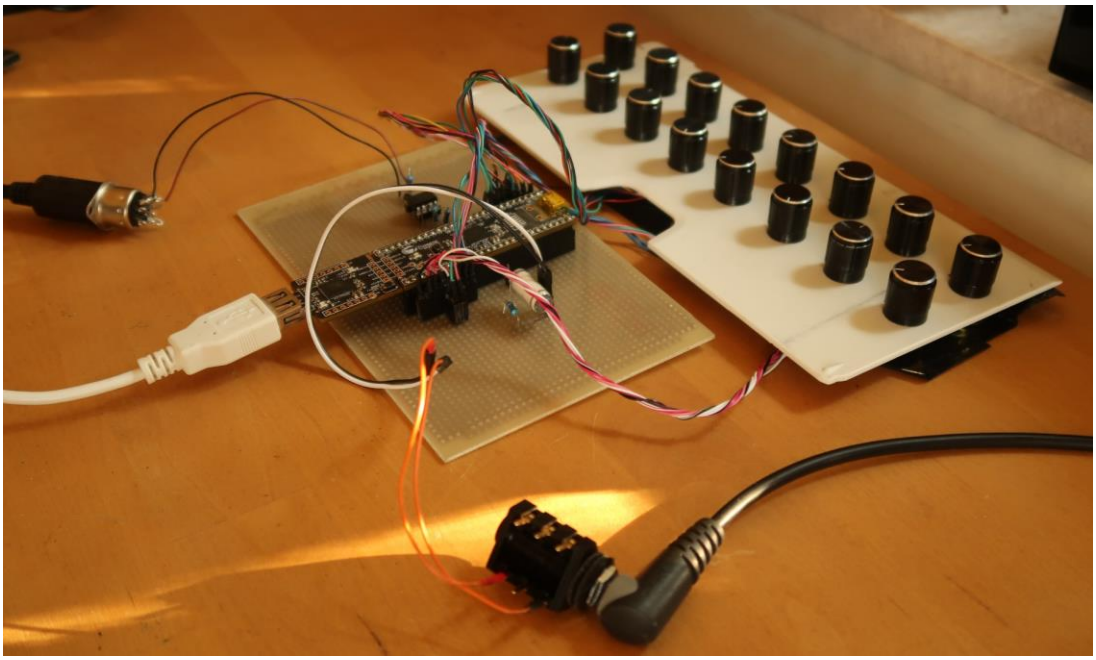
Képek



Ábra 29 KIMENETEN MÉRT SZINUSZHULLÁM



Ábra 30 A MEGVALÓSÍTOTT SZINTETIZÁTOR (1)



ÁBRA 31 A MEGVALÓSÍTOTT SZINTETIZÁTOR (2)

Irodalomjegyzék

[1] The Complete MIDI 1.0 Detailed Specification Published by: The MIDI Manufacturers Association Los Angeles, CA

[2] Wikipedia contributors, "MIDI" Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=MIDI&oldid=1058610546 (December 3, 2021).

[3] Wikipedia contributors, "Piano key frequencies" Wikipedia, The Free Encyclopedia, Piano key frequencies - Wikipedia (December 3, 2021).

[4] MIDI Association website: https://midi.org/ (December 3, 2021)

[5] DAFX - Digital Audio Effects. Edited by Udo Zolzer

[6] PSoC® Creator™ User Guide by Cypress Semiconductor An Infineon Technologies Company 198 Champion Court San Jose, CA

[7] PSoC® 3 and PSoC 5LP: Getting More Resolution from 8-Bit DACs (AN64275)by: Mark Hastings

[8] PSoC® 5LP: CY8C58LP Family Datasheet by: Cypress Semiconductor Corporation, 198 Champion Court, San Jose, CA

[9] PSoC® Creator™ Component Datasheet: 8-Bit Current Digital to Analog Converter (IDAC8) by: Cypress Semiconductor Corporation

[10] PSoC® Creator™ Component Datasheet: Interrupt 1.70 by: Cypress Semiconductor Corporation

Függelék

billentyű	MIDI note	Helmholz	Scientific	temperált	deltaFi (pontos)	deltaFi (kerekített)	mi frekvenciánk	eltérés (Hz)	relatív eltérés
88	108	c'''' 5-line octave	C8 Eighth octave	4186.009	6858.35715	6858	4185.7910	-0.2180	0.005%
87	107	b''''	B_7	3951.066	6473.42653	6473	3950.8057	-0.2603	0.007%
86	106	a♯''''/b♭''''	$A♯_7/B♭_7$	3729.31	6110.1015	6110	3729.2480	-0.0620	0.002%
85	105	a''''	A_7	**3520**	5767.168	5767	3519.8975	-0.1025	0.003%
84	104	g♯''''/a♭''''	$G♯_7/A♭_7$	3322.438	5443.48242	5443	3322.1436	-0.2944	0.009%
83	103	g''''	G_7	3135.963	5137.96178	5138	3135.9863	0.0233	0.001%
82	102	f♯''''/g♭''''	$F♯_7/G♭_7$	2959.955	4849.59027	4850	2960.2051	0.2501	0.008%
81	101	f''''	F_7	2793.826	4577.40452	4577	2793.5791	-0.2469	0.009%
80	100	e''''	E_7	2637.02	4320.49357	4320	2636.7188	-0.3012	0.011%
79	99	d♯''''/e♭''''	$D♯_7/E♭_7$	2489.016	4078.00381	4078	2489.0137	-0.0023	0.000%
78	98	d''''	D_7	2349.318	3849.12261	3849	2349.2432	-0.0748	0.003%
77	97	c♯''''/d♭''''	$C♯_7/D♭_7$	2217.461	3633.0881	3633	2217.4072	-0.0538	0.002%
76	96	c'''' 4-line octave	C7 Double high C	2093.005	3429.17939	3429	2092.8955	-0.1095	0.005%
75	95	b'''	B_6	1975.533	3236.71327	3237	1975.7080	0.1750	0.009%
74	94	a♯'''/b♭'''	$A♯_6/B♭_6$	1864.655	3055.05075	3055	1864.6240	-0.0310	0.002%
73	93	a'''	A_6	**1760**	2883.584	2884	1760.2539	0.2539	0.014%
72	92	g♯'''/a♭'''	$G♯_6/A♭_6$	1661.219	2721.74121	2722	1661.3770	0.1580	0.010%
71	91	g'''	G_6	1567.982	2568.98171	2569	1567.9932	0.0112	0.001%
70	90	f♯'''/g♭'''	$F♯_6/G♭_6$	1479.978	2424.79596	2425	1480.1025	0.1245	0.008%
69	89	f'''	F_6	1396.913	2288.70226	2289	1397.0947	0.1817	0.013%
68	88	e'''	E_6	1318.51	2160.24678	2160	1318.3594	-0.1506	0.011%
67	87	d♯'''/e♭'''	$D♯_6/E♭_6$	1244.508	2039.00191	2039	1244.5068	-0.0012	0.000%
66	86	d'''	D_6	1174.659	1924.56131	1925	1174.9268	0.2678	0.023%
65	85	c♯'''/d♭'''	$C♯_6/D♭_6$	1108.731	1816.54487	1817	1109.0088	0.2778	0.025%
64	84	c''' 3-line octave	C6 Soprano C (High C)	1046.502	1714.58888	1715	1046.7529	0.2509	0.024%
63	83	b''	B_5	987.7666	1618.3568	1618	987.5488	-0.2178	0.022%
62	82	a♯''/b♭''	$A♯_5/B♭_5$	932.3275	1527.52538	1528	932.6172	0.2897	0.031%
61	81	a''	A_5	**880**	1441.792	1442	880.1270	0.1270	0.014%
60	80	g♯''/a♭''	$G♯_5/A♭_5$	830.6094	1360.87044	1361	830.6885	0.0791	0.010%
59	79	g''	G_5	783.9909	1284.49069	1284	783.6914	-0.2995	0.038%
58	78	f♯''/g♭''	$F♯_5/G♭_5$	739.9888	1212.39765	1212	739.7461	-0.2427	0.033%
57	77	f''	F_5	698.4565	1144.35113	1144	698.2422	-0.2143	0.031%
56	76	e''	E_5	659.2551	1080.12356	1080	659.1797	-0.0754	0.011%
55	75	d♯''/e♭''	$D♯_5/E♭_5$	622.254	1019.50095	1020	622.5586	0.3046	0.049%
54	74	d''	D_5	587.3295	962.280653	962	587.1582	-0.1713	0.029%
53	73	c♯''/d♭''	$C♯_5/D♭_5$	554.3653	908.272108	908	554.1992	-0.1661	0.030%
52	72	c'' 2-line octave	C5 Tenor C	523.2511	857.294602	857	523.0713	-0.1798	0.034%
51	71	b'	B_4	493.8833	809.178399	809	493.7744	-0.1089	0.022%
50	70	a♯'/b♭'	$A♯_4/B♭_4$	466.1638	763.76277	764	466.3086	0.1448	0.031%

Ábra 32 Hangok frekvenciabeli pontossága 1 (Felhasználva: Wikipédia: „Piano key frequencies”)

billentyű	MIDI note	Helmholz	Scientific	temperált	deltaFi (pontos)	deltaFi (kerekített)	mi frekvenciánk	eltérés (Hz)	relatív eltérés
49	69	a′	A4 A440	**440**	720.896	721	440.0635	0.0635	0.014%
48	68	g♯′/a♭′	$G♯_4/A♭_4$	415.3047	680.43522	680	415.0391	-0.2656	0.064%
47	67	g′	G_4	391.9954	642.245263	642	391.8457	-0.1497	0.038%
46	66	f♯′/g♭′	$F♯_4/G♭_4$	369.9944	606.198825	606	369.8730	-0.1214	0.033%
45	65	f′	F_4	349.2282	572.175483	572	349.1211	-0.1071	0.031%
44	64	e′	E_4	329.6276	540.06186	540	329.5898	-0.0378	0.011%
43	63	d♯′/e♭′	$D♯_4/E♭_4$	311.127	509.750477	510	311.2793	0.1523	0.049%
42	62	d′	D_4	293.6648	481.140408	481	293.5791	-0.0857	0.029%
41	61	c♯′/d♭′	$C♯_4/D♭_4$	277.1826	454.135972	454	277.0996	-0.0830	0.030%
40	60	c′ 1-line octave	C4 Middle C	261.6256	428.647383	429	261.8408	0.2152	0.082%
39	59	b	B_3	246.9417	404.589281	405	247.1924	0.2507	0.102%
38	58	a♯/b♭	$A♯_3/B♭_3$	233.0819	381.881385	382	233.1543	0.0724	0.031%
37	57	a	A_3	**220**	360.448	360	219.7266	-0.2734	0.124%
36	56	g♯/a♭	$G♯_3/A♭_3$	207.6523	340.217528	340	207.5195	-0.1328	0.064%
35	55	g	G_3	195.9977	321.122632	321	195.9229	-0.0748	0.038%
34	54	f♯/g♭	$F♯_3/G♭_3$	184.9972	303.099412	303	184.9365	-0.0607	0.033%
33	53	f	F_3	174.6141	286.087741	286	174.5605	-0.0536	0.031%
32	52	e	E_3	164.8138	270.03093	270	164.7949	-0.0189	0.011%
31	51	d♯/e♭	$D♯_3/E♭_3$	155.5635	254.875238	255	155.6396	0.0761	0.049%
30	50	d	D_3	146.8324	240.570204	241	147.0947	0.2623	0.179%
29	49	c♯/d♭	$C♯_3/D♭_3$	138.5913	227.067986	227	138.5498	-0.0415	0.030%
28	48	c small octave	C_3	130.8128	214.323692	214	130.6152	-0.1976	0.151%
27	47	B	B_2	123.4708	202.294559	202	123.2910	-0.1798	0.146%
26	46	A♯/B♭	$A♯_2/B♭_2$	116.5409	190.940611	191	116.5771	0.0362	0.031%
25	45	A	A_2	**110**	180.224	180	109.8633	-0.1367	0.124%
24	44	G♯/A♭	$G♯_2/A♭_2$	103.8262	170.108846	170	103.7598	-0.0664	0.064%
23	43	G	G_2	97.99886	160.561332	161	98.2666	0.2677	0.273%
22	42	F♯/G♭	$F♯_2/G♭_2$	92.49861	151.549723	152	92.7734	0.2748	0.297%
21	41	F	F_2	87.30706	143.043887	143	87.2803	-0.0268	0.031%
20	40	E	E_2	82.40689	135.015449	135	82.3975	-0.0094	0.011%
19	39	D♯/E♭	$D♯_2/E♭_2$	77.78175	127.437619	127	77.5146	-0.2671	0.343%
18	38	D	D_2	73.41619	120.285086	120	73.2422	-0.1740	0.237%
17	37	C♯/D♭	$C♯_2/D♭_2$	69.29566	113.534009	114	69.5801	0.2844	0.410%
16	36	C great octave	C2 Deep C	65.40639	107.161829	107	65.3076	-0.0988	0.151%
15	35	B,	B_1	61.73541	101.147296	101	61.6455	-0.0899	0.146%
14	34	A♯,/B♭,	$A♯_1/B♭_1$	58.27047	95.470338	95	57.9834	-0.2871	0.493%
13	33	A,	A_1	**55**	90.112	90	54.9316	-0.0684	0.124%
12	32	G♯,/A♭,	$G♯_1/A♭_1$	51.91309	85.0544067	85	51.8799	-0.0332	0.064%
11	31	G,	G_1	48.99943	80.2806661	80	48.8281	-0.1713	0.350%
10	30	F♯,/G♭,	$F♯_1/G♭_1$	46.2493	75.7748531	76	46.3867	0.1374	0.297%
9	29	F,	F_1	43.65353	71.5219436	72	43.9453	0.2918	0.668%
8	28	E,	E_1	41.20344	67.5077161	68	41.5039	0.3005	0.729%
7	27	D♯,/E♭,	$D♯_1/E♭_1$	38.89087	63.7188014	64	39.0625	0.1716	0.441%
6	26	D,	D_1	36.7081	60.142551	60	36.6211	-0.0870	0.237%
5	25	C♯,/D♭,	$C♯_1/D♭_1$	34.64783	56.7670047	57	34.7900	0.1422	0.410%
4	24	C, contra-octave	C_1 Pedal C	32.7032	53.5809229	54	32.9590	0.2558	0.782%
3	23	B,,	B_0	30.86771	50.5736561	51	31.1279	0.2602	0.843%
2	22	A♯,,/B♭,,	$A♯_0/B♭_0$	29.13524	47.7351772	48	29.2969	0.1616	0.555%
1	21	A,,	A_0	**27.5**	45.056	45	27.4658	-0.0342	0.124%

ÁBRA 33 HANGOK FREKVENCIABELI PONTOSSÁGA (FELHASZNÁLVA: WIKIPÉDIA: „PIANO KEY FREQUENCIES”)