Tartalomjegyzék

1 Bevezetés *3*

2 A kezdetektől mostanáig... ..... *4*

3 A digitális televíziójel előállítása *12*

3.1 A hangcsatornáról röviden **12**

3.2 A videocsatorna digitalizálásának főbb lépései **15**

3.3 A szabványosított csomagok **18**

4 A transport stream *19*

4.1 A transport stream általános bemutatása **19**
4.1.1 A transport stream fejrésze 21
4.1.2 Az Adaptation field 23
4.1.3 A Program Elementary Stream 24
4.1.4 A további táblákról röviden 26

5 A programfa elkészítése *28*

5.1 A feladat rövid áttekintése **29**
5.1.1 A PAT tábla 30
5.1.2 A PMT tábla 31

5.2 A PAT-ok megkeresése **33**

5.3 A PAT feldolgozása **35**

5.4 A PMT feldolgozása **36**

5.5 A program által nyújtott végeredmény **39**

5.7 Az ellenőrzés főbb lépései **40**
5.7.1 Az előkészületek 40
5.7.2 A PAT csomag feldolgozása 41
5.7.3 A 12722_d című PMT feldolgozása 43
5.7.4 A 28521_d azonosítóval rendelkező csomag feldolgozása 45

6 A PCR *48*

6.1 A PCR lényege a dekódolás során **48**
6.1.1 Az adó oszcillátorára vonatkozó megkötések 49

6.2 A PCR beültetési eljárása **50**

6.3 A PCR mérésének lehetőségei **53**

6.4 A PCR méréséhez tartozó elméleti tudnivalók **55**

6.5 Hardveres úton történő mérési módszerek **57**

6.8 Az Ethernet átalakító működése **59**

6.9 PCR_AC mérése 60

6.10 A többi mérés megvalósítása 61
6.10.1 A digitális szűrő megvalósítása szoftveresen 61
6.10.2 A VCO megvalósítása 62
6.10.3 PCR_OJ mérési elve 63
6.10.4 PCR_DR mérési elve 63
6.10.5 PCR_FO mérési elve 64
6.10.6 Összegzés 64

6.11 A PCR_AC mérésének rövid leírása 65
6.11.1 A PCR mező megtalálása 65
6.11.2 A mérés menete 67

7 Az elkészített program működése 69

8.1 Az állandó részek 69

7.2 Az előugró hibaablak 70

7.3 A "Select PID" menü 71

7.4 A "View TS" menü 71

7.5 A "PCR measurement" menü 73

7.3 A kiterjesztett ablak részei 75

8 Mérések végzése az elkészített program segítségével 76

8.1 Ellenőrzés 76

8.2 Egy műholdas rendszer vizsgálata 80

8.3 A DVB-T PCR hibájának vizsgálata 84

8.4 Egy remultiplexer vizsgálata 88

9 Összegzés 92

10 Függelék 94

11 Irodalomjegyzék 106

1 Bevezetés

A televíziózás története folyamatosan íródott az eltelt évek, évtizedek során. Nem rendelkezik nagy múlttal, de nagy változásokon ment keresztül röpke 60 év leforgása alatt. Az elkövetkező években még nagyon sokat fog változni a digitális műsorszórás eljöttével. Az analóg átviteli mód lassan már csak egy szép emlék marad, amit unokáinknak könnyes szemmel mesélünk majd öreg napjainkban.

Mivel ma még javában használjuk ezeket az átviteli módszereket, ezért szeretném a múlt nagyjainak szánni szakdolgozatom bevezető részét. Nem léphetünk át régmúlt idők kiválóságain, hiszen ők indították el azt a folyamatot, aminek folytatását ránk bízták. Ez a mi örökségünk. Nem tervezem részletesre, hanem egy-egy érdekes eseménnyel szeretném a bennünk felhalmozódott emlékeket előhozni. Felesleges volna az AM jelet bemutatni, hiszen erre a két betűre tömeges gondolatok törnek elő. Eme két szóra megjelennek az egyenletek, a vektorábrák, az oszcilloszkóp ábrái… Ezek a dolgok már belénk ivódtak, tehát részletes magyarázatra nincs szükség.

A bevezető fejezetek után megismerkedhetünk a digitális átvitellel úgy, ahogy eddig még nem foglalkoztunk vele. Az iskolában megtanultuk az átvitellel kapcsolatos mérési módszereket, az mpeg kódolás, dekódolás folyamatait, viszont nem törődtünk a TS összerakási folyamatával. Ezt mutatom be diplomám második részében, a programcsomag készítésének mechanizmusát, majd ennek ellenőrzésére írok egy programot Delphi 7 alatt.

A diplomamunkám készítése során szeretném a hangsúlyt a PCR jelek vizsgálatára fordítani. Meg szeretném vizsgálni, hogy mire használják, mit okoz a hibás átvitele, hogyan ellenőrizhetjük az átvitel hibamentességét. Ennek mérésére át kell tekintenem a PCR mérési módszereket, majd ezeket figyelembe véve meg kell valósítanom a saját mérési módszeremet, melyet egy számítógépes program megírásával tervezem.

Itt szeretném megköszönni a CableWorld Kft igazgatójának, kit külső konzulensként is magam mellett tudhattam, és munkatársainak, hogy munkámat tanácsaikkal, ötleteikkel nagymértékben segítették. Köszönöm, hogy ott tölthettem el diplomaírási időmet, gyakorlatot szerezhettem főiskolai tanulmányaim befejezéséül.

2 A kezdetektől mostanáig...

Hol és mikor kezdődött el a „Televízió” történelme? Erre a kérdésre válaszolni igen nehéz. Ha ezt a kérdést szegezné valaki nekünk, csak állnánk bambán. Mikor készült el az első TV? Ez lenne a válasz? Ez, mint tudjuk, nem helyénvaló gondolat, hiszen a nyári fürdőruha-kollekció is tavasszal készül, a szezon előtt több hónappal. A szezon nem akkor kezdődik, mikor elkészítik a fotókat, mikor elkészül a katalógus, hanem mikor az a vásárlók kezébe kerül.

Elnézést a kis kitérőért, de talán így érthetőbb lesz az én válaszom az elsőként feltett kérdésemre. Mikor megalkották az első TV-t, ezzel elkészült az első fotó szabadalma, és ezzel elkészült a katalógus. A kis Farnsworth, aki évesen kitalálta a letapogatás elvét 1921-ben, mit sem kapott találmányából, és szegényen halt meg, mint bármely nagy tudós. Mikor kapták kézhez az emberek a katalógust? Ez a válasz a kérdésre. Szóval, az én szerény véleményem szerint a kérdésre a kielégítő válasz nekünk, európaiaknak, az 1936-os év. No nem az olimpia miatt, bár szorosan hozzátartozik. A berlini olimpia sugárzása volt az első európai televíziós műsorsugárzás. Berlin több pontjára elhelyeztek készülékeket, így azok is élvezhették az olimpia minden egyes mozzanatát, akik nem jutottak jegyhez. Mondhatni ez volt a főpróba, ami igazán jól sikeredett. Ezzel a közönség elé került az új technika. És elkezdődött...

1952-ben kezdődtek meg az első színes televízióadások. 1953 januárjában kezdte meg a Magyar Televízió Vállalat a kísérleti sugárzását, és csak 4 évre rá kezdődött meg a hivatalos vetítés. Ne feledjük azokat az élményeket, melyek akkor keletkeztek. Sajnos én csak hallomásból ismerek ilyen történeteket, most is magam előtt látom apámat, ki csillogó szemmel mesélte nekem, hogy 1Ft-ért járt át a szomszédba, hogy a Zorrót láthassa. Anyám szülei OTP-re vették az első TV-t, ráadásul úgy, hogy fizetésük negyede ráment a hitel törlesztésére. Mindez csak azért, hogy vehessenek egy TV-t. Ilyen, és ehhez hasonló történetek mindegyikünknek eszébe jut, ha egy kicsit visszaemlékezünk, vagy ha másképp nem, megszólaltatjuk az őseinket. Akkor még volt értéke a kép és hang együttes jelenlétének, még akkor is, ha kicsit sípolt a hang, kicsit szemcsés volt a kép. Ma viszont zúgolódunk, dühösek vagyunk, ha nem láthatjuk kedvenc sorozatunkat kellő, vagy jobb

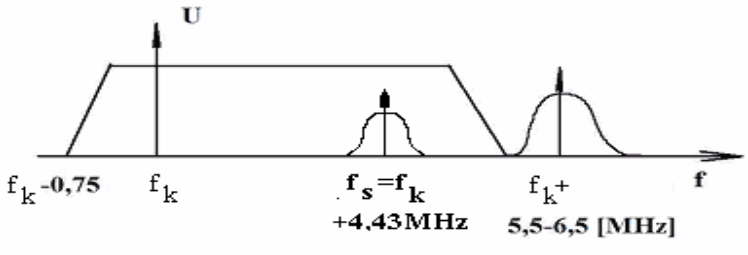
minőségben. Ez nem rossz, hisz így van, lesz nekünk, kik a televíziós szakmát választottuk hivatásunknak, még jó sokáig munkánk. Ne szaladjunk ennyire előre, kanyarodjunk vissza a kicsiny történetünkhöz.

A kezdeti sikerek után nem álltak meg a serény mérnökök, 1949-ben az amerikaiak megalkották az NTSC-t, majd a franciák a SECAM-ot 1957-ben, végül az NSZK-s Walter Bruch 1961-ben a PAL-t. Ezek nem titkos kódok, amit a távirányítós tévénk megjelenít egy bizonyos gombot lenyomva, hanem ezek a nevek egyes eljárásokat tartalmaznak, amikkel a színes információkat tudjuk keresztül vinni az éteren úgy, hogy a fekete-fehér átvitel nem szenved csorbát. A túloldalon készüléktől függően élvezhetjük a műsort eme találmányok segítségével színesen vagy fekete-fehéren.

A televíziózás nem maradt a Földünk szűk keretei közt, mivel 1957. október 4-én startolt a Szputnyik1, a szovjetek első műholdja, és igaz, hogy csak rövid ideig működött, de addig sugározta a bip-bip-eket serényen, amit még az amerikaiak is tudtak venni. Ezzel megkezdődött az űr meghódítása. Egy évre rá megszületett az amerikai verzió is. A szovjetek azért erőltették a műholdas műsorszórást, mert országuk hatalmas területét ezzel a módszerrel lehet a legkönnyebben ellátni. Ne gondoljuk, hogy csak ilyen „önzetlen” célok vezérelték ezt az országot a hidegháború idején. Azért az amerikaiakat sem kellett félteni.

1. ábra Egy műholdról készített látványkép

Ne feledjük azonban, hogy a háttérben azért szépen, lassan terjedt a földfelszíni műsorszórás. Egyre több háztartás rendelkezik televízióval, egyre többen élvezhetik ennek a szépségét. A sugárzás analóg módon történt, AM-VSB modulációval a video-, és FM segítségével pedig a hanginformáció (2. ábra).



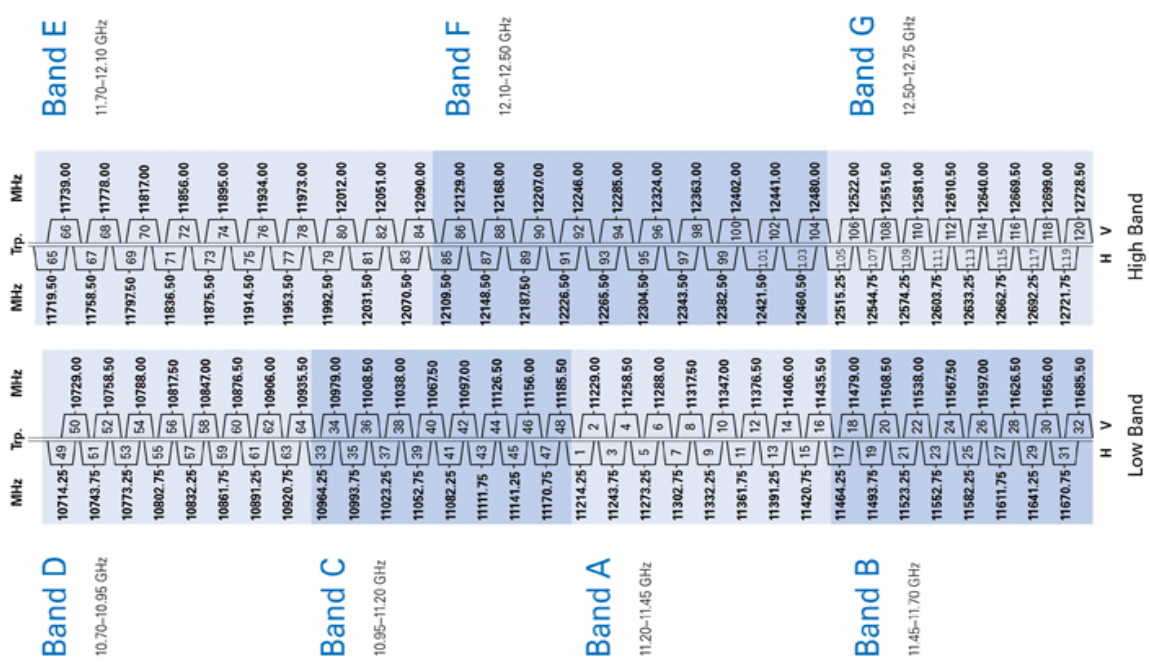
2. ábra Analóg TV jel spektruma

Mint mindig, voltak olyanok, akik nem elégedtek meg a minőséggel, akik elégedetlenkedtek. A kétkedők is belejátszottak a fejlődés ütemébe, hiszen 1987-ben egy műhold már sugározta első HDTV jeleit. Igaz, még csak analóg módon. 1991-ben elkezdődtek a digitális tévé jelátvitelével kapcsolatos kutatások, tesztelések. Miután megoldották az átvitellel kapcsolatos új problémákat, újabbak keletkeztek. Hogyan jellemezzük az újonnan kapcsolatos átvitelt? A régi, analóg mérési módszerek nem jellemezték jól a varázslatosan különc digitális jelet. Erre is megtalálták a választ a serény mérnökök. Megjelent a BER, a MER, a Grade of Service. Talán csak a frekvencia és a teljesítménymérés maradt meg. Most már tudtak digitális jelet továbbítani.

"Jó, jó"- legyinthetnénk erre unalmasan, de nekünk csak az analóg sugárzás jut, hiába van digitális sugárzás. Ez igaz, meg az is, hogy először az égbolton, tőlünk 36000km távolságra, geostacionárius pályán (illetve a szovjetek által kitalált elliptikus) lévő műholdak használták ezt az átvitelt. Miért is? Az első sikeres pályára állítás után nem állt meg a fejlődés. Az évek multával rájöttek, hogy ebben jó üzlet van. Ha van egy műholdam, mellyel műsort sugározhatok, akkor a televíziós társaságok egymás kezébe adják szobaajtóm kilincsét, mindenkinek az én műholdam fog kelleni. Egészen merésznek hathatott ez az több évtizeddel ezelőtt. Most visszagondolva van realitása ennek a feltevésnek. Hiszen minél több helyen fogható az adásom, annál nagyobb lehet a nézettségem, annál több pénzt kérhetek el a reklámaimért. A műhold elterjedésével, például nem kell nélkülöznie a német turistának az RTL-en futó kedvenc sorozatát, miközben a Balatonon sütteti a hasát. Mint látjuk, ez az ipar csak akkor életképes, ha van piaca. Ehhez meg az kell, hogy a vevő elégedett legyen. Lehet, hogy csak 4-5 csatornát néz, de szüksége van arra az élményre, hogy neki 150 csatornája van.

Kicsit letértem az általam tervezett ösvényről, de eme kis kerülő után térjünk vissza

a kérdéshez. Miért kell digitális? Mint azt az előző bekezdésben már említettem, ebben nagy üzlet van. A hozzám ellátogató tévé társaságoktól is többet kérhetek, ha a műholdpozíciómról több adás fogható, mint a vetélytársaim adóiról. Igen ám, de a frekvencia véges. A 3. ábrán láthatunk a frekvencia kiosztásra példát. Ezt az Astra, ami egy műholdas sugárzással foglalkozó cég, honlapjáról töltöttem le. Ennyi csatornát foghatok a parabolaantennámmal, ha az Astra 1 pozícióra állok.



3. ábra Az Astra egy frekvenciakiosztása

Mint az a képen látható, kétféle sugárzási móddal is kb. 120 analóg csatornát tudunk biztosítani 10,7-12,75GHz-ig terjedő frekvenciatartományunkban. Azért nem ennyire egyszerű a dolog, hiszen ezt a frekvenciasávot több műholddal fogják közre. A fenn látható frekvenciaterv elvi jellegű, hiszen a műholdak átcsoportosítása bármikor megtörténhet. Az itt látható csatornakiosztás az Astra 1-es műholdakra készült, viszont a most fogható műholdak: 1B, 1C, 1E, 1F, 1G, 1H, 2C. Tehát a 120 db csatorna sem mindig tartható. Ez a frekvenciatartomány, és a műholdak egymás mellett élésének megőrzése nem kínál nekünk kellő számú csatornát, ami kielégíti igényeinket. Mivel ez a darabszám igen kevésnek bizonyult, így váltani kellett. Az új, digitális rendszernek bele kellett férnie a műholdsugárzást szolgáltató cégek által meghatározott csatornánkénti sávszélességbe. Ebbe a sávszélességbe eddig is elfért az analóg videojel mellé több hangcsatorna, más,

rádióállomások jelei. A digitális jelátvitel gazdaságossága abban rejlik, hogy az átviendő információ nemcsak frekvenciában, de időben is multiplexálható. Habár ezzel a módszerrel akár nyolcszor annyi csatornát tudnék átvinni, mint eddig. Ha ennyi plussz tévés nem fog hozzám jönni, majd felhasználom másra a felszabaduló frekvenciasávot. Talán internetet szolgáltatok, persze, hogyha igény van rá.

Tehát 1994-re készen állt a technika, már csak rendszerezni kellett az átvitelt az érthetőség miatt. 1993-ban elkészült az MPEG-1 szabványa, mi jelentősen letömörítette az átviendő bitek számát. Igaz, a biteket ki kellett egészíteni még különböző segédbitekkel (hibajavító, fejrész, stb.), de így is gazdaságosabb volt, mint az analóg átvitel. Már csak egy szabvány kellett, ami az átvitelt szabályozta. 1994. november 13-án ez is megszületett. Az ISO/IEC 13818-1. Igaz, ez csak a „Transport Stream”-re terjed ki (A transport stream ismertetése a következő fejezetekben megtalálható.). Ez rögzíti a digitális jelfolyamok összeszövésére vonatkozó szabályokat. Ez a szabvány már tartalmazta az MPEG-2-es átviteli mód jelzését is. Igaz az MPEG-2-es szabványosítása csak 1996-ban történt meg. Szükség volt az MPEG-2-es kódolási eljárás megvalósítására, hiszen az első verzió még csak egy hangot volt képes kezelni.

A digitális átvitelnek többé nem volt akadálya. Megkezdődhetett a programok multiplexálása, és a műholdak lassan rátértek erre a módszerre. A következő kérdés, ami felmerülhet bennünk, az az, hogy hiába sugároznak digitálisan, azt ki veszi. Arra a "műholdtulajdonosok" hamar rájöttek, hogy nem csak a televízió-társaságoktól lehet pénzt kérni, hanem a fogyasztóktól is. Mivel nem vett mindenki műholdvevőt, és a földfelszíni műsorsugárzás sem sugározhat több 10 csatornát bevétel nélkül, hiszen, mint tudjuk, már nincs negyedévente jövő TV- és Rádióadó. Ezért megjelent egy új ágazat a műsorszórás terén, ez a kábeltelevízió. A kábeltelevíziós rendszerek zárt rendszert alkotnak a fogyasztók felé. A kábeltelevíziós fejállomások veszik a műholdak jeleit (ezért jogdíjat fizetnek a műholdszolgáltatónak), és a földi műsorszórás jeleit, majd ezekből összeállítanak egy- vagy több programcsomagot, melyet elküldenek koaxiális vagy optikai kábelen, illetve mikrohullámon (Antenna Mikro) a fogyasztók háztartásaiba. A fogyasztó kiválaszthatja a számára és pénztárcájának legmegfelelőbb csomagot, majd a havonta küldött csekket rendezve élvezheti a jobbnál jobb műsorokat. Mivel ez egy zárt rendszer, így a kábeltelevízió szolgáltatójának rendelkezésre áll az egész frekvenciasáv majd' 1MHz-ig.

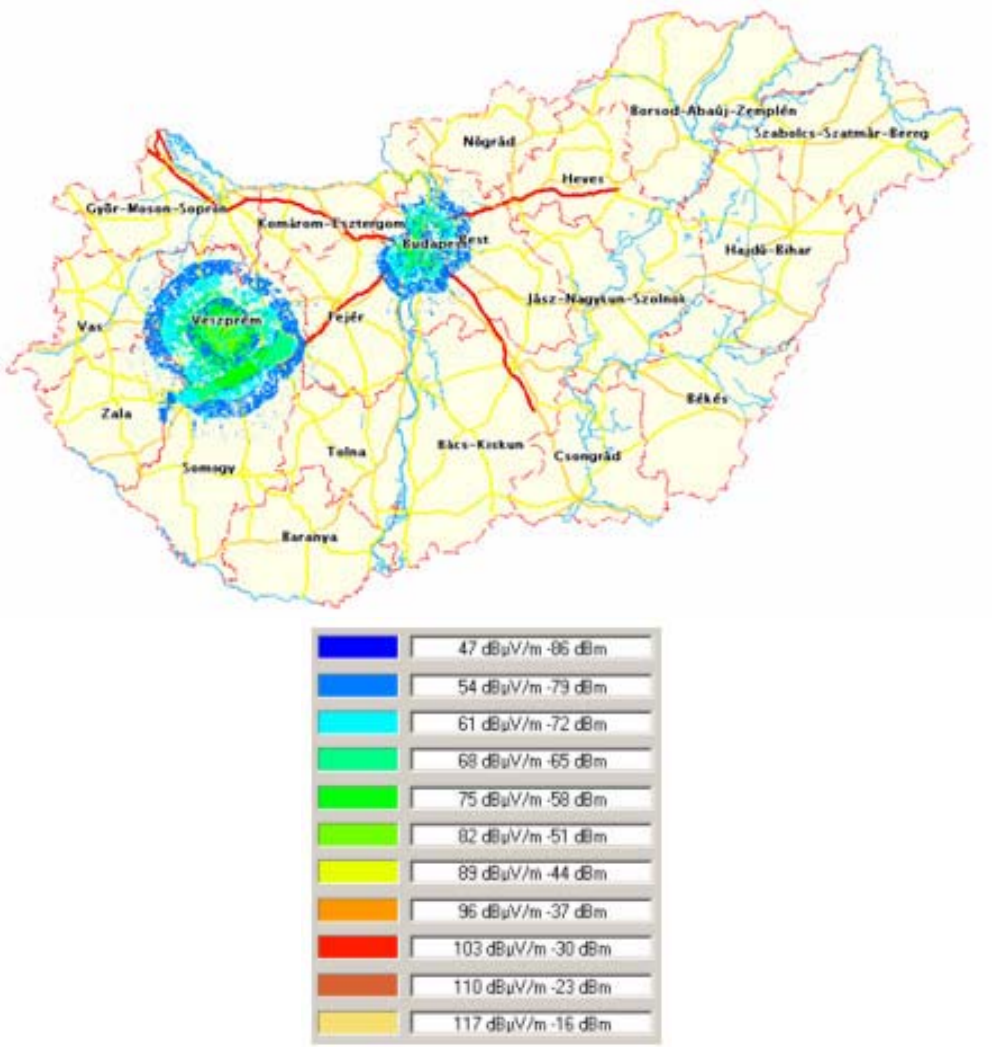
Én nem a kábeltelevíziós rendszereket szeretném bemutatni, így visszatérek az előző gondolatmenetemhez. A "kábelesek" megjelenésével lett vevőkerete a digitális jelnek, melynek igen nagy előnye az is, hogy jobban kódolható, mint az analóg jel. Így a műholdak is képesek programcsomagokat kínálni. A szekér tehát beindult, de a fejlődés nem állt meg. Hiszen, ha megállt volna, akkor egy igen rövid összefoglalás után be kéne fejeznem a szakdolgozatom.

Most már szinte minden megoldottnak tűnik. A finnyásaknak ott a HDTV, meg a digitális műholdjel (vegyenek digitális vevőt), a többieknek meg marad a jó öreg analóg továbbítás. (Nem is olyan öreg.) Mint mondottam, a fejlődés nem állt meg, és az igény is folyton nő. Valaki kitalálta az interaktivitás szócskát. Valaki úgy gondolta, hogy egy tévé miatt nem fog kerülgetni otthon egy "madzagot". Használjuk internetezésre, döntsük el azt, hogy mit nézzünk stb. Ezen új kérések kielégítésére az analóg átviteli mód nem alkalmas. Digitalizálni kellene a KTV-t. Megalakult a DVB-C szabvány. A digitális kábeltelevízió elterjedése a mi kis hazánkban az elkövetkező pár évre tehető. Mindemellett a földi műsorszórók is elgondolkodtak, hiszen a lakosság többsége még mindig csak őket veszi. Erre a megoldás a DVB-T.

Európában több ország esetében már működik a kereskedelmi DVB-T sugárzás, az első rendszeres DVB-T adást Nagy-Britanniában kezdték el 1998-ban, és azóta már üzemszerű szolgáltatások működnek Finnországban, Spanyolországban, Svédországban, Olaszországban, Hollandiában és Németország berlini régiójában is. A berlini sugárzás külön érdekessége az, hogy 2003 augusztusában teljesen leállították ebben a régióban az analóg földfelszíni műsorszórást. Európa szinte teljes területén, valamint számos Európán kívüli országban is kísérleti sugárzások indultak DVB-T formátumban. A legtöbb nyugat-európai országban már készítik a terveket a digitális rendszerekre való végleges áttérésre, az analóg szolgáltatások fokozatos megszüntetésére. Ezek szerint a fejlett országok többségében várhatóan 2010-re, 2015 után pedig már mindenhol szinte kizárólag csak digitális rendszerek fognak üzemelni. A 2005-ben már működő szolgáltatások várhatóan megtarthatják majd helyüket, így a többi ország részére csak a maradék frekvenciák kerülnek kiosztásra a konferencián.

Az Antenna Hungária Rt. az elmúlt években megtette a szükséges technikai előkészületeket a DVB-T sugárzásra, adóberendezést telepített a Széchenyi-hegyi

adóállomásához, majd 1999. július 9-én megkezdte az első hazai kísérleti adást. Kezdetben kölcsönberendezésekkel, majd 2001 októberétől saját eszközeivel folyamatosan sugározza a három közszolgálati műsort, illetve mérőjeleket a vizsgálatokhoz 1kW kisugárzott teljesítménnyel. A sugárzás a 43. és 51. csatornán történik. A kísérleti sugárzás alatt az Antenna Hungária Rt. folyamatosan vizsgálja a vételi lehetőségeket. Állandó kapcsolatot tart fenn az adóberendezés- és a vevőkészülék-gyártókkal, többféle adó- és vevőkészülék tesztelését végzi. 2002 májusában beindította az első vidéki DVB-T telephelyet Kab-hegyen, ahol új antennát telepítettek. A kab-hegyi adó a budapesti műsorcsomag jeleit, kiegészítve a környék regionális híradójával, sugározza. A lefedettségi térképen figyelemmel kísérelhetjük a vételi lehetőségeket [14].



4. ábra A magyarországi DVB-T jelenlegi lefedettsége [14]

Új feladatok elé kell néznünk. Meg kell tanulnunk összerakni egy "Transport Stream"-et (továbbiakban TS), ki kell tudnunk tölteni egy-egy PAT, CAT, PMT, NIT táblát. Mégpedig oly módon, ahogy azt összerakták azok a "műholdas" fickók. Több TS-ből ki kell tudnunk választani a szórni kívánt programokat, ezeket össze kell tudnunk rakni egy TS-é, meg kell tudnunk mérni az így kialakított jelfolyamunkat és ha valami gond van, bele kell tudnunk nyúlni. A nagy cégek kínálnak ilyen berendezéseket, de ők hozzá vannak szokva a "műholdasok" fantasztikus áraihoz, így kínálatuk igaz, hogy profi, de egy kis kábeltévés társaság nem engedheti meg ezen berendezéseket. Így nekik gyártani kell olcsóbb, de tudásban azonos mérőberendezéseket, mérőszoftvereket.

Ezért választottam ezt diplomatémámnak. Szeretném bemutatni számomra mindeddig homályos TS-t, annak felépítését, egy ilyen elkészítésének főbb mozzanatait. Emellett szeretnék megírni egy kis mérőszoftvert, ami felrajzolja a TS programstruktúráját és megméri a fellépő PCR jittert. Tudom, hogy ezen dolgok még ismeretlennek tűnhetnek, bevallom még nekem is, de remélem, hogy a végére minden kitisztul a fejünkben.

3 A digitális televíziójel előállítása

Mint azt jól tudjuk, ahhoz, hogy a jelünket, amely alapvetően analóg, és amit továbbítani szeretnénk, valamilyen módon 1-esek és 0-ák sorozatává kell alakítanunk. Ezért, mielőtt bemutatom a kész csomagot, melyet műholdon, vagy más egyéb úton el akarunk juttatni az előfizetőnk felé, szeretném röviden, csak a lényeget leírva feleleveníteni a digitális jelsorozat (legyen az bármelyik MPEG szabvány) felé vezető rögös út főbb állomásait.

Az analóg jelünkkel addig nem érünk semmit sem, amíg nem alakítottuk át digitális jellé. Ezt úgy valósítjuk meg, hogy külön digitalizáljuk be a videocsatorna jelét és külön a hangcsatornák jeleit.

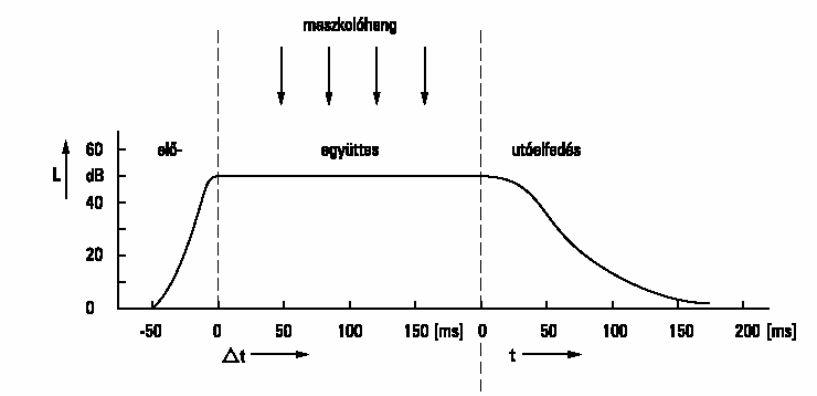
3.1 A hangcsatornáról röviden

Kezdjük először az egyszerűbbel, a hangcsatornák digitalizálásával. A digitalizálás előtt tisztában kell lennünk az ember hallásának jellegzetes vonásaival. Hiszen ezek segítségével hatékonyabban tudjuk digitalizálni, illetve a digitalizált jelet tömöríteni.

Tudnunk kell, hogy mekkora a hallásküszöb, hiszen annál kisebb mintákat felesleges kódolnunk, és a fájdalomküszöb fölé sem érdemes mennünk, hiszen az előfizetőnk biztos nem venné jó néven. A kettő között 120dB a távolság, tehát ezt a tartományt kell átfognunk.

Nem csak feszültségben lehet korlátozni, hanem időben is. Ennek alapja a hangelfedés jelensége. Ilyen például a folytonos hang okozta hangelfedés, melyre példát a mindennapi életből könnyen vehetünk. Képzeljük magunkat egy csendes szobába, ahol lágyan szól Vivaldi 4 évszakja, amikor az utcán beindul egy légkalapács. Ennek hatására a zene élvezhetetlenné válik, átviteli rendszerünkön elég csak a légkalapács hangját átvinnünk. A másik jelenség egy rövid ideig hallható hang, melynek hatására egy ideig nem hallunk mást. Biztosan mindenki füle mellett csattant már el levegővel felfújt zacskó, mely elég kellemetlen érzést nyújt a több másodpercig tartó csengés okán. Persze ennek van hasznos felhasználási területe is. Mielőtt a tüzérek kilőtték volna mozsárágyújuk tartalmát, fülük mellett sütötték el a közlegények puskájukat. Így a puska által okozott hanghatás

eltartott addig, míg az ágyú elsült. Ezzel a módszerrel megóvták fülük épségét. Egyébként erre elkészült egy tanulmány is, melyet az 5. ábra mutat be.



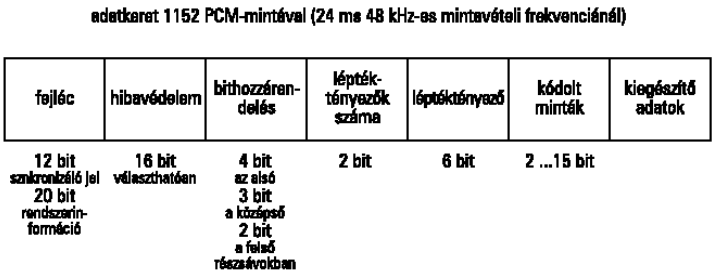
5. ábra A maszkoló hang hatása [9]

A maszkoló hanggal időben egybeeső elfedési jelenségeket szimultán elfedésnek hívjuk. Az elő-elfedési hatás sokkal rövidebb az utóelfedésnél, 5 ms-mal a maszkoló hang előtt már csökkenni kezd.

Persze ezeket csak úgy vehetjük számításba, hogy betartjuk az ISO/IEC 11172-3 (MPEG-Audio) és az ISO/IEC 13813-3 (MPEG-2 Audio) nemzetközi MPEG-szabványokban rögzítetteket. A minősége jobb, mint az FM átviteli rendszereké, így ahhoz, hogy a digitális tv sávszélessége elfogadható értékű legyen, megfelelően kialakított adatcsökkentési módszer alkalmazására van szükség a hangjelben éppúgy, mint a videojelben. A digitális kísérőhang átvitelének továbbá olyannak kell lennie, hogy a programszolgáltató szabadon választhasson a különféle adatsebességek és ezzel együtt a hangjel különféle minőségi fokozatai között. A hangcsatornának biztosítani kell két kísérőhangos, sztereo-, ill. surround-üzemmódokat is. Emellett mindezen a követelményeket kompatibilisen kell teljesíteni; így minden vevő képes a hangjelet dekódolni, függetlenül a választott üzemmódtól és a kódolás adatsebességétől. Ennek megfelelően alakították ki az egyes, a kettes, illetve a hármas rétegű kódolást. A digitális tv hangjának adatcsökkentése az MPEG-szabvány felhasználásával történik. Az eljárás a spektrum 32 részsávra bontásán és egy 12 vagy 36 mintából álló blokk képzésén alapul. A blokkok mintáinak kvantálása az emberi hallás elfedési sajátságait figyelembe vevő pszihoakusztikus modell alapján valósul meg. A létrejövő kvantálási zajt a jel spektrálisan és időben elfedi, aminek következtében

azt nem lehet meghallani. A multiplexált adatfolyamból a dekóder az eredeti hangmintákat rekonstruálja. Az 1-es rétegű kódolás teremtette meg a hangátvitel alapjait. A 2-es ennek továbbfejlesztett változata, mely kb. 100 kbit/s adatsebességgel rendelkezik, miközben a hangminőségben kódolásból származó romlás nem érzékelhető. A 3-as rétegű MPEG-hangkódolást azzal a céllal fejlesztették ki, hogy egy összetettebb kódolási folyamat révén a 2-es rétegűnél nagyobb tömörítést érjenek el amellett, hogy különféle térbeli hangok átvitelére is lehetőséget biztosítson. Az MPEG-hangdekóder valamennyi üzemmódú és adatsebességű hangjelet dekódolni tudja, ami a felhasználó mindenkori igényének kielégítését teszi lehetővé.

Mint minden digitális jelet, így a hangot is el kell látni segédbitekkel, melyek segítik a dekódolást a vételi oldalon. Erre látunk most példát a következő ábrán.



6. ábra Egy kódolt audiohang keretszerkezete [9]

Természetesnek tűnik a fejléc és a hibavédelem is, de a bithozzárendelés és a léptéktényező már nem annyira. Mivel egyes hangok tisztább hangzása (pl. zene) fontosabb, mint másoké (pl. hírek), így ezek átviteléhez több bitet kell felhasználnunk, míg a másikhoz kevesebbet. A minta leírásához felhasznált bitek számát adja meg a bithozzárendelés. Viszont egy versmondás, illetve egy beszélgetés rögzítéséhez elég kevesebb mintavétel, míg a nagy lövöldözős jelenethez, melyben a főhőst halálosan megsebesítik, több mintavétel szükséges. Ezt mutatja be a léptéktényező. Azért is hoztam fel a főhős tragikus végét, mert a hangot nem folyamatosan kódolják, hanem blokkokra bontva. Így megeshet az a furcsa helyzet, hogy egy blokkra esik a nagy dirr-durr és a beálló halotti csend. Ezért ebben a blokkban kettő vagy három léptéktényezőre is szükség lehet. Ha ellenben kevés a változás, egy is elegendő. A kódolóban külön egység vizsgálja, hogy hány léptéktényezőre van szükség. Ezt követően megtörténik a blokk maximumainak, azaz a léptéktényezőknek a

meghatározása. A léptékényezők mellett a dekóder számára közölni kell azok számát is. Most már nem maradt más, mint a kiegészítő adatok. Mi másra is használnánk, ha nem a térbeli hangzás megvalósítására? Bármi másra, ami csak a képzeletünk elő tud varázsolni. Most már van digitális hangunk, a rádióműsort-szolgáltatók kényelmesen hátradőlhetnek, amíg én elmesélem a következő rövidke mesét.

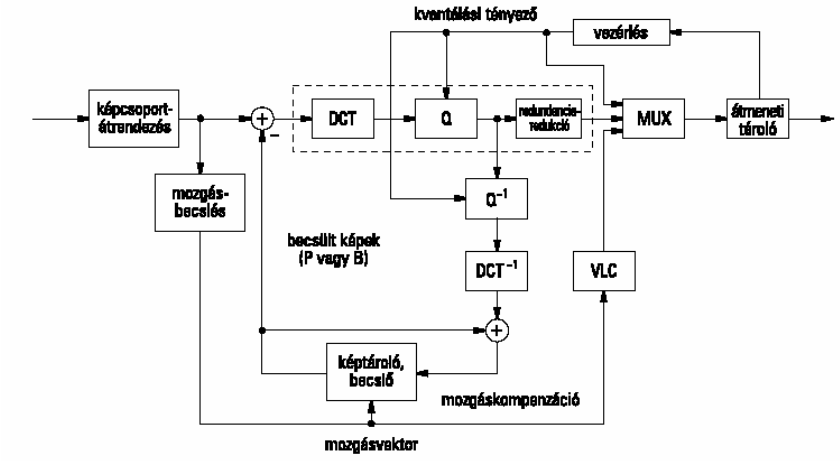
3.2 A videocsatorna digitalizálásának főbb lépései

Ez már nehezebb kérdés, hiszen a digitális videojel nem fér el az analóg sávban, pedig bele kell férnie! Ezért ezt is tömöríteni kell. Arra hamar rájöttek, hogy veszteségmentes eljárást kidolgozni lehetetlen, biztosan el kell dobnunk egy csomó információt. Mennyit és melyiket? Fogas kérdés, egy filmnél még meg tudjuk mondani, hogy tovább csókolózzanak, és a rohangálásukat meg hagyjuk ki mondván, azt úgyse nézik. A televízió azonban nem csak előre felvett dolgokat közvetít, hanem például híradót. Azt még könnyű, hiszen megmondom a csinos bemondónőnek, hogy ne mozogjon sokat, így nem kell átvinnem mindig őt, és a háttér is állandó. Igen ám, de ott vannak az élő foci közvetítések. Ott már nincs a háttérben egy gyors észjárású kisöreg, ki eldönti a kihagyható képkockákat. Ezért olyan rendszert kellett alkotni, ami maga mondja meg, hogy mit is hagyjon ki. Akinek önálló döntést biztosítunk. Sok ilyen keletkezett, például az AVI, a DIVX, és amit elfogadott a televíziós társadalom, az az MPEG alapú tömörítés.

Tegyünk egy gyors időutazást. 1993-ban készült el az MPEG-1 tömörítési eljárás, mely még csak egy hang kezelését tudta ellátni. Így hamar kinőtte magát, és megalakult a máig életképes MPEG-2, mely már kezelte a több hangot tartalmazó csomagot. Eléggé kötött ez a szabvány, tehát a hiperaktív emberek nem szeretik. Megjelent hamarosan az MPEG-3, melyet a HDTV jelekre, nagyobb felbontás megvalósítására fejlesztettek ki. Ez az eljárás nem tudta kiütni, az akkorra már szabványosított 2-es eljárást. 1999-ben megszületett az MPEG-4. Ekkor igencsak megörültek a hiperaktív emberek. Hiszen ez a szabvány szabadkezet ad szinte az egész tömörítési eljárás megváltoztatásához. Tehát magunk írhatunk tömörítési eljárásokat, döntési függvényeket, meg sok egyéb mást is. Még az a különlegessége ennek a szabványnak, hogy tartalmaz objektumkezelést és tartalomkezelést is. Ezek lehetővé teszik, hogy otthonról mi állítsuk be a képet. Mondok

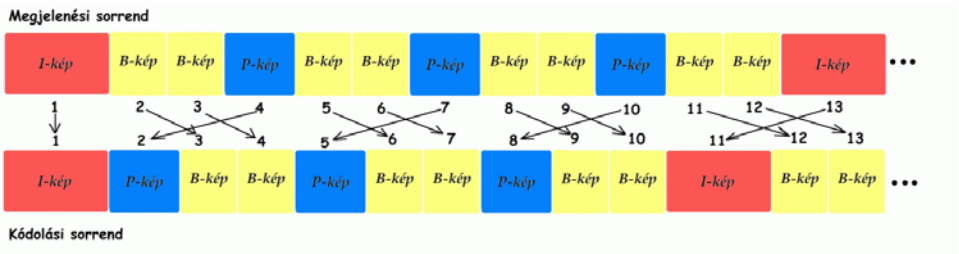
egy példát. Egy idegenvezetőnő elkalauzol bennünket Velencébe. Láthatjuk a csinos alakját és a látnivalókat egyszerre. Viszont annyiban tudunk beleszólni, hogy a hölgyet bárhova mozgathatjuk, például egy egérrel, a tévénk képernyőjén. Igaz, első hallásra kissé elképzelhetetlennek tűnik, és inkább számítógépet jelenít meg előttünk, de egy ismerősöm teljesen fel van buzdulva. Én nem vagyok ennyire optimista, hiszen még sokat kell várni, hogy kialakuljanak a végleges megoldások. Az már most biztos, hogy jól meg kell néznünk a vásárolni kívánt „hiperszuper mindent tudó készülékünket", melyre az is rá van írva, hogy tudja még az MPEG-4-et is. Melyiket a sok közül? A franciák is kifejlesztettek egyet, melyet felhasználva indítják be a francia DVB-T sugárzásokat. Mivel nem egységes ez a eljárás, lehet, hogy sokfajta set-top-box-ra lesz szükségünk, vagy egy hiper-szuperre, ha élvezni akarjuk egyszerre a különböző országok műsorait. Egyébként megszületett már az MPEG-7-es és az MPEG-21-es eljárás is 2001 és 2002-ben. Az biztos, hogy ha az angoloknak és a németeknek tökéletesen megfelel az MPEG-2-es szabvány, akkor azt még jó sokáig nem fogja felváltani semmi sem. Tehát nekünk elsősorban ezt az eljárást kell kívülről fújnunk, ha műsorunkat digitálissá kívánjuk tenni. Ez pedig nem más, mint az ISO/IEC 13818-as szabványsorozat. Most lássuk ennek érdekességeit.

A 7. ábrán láthatjuk az MPEG video tömörítés blokkvázlatát. Elemezzük csak egy kicsit.



7. ábra Az MPEG video tömörítés blokkvázlata [9]

A szaggatott téglalapban található három blokk végzi a kép kódolását. Ez ugyanazon az elven működik, mint a JPEG szabvány. A diszkrét koszinusztranszformáció egy 8x8-as méretű képpontnak a helytartományból egy ugyanakkora 64 tényezős frekvenciatartománybeli blokkba való leképzését valósítja meg. Ezután ezt a blokkot újrakvantáljuk, melyet kisérleti úton kidolgozott segédtáblák is segítik. Külön készült a luminanciajelre és külön a krominanciajelre is. Ezen segédtábláktól el lehet térni, de akkor azt is át kell vinni a demodulátor számára. Végül redundancia redukciót végzünk rajta. Itt is van lehetőségünk cikk-cakk, illetve opcionális cikk-cakk blokk kiolvasásra is, melyeket a Huffman-kódtábla is segíti. Ez a három eljárás a lelke az egész leképzésnek. A többi is nagyon fontos, hiszen a mozgásbecslő és a képtároló becslő végzi azt a nehéz munkát, amire igen nagy szükségünk van az átviendő jelsorozat csökkentése közben. Amiről még nem ejtettem szót, bár nagyon fontos az az első téglalap. Ez pedig a képcsoport átrendezés. Mivel tudjuk, hogy az egymást követő képek általában igen hasonlóak, ezért ha átvisszük az első képet, és utána csak a különbségi képeket kódoljuk, akkor alig kell átvinnünk valamit. Ezt fel is használjuk, de a kezelhetőség miatt kicsit módosítjuk.

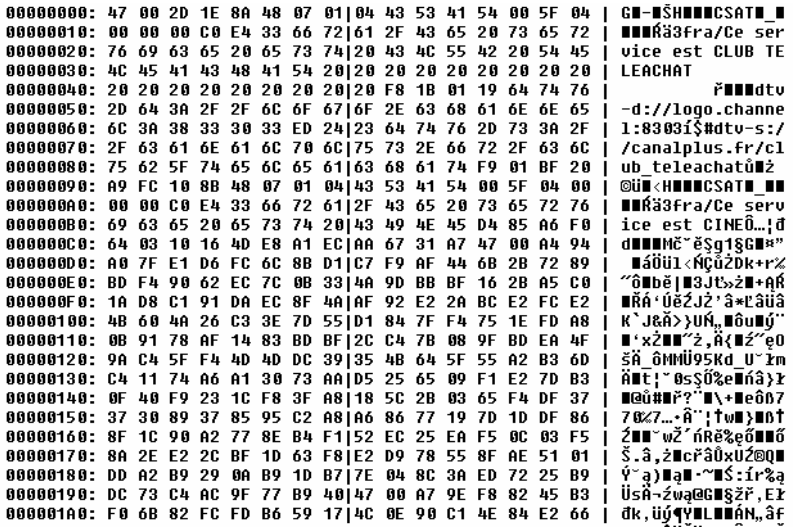


8. ábra Megjelenítési és kódolási sorrend

Így a képeket nem egyformán tömörítjük. Az I képeket egy az egyben, a P képeket csak az I-ből, míg a B képeket a P és az I képből származtatjuk. Tehát legalább négy képet kell tárolnunk, hogy vissza tudjuk alakítani a képeket. 12 képkockánként viszünk át egy I képet, ami elengedhetetlen a kép visszaalakításához. Tehát csatornaváltásnál lehet, hogy kell várnunk 11 képkockányi időt is, míg a kép láthatóvá válik.

Ezt ne felejtsük el, mert ennek még szerepe lesz az elkövetkezendőkben. Most már megvan a video jelfolyamunk, tehát már csak a továbbítás van hátra. Ilyenkor szoktuk azt mondani, hogy erősítjük, vivőre ültetjük, modulációs módot választunk és megfelelő teljesítménnyel kisugározzuk. Igen, ezt mind meg kell tennünk, de nem beszéltünk még arról, hogy milyen keretbe foglalva visszük át. Erre általában csak legyintünk, mondván azt

a műholdon már egyszer összerakták, és amúgyis ez csak adatátviteli kérdés már, az pedig az informatikusokra tartozik. Nem álthatjuk magunkat tovább, hiszen egy-két éven belül már nekünk kell összeraknunk a saját csomagunkat, és ehhez elengedhetetlen a szabvány pontos ismerete. Mindamellett nem elég csak a videojelünket beültetnünk pontosan a helyére, hiszen gondolnunk kell az újabb interaktív szolgáltatások iránti igény kielégítésére is. Tehát vonatkoztassunk el egy kicsit a televízió-műsor átvitelétől, és tekintsük a jelünket annak, ami valójában is. Digitális számok, egyesek és nullák, szabályos sorozatának. Esetleg HEXA formátumban:



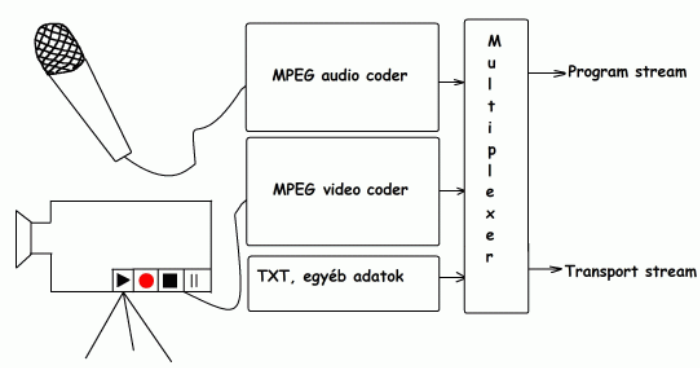
9. ábra Egy transport stream jelfolyam

Ez csak egy részlet abból az adatfolyamból, amiből mi ki tudjuk nyerni a videojelet, a hangokat, a teletexet, és az újabb szolgáltatásokat is…

3.3 A szabványosított csomagok

Az MPEG által előállított jelekre kétfajta csomagot is szabványosítottak, az egyiket „program stream”-nek nevezték el, míg a másikat „transport stream”-nek. Míg az elsőt olyan átviteli rendszerekre fejlesztették ki, ahol a csomagok változó mérettel rendelkezhetnek, azonos időalappal rendelkeznek az adatfolyamrészek és nem utolsó sorban zavarmentes átvitelre találták ki. Ilyen például a merevlemezes rögzítés. A másik módszernek az az előnye, hogy az adatfolyamrészek változó időalappal rendelkezhetnek,

állandó csomagméretek vannak, és elsősorban zavartól cseppet sem mentes átvitel céljára készült. Mint tudjuk, nekünk csak ilyen áll rendelkezésünkre. Ezek a műholdon, a kábelen és földfelszíni sugárzáson alapuló jelátvitelek. Tehát nekünk ezt kell szeretnünk minél jobban.



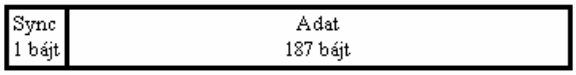
10. ábra Egy egyszerű videojel

Egy hagyományos műsor digitalizálását és adatcsomaggá alakulását láthatjuk a 10. ábrán. Egy mono hang, a videojel és a TXT jel átvitelét kell úgy megoldanunk, hogy az a vételi oldalon jól rekonstruálható legyen. Erre egyaránt alkalmas mindkét „stream", mivel nekünk az átviteli csatornánk zajos, így marad a kettőből egy, mégpedig a „transport stream".

4 A transport stream

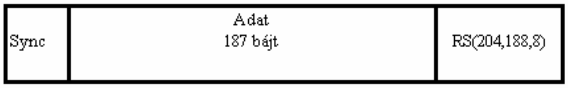
4.1 A transport stream általános bemutatása

A TS állandó csomagmérettel rendelkezik, ezt láthatjuk a 11. ábrán. A 188 bájtos struktúra első bájtja a szinkronbájt. Ennek értéke rögzített, és értékét még álmunkból felverve is kívülről kell fújnunk. Decimálisan 71, binárisan 0100 0111 és hexadecimálisan 47. Még ellenségemnek sem kívánom azt, hogy álmából ezzel az üzenettel keltsék: „Nagy baj van főnök, nincs meg a h47!". Hiszen ennél rosszabb nem is történhet. Mert ha nincs h47, akkor nincs csomag sem, azaz se kép, se hang, se adat.



11. ábra A transport stream keretszerkezete

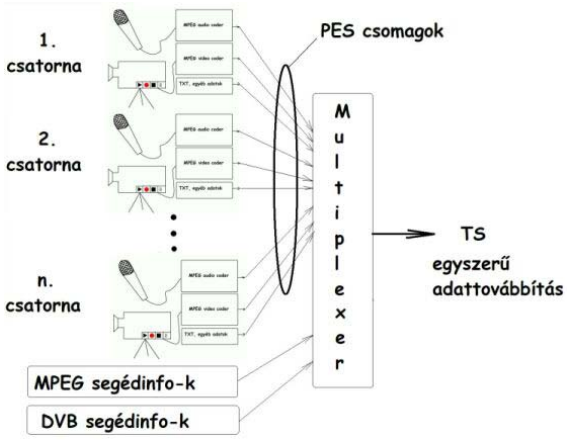
Mint azt már említettem, és még fogom is egy párszor, a zajtól meg kell védenünk a csomagunkat. Ezért kerül még a végére egy Reed-Solomon hibajavító kód, ez 8 többletbájtot eredményez.



12. ábra A transport stream kerete a hibajavító kóddal

Most már megvan a csomagunk, ami összesen 204 bájtot tartalmaz. De még mielőtt elkezdenék itt dobálózni különféle rövidítésekkel, mint pl.: PES, PMT, NIT, stb., nézzük meg azt, amit eddigi tudásunkból már ki tudunk következtetni.

Mielőtt nagyon belemerülnénk, azelőtt azt már tudnunk kell, hogy mi az a PES. Ez egy rövidítés, mely mögött a Program Elementary Stream áll. Ez az összefoglaló neve minden olyan jelfolyamnak, amit át akarunk vinni az előfizetőnknek. Legyen az video, audio, TXT jel, vagy bármilyen más adatsorozat. Hogy éppen mit is küldünk át, azt majd valamilyen módon jeleznünk kell. Erre több módszer is van, de először nézzük meg az összerakás blokkrajzát (10. ábra alapján).



13. ábra A TS multiplexálási folyamata

Azt már tudtuk, hogy nem egy digitális csatornát tudunk átvinni egy analóg csatornán, hiszen akkor nem érne sokat ez az új technika. Mennyi is az az n? Ha normális minőséget szeretnénk biztosítani, akkor 8 db műsort viszünk át. Ennél átvihetünk kevesebbet, így adatátvitelre is felhasználhatjuk a fennmaradó üres időszakokat, és többet is. Igaz, azt mondtam, a műsorunkat is tekintsük most adatnak, de az adatátvitel alatt most az egyéb interaktív szolgáltatásokat értem.

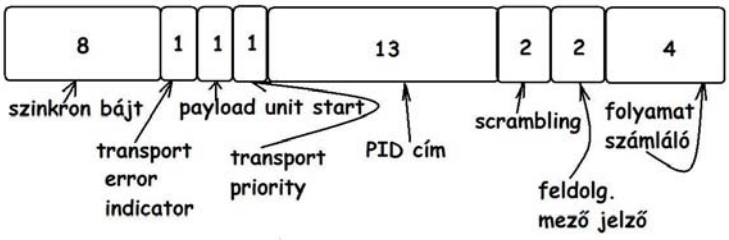
Nem csak ezeket a PES csomagokat látjuk az ábrán, hanem még két segédinfo dobozzal is kibővült a multiplexálásra kerülő adatok száma. Az elsőre azért van szükség, hogy a vételi oldalon össze tudjuk rendelni a hangokat a hozzájuk tartozó videojellel és más segédadatokkal. Ezek szerint csak úgy nem sikerülhet.

A DVB segédinformációk továbbítására azért van szükség, hogy a rendszerünket vezérelni tudjuk üzem közben is. Képet ad nekünk az átviteli út terheltségéről, az esetleges átkapcsolásokat is segíti. Mostanság nagy divat az, hogy egy csatornán napközben gyerekfilmeket sugároznak, majd este átváltanak egy másik adásra. Ez az átkapcsolási folyamat sok időbe telik, mely jelentősen csökken, ha figyeljük a DVB segédinformációkat.

Amiről még nem tettem említést, az a csomagazonosítás. Azaz, hogy milyen módon bélyegezzük meg a csomagunkat. Ezt néhány bájt segítségével valósítjuk meg, ezeket nevezzük a csomag fejrészének. A következőkben nézzük meg ezt kicsit közelebbről.

4.1.1 A transport stream fejrésze

Mint azt az előbbiekben már említettem, van néhány olyan adat, melyek segítségével meg tudjuk különböztetni a csomagokat egymástól. Ezeket az adatokat helyezzük el a TS elején, és ezeket nevezzük a csomagok fejrészének [1].



14. ábra A transport stream fejrésze

Természetesen a legfontosabbal kezdődik, a h47-es szinkron bájttal. Utána következik, mint ahogy az a 14. ábrán is látszik, a csomag érvényesítő, angol nevén a transport error indicator. Ez 0 értékű, ha a csomag jó és feldolgozható. A program unit start indicator, vagy magyarosan program egység kezdetjelző, mely 1-es, ha PES, vagy PSI kezdődik. A PES-t már ismerjük, de a Program Specific Information-t még nem. Erről később még fogok eleget írni, most csak annyit árulok el, hogy ez az összefoglaló neve azoknak a tábláknak, amelyeket azért kell átvinni, hogy a vételi oldalon sikeres legyen a dekódolás. A „transport priority" bitet nem használják sűrűn, hiszen feladata az lenne, hogy jelezze azt, hogy éppen elsőrendű adat jött, azaz video, kéretik ezt különösen kezelni, erre nagyon kell vigyázni. Mivel ezt nem használják, vagy csak alig, ezért ez általában 0.

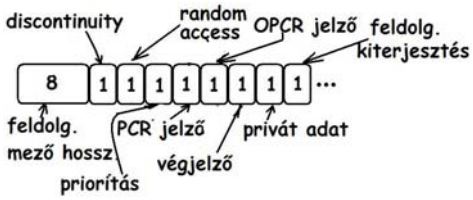
Elérkeztünk most már ahhoz az egységhez, amiért belekezdtem a fejléc tárgyalásába. Ez pedig a PID cím. Azaz Packet IDentifier, magyarul csomag azonosító. Ez egy 13 bites bináris szám, mellyel megbélyegezzük minden egyes csomagunkat. Persze az összetartozó jelfolyamoknak egy címük van. Tehát, ha ki akarjuk nyerni például egy rádiócsatorna jeleit a TS-ből, akkor elég figyelni a csatorna PID-jét, és ha ilyen érkezik, akkor át kell irányítanunk a dekódolóra és máris élvezhetjük a rádióműsort. A következő két bit igen fontos, hiszen ez mondja meg nekünk, hogy az adott PID-ű csomag kódolva van-e. Hiszen ebben a világban csak úgy lehet tévé díjat beszedni, ha korlátozzuk a hozzáférhető programok számát, az érdekesebbeket, a nézettebb csatornákat lekódoljuk, és csak egy bizonyos havidíj befizetése után adjuk a fogyasztónak fogyasztható állapotban.

A csomagok beérkezési sorrendje fontos, hiszen a dekódolás során a csomagok sorba rendezése nagyon fontos. Ezt segíti a fejrészünk végén található folyamatosság-számláló. Ez a számláló az azonos PID-ű csomagokra egyesével lép. Amiről viszont nem beszéltem még, az a feldolgozó mező jelenlétét jelző, angol nevén adaptation field control. Ha x1-es értékű, akkor található hasznos adat az aktuális csomagban. Ha 1x értékű, akkor van a csomag elején feldolgozó mező. Azaz 11 esetén a csomag elején feldolgozó mező van, utána pedig folytatódik az adatfolyamunk.

Mielőtt még bemutatnám a PID összerendelés mehanizmusát, nézzük meg a következő fejezetben a feldolgozó mezőt kicsit közelebbről.

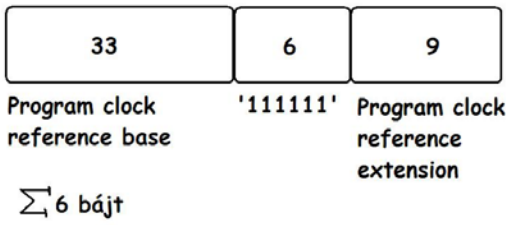
4.1.2 Az Adaptation field

A rendszer helyes működéséhez elengedhetetlen dolog olyan információk átvitele, amellyel az adó körülményeihez igazítjuk a vételt. Ezt a dolgot tartalmazza a feldolgozó mező [1]. Most nézzük meg, hogy mire is van szüksége a vevőoldalnak.



15. ábra Az adaptation field egy részlete

Mint ahogy az a 15. ábrán látható, először megtudhatjuk a feldolgozómező hosszát. Ha csak feldolgozó mezőt tartalmaz csomagunk, azaz, ha a feldolgozó mezőt jelző bitek b10 –ás értékűek (Ugye nem felejtettük el a fejrészt?!), akkor a discontinuity bitünk 1-es, azaz a fejrészben a folyamatosság számlálót nem szabad növelni az adóoldalon, a vevőoldalon meg nem szabad meglepődnünk, ha értéke nem, nőtt 1-gyel. A random access, azaz a véletlenszerű hozzáférést jelző bit jelzi, hogy a legközelebbi ilyen PID-del rendelkező csomag fejrésze után következő első byte-ja a csomag tartalmáról ad felvilágosítást. Itt is találunnk prioritásjelzőt, melynek 1-es értéke jelzi, hogy videojel feldolgozó mezőjét látjuk. Ha a következő bit egyes értékű, akkor nagyon boldog lesz a helyi oszcillátorunk. Hiszen akkor a 15. ábrán található végjelző („…”) után következik a Program Clock Reference. Ennek szintaktikáját a következő ábra szemlélteti [1].



16. ábra A Program Clock Reference

Mint azt a 16. ábra is jelzi, ez egy 6 bájtos adat, mely az adó helyi oszcillátorának állapotáról ad felvilágosítást. Mivel a helyes vételhez azonos órajelen kell működnie az adó órajelének és a vevő órajelének, ezért időközönként szinkronizálni kell a vevőnket az adó

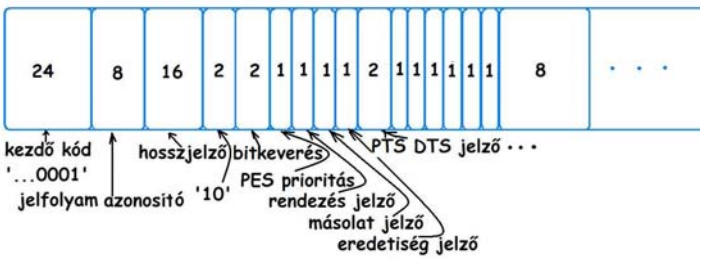
oszcillátorához. Az adó oszcillátora sokkal pontosabb és stabilabb, mint azt egy vevőkészülék oszcillátorától elvárnánk. Diplomamunkám fő része a PCR ismertetése, azaz a szinkrozinálás megoldása, annak hibája, ezért szeretném ezt a témát később folytatni, hiszen oda jobban illik és ott sokkal részletesebben ki tudom fejteni. Most legyen elég csak annyi, hogy ez a 33+9bites szám vezérli a vevőkészülékünk szinkrozinálását.

A következő bit jelzi az OPCR jelenlétét, ami sokkal ritkábban található meg a jelfolyamban, hiába az eredeti PCR nevet viseli. Ennek jelenléte tényleg nagyon ritka, az oszcillátorunk (adó és vevő együttes) lassú elmászását igyekszik visszaállítani. Az erre rákövetkező bit jelzi, hogy van-e hátralévő csomagjelző a feldolgozó mezőben. Utána már csak a privát adat jelző van, amely használata nem specifikus, bár ajánlások találhatók rá. Ha van a feldolgozó mezőnek kiterjesztése, akkor a következő bit 1-es értékű. A kiterjesztett mezőben többek közt az összeillesztési sebességről kapunk információt, azaz, hogy milyen mintavételezési sebességgel történt a felvétel az adott műsorról.

Most már azt is tudjuk nagyjából, hogy mire is jó a feldolgozó mező. A pontos bájtsorrendről és egyébb megjegyzésekről találhatunk néhány oldalt a függelékben, most a további elemzést mellőzve nézzük meg, hogy milyen információkat tartalmaz a PES csomagunk a hasznos videojel mellett.

4.1.3 A Program Elementary Stream

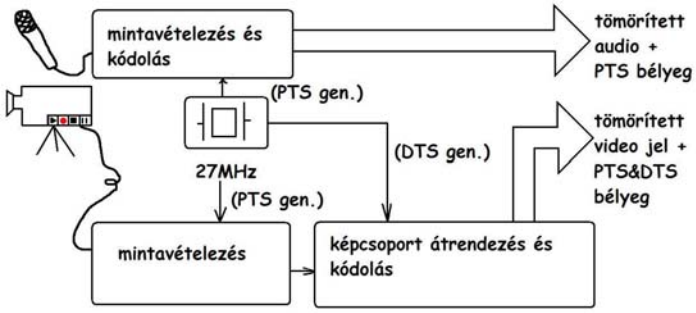
Persze PES adat nem minden csomagban van, hanem egy PES csomag egy adat blokkhoz tartozik. Így egy bizonyos adatmennyiséghez, például videojel esetén képenként. A tábla rövid, de minden bitet bemutató leírását ugyancsak a függelékben megtalálhatjuk. Itt röviden, de annál inkább a lényegre rávilágítva szeretném megmutatni ennek hasznosságát, jelzésének felismerését és felhasználását [9].



17. ábra A PES keretének egy részlete

A 17. ábra nem mutatja a teljes felépítést, azt azért mutatja, hogy a PES kezdetét jelzi a 3 bájtos kezdetjelző, majd a következő bájt jelzi az aktuális adat fajtáját. Lehet ez adat, hang, TXT jel video, meg amit csak el tudunk képzelni. A hosszjelző a PES csomagunk hosszát jelzi, mármint a kitöltő adatok hosszát. Adatunk kódoltságát itt is 2 biten jelezzük, majd a prioritásjelző következik, amit ugyancsak nem használunk. A rendezés jelző megmondja, hogy található-e a TS-ünkben jelfolyam rendezést segítő, leíró mező. A következő két bit jelzi, hogy az adat az eredeti, vagy csak másolat, azaz ha eredeti, akkor kódold le nyugodtan, de ha másolat, akkor lehet, hogy ez csak egy blokk újraküldése vagy csak egy próba adás…

A következő két bit nagyon fontos, hiszen ez jelzi, hogy PTS, DTS vagy mindkettő van-e a teljes PES fejrész után. Mi is az a Presentation Time Stamp és a Decoding Time Stamp? Mire használljuk és miért van rá szükségünk? Mielőtt ezt leírnám, térjünk csak egy kicsit vissza az MPEG kódolási mehanizmusához.



18. ábra PTS, DTS keletkezése

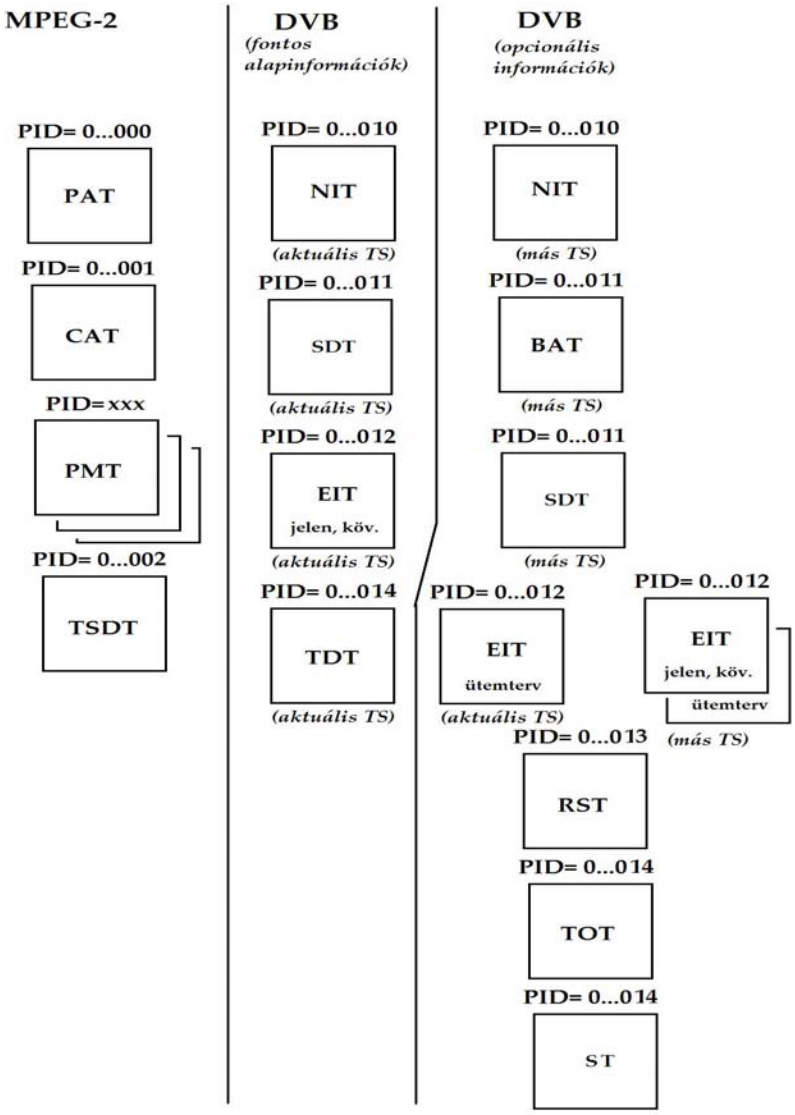
A 18. ábra azt szeretné bemutatni, hogy hogyan is készül a PTS, azaz a Presentation Time Stamp és a DTS, azaz a Decoding Time Stamp. Az ábrából látszik, hogy audio jel kódolásánál csak PTS keletkezik, hiszen itt a hangot egymás utáni sorrendben kódoljuk és küldjük ki. Viszont a videojel kódolási mehanizmusa más, hiszen a megjelenési sorrend és a kódolási sorrend nem azonos. Talán még emlékszünk, vagy ha nem, akkor a 8. ábrára rápillantva észrevesszük az I, P és B képek sorozatának megváltozását. Mivel a kódoló is 27MHz-en „rezeg”, ezért rajzoltam a 27MHz-es oszcillátort. Általában a műsor kódolása így történik, de ha nem, akkor a PTS és DTS időket az adási oldalon a 27MHz-es órajellel regenerálni kell. Tehát a PTS, DTS jelzőbitből ugyancsak megmondhatjuk, hogy milyen

fajta adatról van szó. Ha viszont nem találunk ilyen jelzést, akkor biztosan adat vagy TXT jelfolyamot tartalmaz az adott PID-ű csomag. Bár PTS bélyeget találtam TXT adatfolyamban is. A PTS és DTS nem meglepő módon ugyancsak 33 bitből állnak. Az előbb láttuk, hogy az alap referenciaórajel is 33 bitnyi, az azonosság nem meglepő. A PTS, DTS értékeket a PCR base értékéhez viszonyítjuk. Mivel ezek pontatlansága nem annyira zavaró, így nincs kiterjesztésük. Egyébként 90kHz-es lépésekben változik az értékük, ugyanúgy mint az alap PCR-nek. Egy bitnyi hiba kb. 10μs-nyi késedelmet jelent a kép és hangmegjelenítésben. Amit talán nem veszünk észre, ezért is elégedhettek meg a szabványosítók ezzel a pontossággal. A következő jelzőbitek jelzik a PES mező további információit. Így például itt is küldhetünk PCR-hez hasonló Elementary Stream Clock Reference-t, de definiálhatjuk a PES jelfolyam beérkezési sebességét, a videojel kiolvasási sebességére adhatunk utasítást, CRC ellenőrzést jelezhetjük, és a kiterjesztett PES-t is megmutatja. Ezután következnek a jelzett adatok tovaküldése, majd pedig a kiterjesztésről kapunk újabb jelzéseket. Ezek, mint már említettem, a függelékben megtalálhatóak.

4.1.4 A további táblákról röviden

Mielőtt továbblépnénk, tartsunk egy kis összegzést. Most már ott tartunk, hogy megvan a videojelünk, az audio jelünk és elő tudjuk állítani a TXT jelünket is, sőt bármilyen más adatjelet is, csak a megfelelő eljárást kell ismernünk. Ha ezzel megvagyunk, már csak meg kell címeznünk csomagjainkat. Videojelet tartalmazó csomagjainkat el tudjuk látni PTS, DTS, az audiokat PTS bélyegekkel. Talán már át is vihetnénk. Igen ám, de ne feledjük, hogy egy TS egy analóg csatornát foglal el. Viszont egy TS több műsort és más adatot is képes átvinni. Honnan fogja tudni a vevőkészülékem, vagy a túloldalon lévő program szerkesztő, hogy melyik videojelhez melyik audio jel tartozik. Hiszen a PES csomag azt azért elárulja nekünk, hogy melyik adatot melyik dekódolóra kell küldenünk. Legyen tehát egy táblánk arra is, ami elárulja nekünk a jelek közti kapcsolatot. Nevezzük el csak úgy magyarosan program-térkép táblának. Angol neve legyen Program Map Table, rövidítése meg PMT. Már kész is vagyunk. Persze csak akkor, ha megmondjuk, hogy melyik PID-ű csomagok tartalmazzák az összerendelés szabályait. Hiszen 8-10db PMT tábla kell ahhoz, hogy leírjuk az adott TS-ünket. Arról nem is beszélve, hogy műholdas

átvitel esetén kb.: 56*8-10db PMT tábla kell. Mi lenne, ha a PMT tábláinkat is egy csokorba összefognánk, és az így keletkező új táblánknak egy állandó táblacímet adhatnánk, pl. a bináris 0-át. Mi legyen a neve? Legyen program társító, vagy egyesítő tábla. Angolul legyen Program Association Table, röviden PAT. Jaj, de lekódoltunk több csatornát is, mi lesz most? Annak is kell egy tábla, a PAT-hoz jól csengő név kellene, legyen CAT, Conditional Acces Table. Persze kell neki egy magyar név is, hiszen nekik fontos a saját név. Legyen feltételes hozzáférést biztosító tábla. És a táblák csak szaporodtak ily módon, lettek többen és még többen. Jelenleg annyit tartanak számon, amennyit a 19. ábra megmutat nekünk [3] [6].



19. ábra A TS táblái

Persze nem csak ezek a táblák létezhetnek, és ezek közül sem kell jelen lennie mindegyiknek. Az első oszlopot már végigmeséltem, nézzük a második oszlopot. A NIT táblában az aktuális hálózatról kaphatunk felvilágosítást, a modulációról kaphatunk értékes információt, ez tartalmazza az adó hivatkozási számát és még sok más dolgot is. Az SDT-t talán forgalomirányító táblának nevezhetnénk, ez felel többek közt a „dugó mentesítés”-ért. Az EIT talán elhagyható, de ez tartalmazza az adott programok tartalmát, mutatja nekünk, hogy a most következő filmhez van felirat is, vagy több nyelv is választható. Így már sokkal könnyebben felkészülhetünk a végponton ezen más beállításokra. Emlékszem, kisebb koromban nem értettem, hogy az Eurosport egyszer angolul egyszer meg magyarul szólt. Most már tudom, hogy a kábeltelevíziós szakemberek erősen figyelték az EIT-t és örömmel átkapcsolták a hangot egy-egy magyar közvetítés erejéig. A TDT táblából kinyerhetjük az aktuális időt is, ezzel végéhez is értünk a rövid ismertetésnek. A harmadik oszlop nem szükséges, de például a TOT segítségével megadhatjuk az aktuális idő és a megjelenítési idő közötti különbséget, ezzel vezérelve a megjelenítés, a műsorkezdés idejét. Ne feledjük, hogyha később kezdjük a vetítést, és van elég nagy méretű memóriánk, akkor egyes rendszerbeli hibákat észre sem veszi a fogyasztónk. Megvalósíthatunk automatikus eseménykapcsolást is, ha van RST-nk. Átküldhetünk több órás programtervet is, vagy más TS-en futó műsor-ismertetőket is. Az EIT lehetőséget ad rá. Ami itt nincs felsorolva, az a DSM-CC, ami az átmeneti tár vezérlő és felügyelő üzeneteit tartalmazza. Amiről még nem tettem említést, az a BAT, azaz Bouquet Association Table, magyarul a csomagazonosító tábla. Az SDT táblában megtalálható az adott program neve (CNN, Travel…)

Persze az az adott rendszertől függ, hogy milyen táblákat tartalmaz, mely táblákat tart fontosnak. Ami biztos, hogy PAT, PMT és PES táblát biztosan találunk TS-ünkben. Most nézzük meg azt, hogy milyen kapcsolat is van közöttük, hogyan is kell az összerendelést megvalósítani.

5 A programfa elkészítése

Mivel diplomatémám fő része az a PCR mérésének egy lehetséges megoldása egy felhasználóbarát program segítségével, ezért első lépésben el kellett készítenem egy

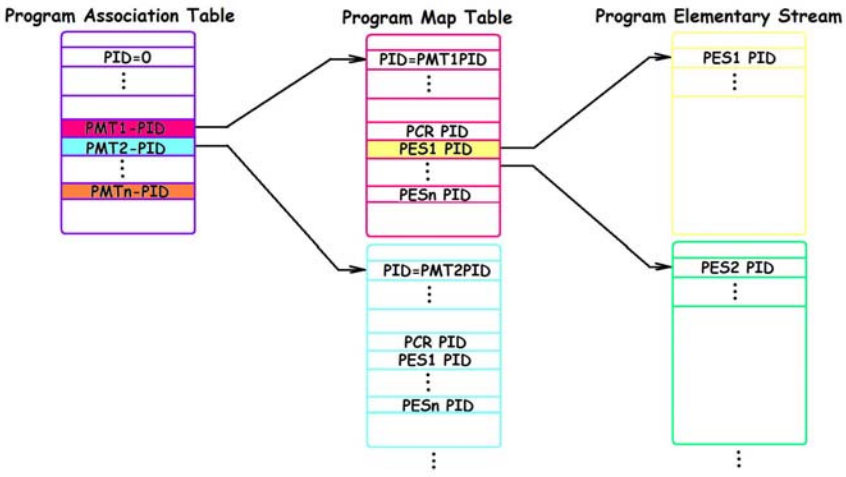
programfa analizátort. Persze az is használható megoldás, hogy kikeressük az adott műhold adott TS-ének a programtervét, hiszen léteznek ilyen újságok, biztos mindenki látott már. Minek szenvedjünk a keresgéléssel, mikor ott van előttünk, csak ki kell szedni belőle. Azt meg tudjuk, hogy kell, vagy ha nem, akkor közösen kitaláljuk.

Első lépés talán a legnehezebb. Ki kell választani a legmegfelelőbb programot. Előtanulmányaim során Megismerkedtem a C nyelvvel, és Pascalban is szereztem tapasztalatokat. Ma már egy DOS felhasználófelülettel rendelkező programot eladni nem lehet a Windowshoz szoktatott felhasználóknak. Viszont grafikus teret készíteni nagyon nehéz ezekkel a programokkal. A programozó társadalom is váltott. Olyan programokat kaptak kézhez, melyekkel könnyen lehet felhasználóbarát felületet létrehozni, mégis megmaradtak a régi programnyelvek. Az egyik ilyen a Visual C, a másik pedig a Delphi család. Habár a főiskolán C-t tanítottak, én mégiscsak Pascalban nőttem fel, így hozzám közelebb áll a Delphi. Választásom a Delphi 7-re esett.

A kiválasztás megtörtént. Most át kell látni a megoldani kívánt problémát, meg kell rajzolni a program folyamatábráját, utána már csak meg kell írni. Igen egyszerűnek tűnik.

5.1 A feladat rövid áttekintése

Mielőtt részletesen megnéznénk a PAT és PMT táblák felépítését, előtte rajzoljuk fel azt, amit már tudunk.

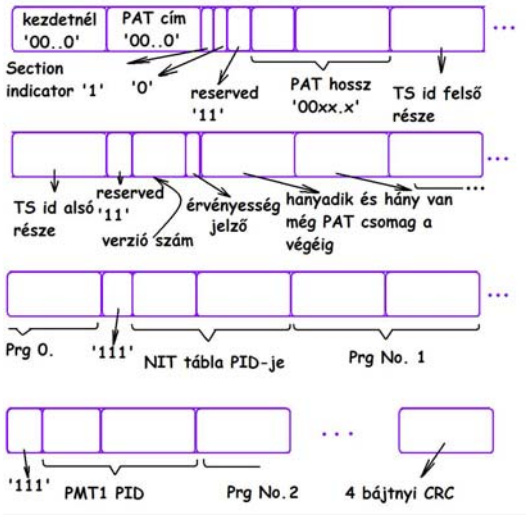


20. ábra A programfa struktúrája

Mint az az ábrán is jól látható, nem is olyan nehéz a feladat. Hiszen elég kiszűrni a PAT táblát, kinézni belőle a PMT táblák címét, ha már ezzel megvagyunk, akkor a PMT csomagokból már ki tudjuk olvasni az egymáshoz tartozó PID-eket. Közben nagyon sok olyan töltelék információt kell figyelembe vennünk, amelyeknek értéke érvényesíti az információt. Ahhoz, hogy ezeket fel tudjuk mérni és bele tudjuk venni a feldolgozó program eljárásába, ismernünk kell a táblákat közelebről. Mielőtt tehát a program folyamatábrájához hozzákezdenénk, ismerjük meg a két táblát közelebbről is.

5.1.1 A PAT tábla

Az első és legfontosabb tábla a PAT, persze ha lehet egyáltalán ilyet mondani egy olyan jelfolyamra, ahol minden táblára egyformán szükség van. [1]



21. ábra A PAT felépítése

A számunkra legfontosabb bájt a második, mely ismételten megmondja nekünk, hogy ez egy PAT tábla. Ne feledjük, hogy erről már a fejrészben, a PID értékéből is következtethetünk. Ezután a PAT hosszúságáról árulkodik a 3. bájt alsó négy, és a 4. bájt bitjei. Ezek megmondják, hogy hány bájt hosszúságú a PAT. A következő két bájt tartalmazza a TS-ünk nevét melyet tekinthetjük a TS legfontosabb azonosítójának. A 7. bájt, vagy inkább a 11., hiszen a PAT-nak is van 4 bájtnyi fejrésze, tartalmazza a

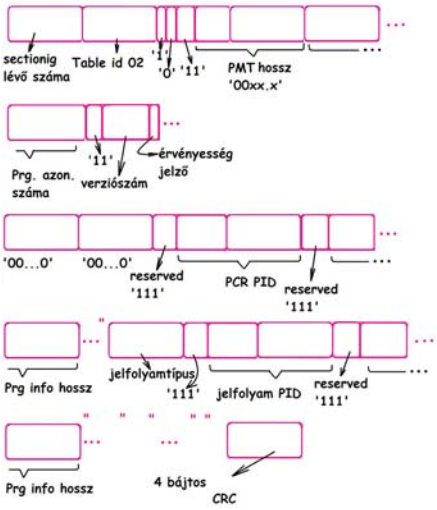
verziószámot, ami minden változásnál eggyel nő. Itt található az érvényességjelző, amely a PAT-ot érvényesíti. Ha ez egyes, akkor hihetünk a PAT-nak, amit tartalmaz, az aktuális. A version number az állandó felügyeleti rendszerek számára fontosak, mivel ezek az automata rendszerek a verziószám figyelésével segítségével azonnal tudja frissíteni adattábláját. A 12. és 13. bájtot ritkán használjuk, így értékük általában 0. Használatuk akkor lenne célszerű, ha PAT táblánk nagyobb lenne, mint 188 bájt, de mivel ez általában nem teljesül (egy 8 programos TS 65 bájtnyi PAT-ot tartalmaz), így találjuk mindig 0-ának. Persze ez nem jelenti azt, hogy erre nem lehet szükség a jövőben, mikor erősen adatátviteli rendszerként is fog működni TS-ünk. Utána két bájt 0, majd megkapjuk a NIT tábla PID-jét, kissé feleslegesen, hiszen mint tudjuk ennek értéke kötött. Ezután már csak egyszerű dolgunk van, hiszen 4 bájtos ciklusban képet kapunk a programunk nevéről és PMT táblájának PID-jéről. A végén 4 bájtos CRC ellenőrzőt tartalmaz.

Végezzünk a végén egy kis fejszámolást. 188 bájtból vonjunk le négyet a fejrész miatt, négyet a CRC miatt és még tizenhármat a PAT alapinformációiért. Így maradt 167 bájt, ami 41 program tárolására, rögzítésére ad lehetőséget. Mit ez a gyors fejszámolásból is kitűnik, ezt a nagy mennyiségű helyet az analóg csatornán továbbított tévé és rádiójelek nem tudják kitölteni. Persze a közeljövő interaktív szolgáltatásai, illetve a kábeltelevíziós hálózat digitális adásának TS-e már képes lehet ezt túlszárnyalni. A PAT hossz értéke maximum 1021 lehet, ez 5 csomagot képes megtölteni. Azaz több, mint 200 PMT táblánk lehet.

Ennyit a PAT-ról, ezért most nézzük meg a PMT táblát is, melynek a címeit megkapjuk a PAT tábla feldolgozásából.

5.1.2 A PMT tábla

Az előző fejezetben megtudtuk, hogy miképpen keressük meg a PMT tábláink PID címeit. Ha megkeressük az ilyen PID címmel rendelkező csomagokat és feldolgozzuk őket, akkor megtudhatjuk az összetartozó csomagok PID címeit, és a csomagok fajtáit. Ehhez ismernünk kell a PMT felépítését. Tekintsük most át ezt. [1]



22. ábra A PMT általános felépítése

Természetesen, mint minden tábla, ez is fejrésszel kezdődik, mely elárulja a címét, és árulkodik arról is, hogy tartalmaz-e adaptation field-et. Ha nem, akkor a fejrész után a 22. ábra alapján folytatódik. Itt is a második bájt fontos számunkra, hiszen ennek 02-es értéke mondja nekünk, hogy ez a csomag egy PMT. Az állandó és a még fel nem használt bitek után megkapjuk a PMT hosszát 10 biten ábrázolva, mely a hosszt bájtban tartalmazza. Ezután a kövatkező két bájt a program azonosító számát tartalmazza. Általában ezzel a számmal szoktunk az egyes programcsomagokra hivatkozni a TS-be építése előtt és közben. Mondjuk, a PAT-ból kiolvastuk, hogy a 12722-es programnak a PMT pidje 167-es. Ha mi a 167-es csomagot dolgozzuk fel, akkor az itt kapott számnak 12722-nek kell lennie, ha nem, akkor valami gond van, és a csomagot tovább már nem kell néznünk. Ha egyezik, akkor bátran továbbmehetünk, és ha az érvényességi bit is 1-es, akkor azonnal megkaphatjuk a PCR információt tartalmazó PID számát. Mivel a PCR elengedhetetlen a helyes dekódoláshoz, ezért minden programcsomag egyik dekódolásra váró jele tartalmazza ezt. Általában a legnagyobb prioritással rendelkezőhöz csapják hozzá ezeket a vezérlőbiteket. Tehát TV műsor esetén a videojel, míg rádiócsatornánál az audiojel tartalmazza. Ezáltal a PCR PID-je videocsatorna esetén megegyezik a videojel PID értékével, rádióprogram esetén pedig az egyik audiojel PID értékével. Ezután a program információ hossza és a programinformáció következik. Ezeknek az információknak is megvan a saját struktúrájuk, ami megtalálható az IEE/ISO 3818-1-es szabványban. Az információk lehetnek videojel esetén a videojel formátuma (4:3-as vagy 16:9-es),

tömörítési arány és egyéb, a dekódoláshoz szükséges adat. Audiojel esetén a tömörítési arány, mono vagy sztereo jelet küldünk, angol, magyar vagy más nyelvű az adott aodiojel, stb. A prog. info utáni bájt elmondja a következő bájtokban megadott PID-ű csomagok fajtáját. Mivel ezek csak számok, ezért készítettek egy táblázatot a szabvány kigondolói, és az IEE/ISO 3818-1-es szabvány 2.36-os táblázata alapján megtudhatjuk a számok mögött rejtőzködő lényeget. Az 1-es számjegy MPEG-1-ben kódolt videót, a 2-es MPEG-2-es videót, míg a 3-as és 4-es számok audio jelet takarnak. Általában a 6-os szám takarja a TXT jelet, habár ezt másra is fel lehet használni. A szabvány ezt privát adatként tartja számon. Az idő úgy hozta, hogy ez lett a teletext jelölője. Miután megtudtuk, hogy milyen jel PID-jét is kapjuk meg, megnézhetjük a PID értékét is, majd átugorva a speciális információkat (ezekről beszéltem az előbb, ezek a descriptorok, melyek az adott jel egy-két fontos jellegzetességét tartalmazzák) megtudhatjuk a következő jel típusát és címét. Ezt egészen addig kell folytatnunk, amíg el nem érjük a PMT hossz által megadott távolságot. Ha nem sikerül a végére érnünk az adott csomag 188 bájtja alatt, akkor meg kell keresnünk a következő ugyanilyen PID-ű csomagot, majd annak fejrésze után onnan folytathatjuk az elemzést, ahol az előbb abbahagytuk. Ebben a csomagban már nem lesznek a PMT-t jellemző információk (PMT hossz, programcsám, érvényességjelző…), hanem csak tisztán az előző csomag folytatásaként kell kezelnünk.

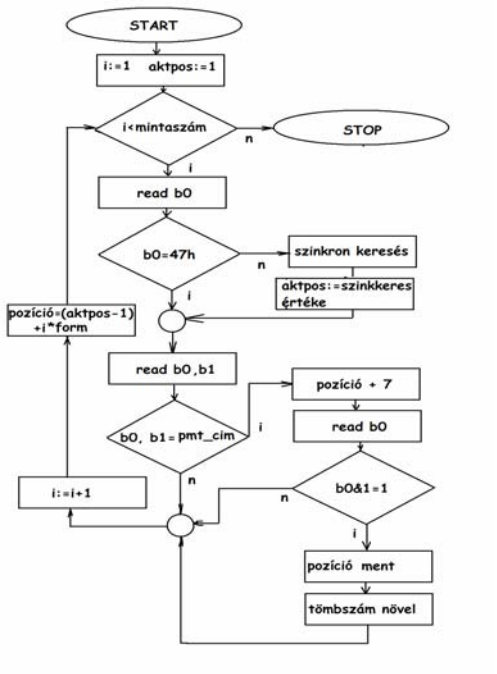
Ha ezt a műveletet a PAT-ban megtalált összes PMT táblára elvégezzük, akkor megtudjuk az összes program összetartozó csomagjainak PID címeit.

Ezzel meg is ismertük azokat a táblákat, amikre szükségünk van a programfa kirajzolásához. Lépjünk a következő lépésre, és alkossuk meg a feldolgozás folyamatábráit. Most az előbb megismert táblák érvényességi bitjeit, illetve az állandó bájtokat kell vizsgálnunk, melyek aktuálissá teszik a tábláinkat, ezután ki kell olvasnunk a számunkra fontos információkat.

5.2 A PAT-ok megkeresése

A TS csomagunk egy ethernet átalakítón keresztül a számítógép egy bizonyos memóriatömbjébe elmenthető. Az elmentett TS jelfolyam első bájtja kapja a 0-s pozíciót. A PAT táblák feldolgozása előtt létrehozunk egy PAT_CIMEK nevű tömböt, amibe

elmenthetjük a PAT csomagok kezdő pozícióit, azaz azon csomagok pozícióit, melyek PID-je 00h és átmentek az érvényességi vizsgálaton.



23. ábra PAT keresés folyamatábra

A kezdőszámot állítsuk egyre, majd kezdődhet is a keresés. A mintaszám alatt a feldolgozandó adatok 204-ed (vagy 188-ad, ha 188-as csomagokat dolgozunk fel) részét értjük. Le kell futtatnunk eljárásunkat a mintaszám által mutatott értékszer. Beolvassuk az első számot a TS-ből, és ha ez nem a szinkronbájt, akkor el kell ugornunk a szinkronkeresést végző eljáráshoz, ami megkeresi nekünk a csomagkezdetet. Megvan a szinkron, beolvassuk a következő két bájtot, melyből kivesszük az utolsó 13 bitet, melyeknek értéke dönti el a következő elágazás végét. Ha a beolvasott bitek értéke nem 0, akkor ez nem egy PAT csomag, tehát i értékét növeljük meg 1-gyel, és ugorjunk a következő sor elejére. Ellenkező esetben ugorjunk az érvényességjelzőhöz, majd ha ez is 1-es, akkor elmentjük a csomag kezdő pozícióját a PAT_CIMEK tömbbe. Mire lefut a program, az összes PAT kezdőcímét megtalálhatjuk a tömbünkbe. Erre most még nincs szükség, hiszen 1 TS-nek jelenleg csak egy aktuális PAT-ja van. Így a megtalált PAT táblák közül elég egyet feldolgoznunk, a többi PAT lényegi tartalmával egyezni fog.

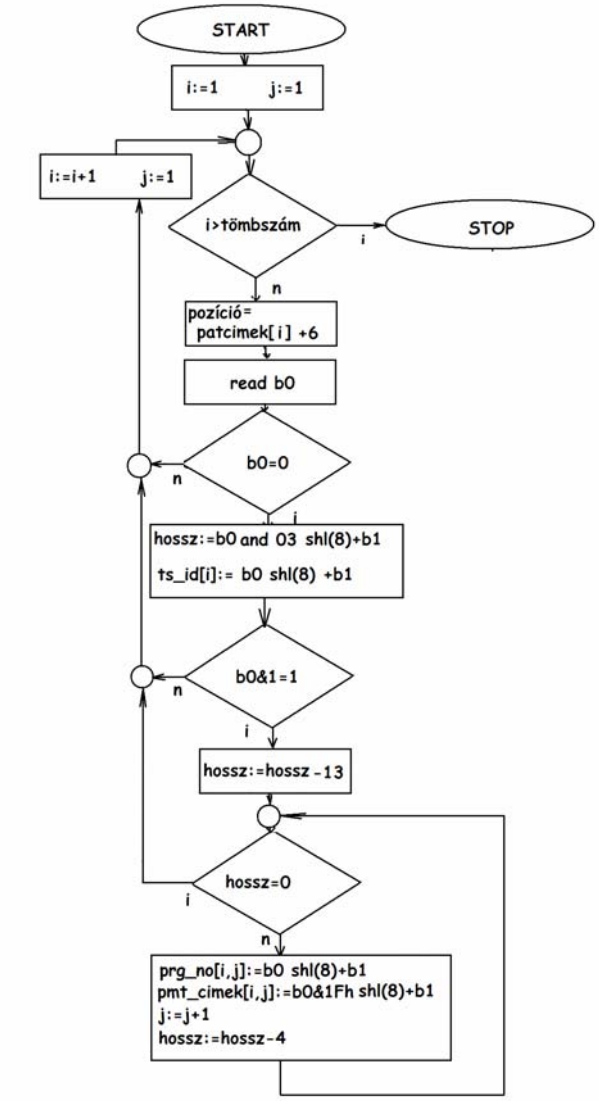
Viszont a jövőben erre még szükség lehet, ha az adatátviteli felhasználás előtérbe kerül, lehet, hogy nem fogunk beleférni egy csomagba.

Az eljárás lefuttatása után megvan a PAT csomagunk kezdőcímei, melyből egyet kiválasztva elkezdhetjük a feldolgozását.

5.3 A PAT feldolgozása

Most egy ciklust kell szervezni, ami kiveszi a PAT csomagból a PMT PID értékeit és a program azonosító számát. A feladat egyszerű, hiszen létrehozunk két kétdimenziós tömböt, melyekbe elmentjük majd a program táblák neveit és PID-üket. Kétdimenziósnak azért terveztem, hogy igény szerint több PAT csomag feldolgozására adjon lehetőséget a program. Ennek akkor lehet értelme, ha az elsőként kiválasztott PAT csomagunk igaz, hogy átment az érvényességi vizsgálaton, mégis hibás bájtokat tartalmaz. Több PAT feldolgozása és összehasonlítása kiszűrheti a hibásan feldolgozott adatainkat.

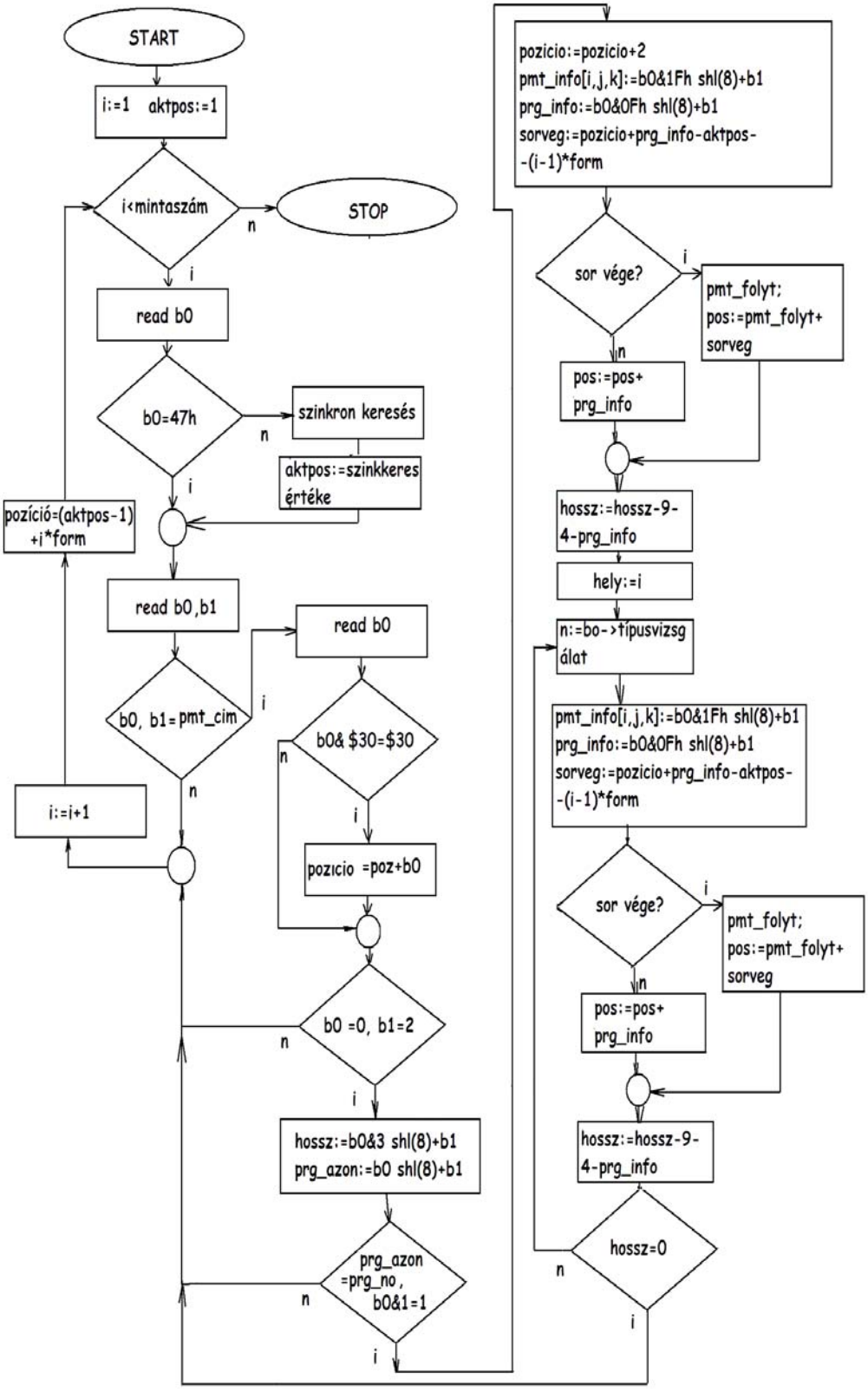
A pozíciót beállítjuk a fejrész utáni 2. bájtra (a PAT_CIMEK nevű tömbből kiolvasva) és megnézzük ennek az értékét. Ha nulla, akkor tényleg PAT-ot találtunk, és folytatjuk tovább a feldolgozást. A következő két bájt megadja a csomag tartalmának hosszát, amit elmentünk. Azon nyomban kiolvassuk a TS címét, mely a TS-ünk legfontosabb azonosítója. A következő bájtban található az érvényességjelző, aminek érvényességi vizsgálata után a hosszból kivonunk 13-at. 13-at, mert a kiolvasott TS_ID két bájt nagyságú, hét bájt elment a vizsgálatokra, és a csomag végi négy bájtos CRC-t is itt vonjuk ki. Megtehetjük, hiszen ezzel a lépéssel a programunk még véletlenül sem fog hozzányúlni a CRC bájtokhoz. Ezzel elérkeztünk egy következő ciklusig. Amíg a hossz nem lesz 0, addig kiolvassuk a két bájtos program azonosítót és a két bájtos PID-címet. Vonjunk le négyet a hosszból és ugorjunk a ciklus elejére. Ha mindent kiolvastunk (hossz=0), akkor ugorhatunk a következő PAT csomag elejére (ez nem fontos, de ellenőrzésnek jó).



24. ábra PMT-k megkeresése

5.4 A PMT feldolgozása

Most már csak a PMT táblákat kell feldolgoznunk, és ki kell íratnunk a kapott tömböket. Még létre kell hoznunk egy háromdimenziós tömböt, amibe elmenthetjük az összetartozó PES PID-jeit. Lássuk, hogyan tesszük ezt meg.



25. ábra PMT feldolgozása

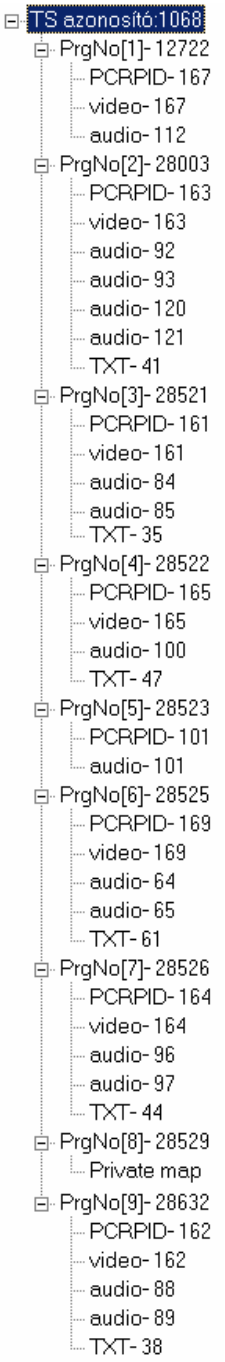
Ezt az eljárást is le kell futtatnunk az egész beolvasott mintára, ahol először megkeressük a szinkronbájtot, majd beolvasva az adott csomag PID-jét megnézzük, hogy ez egy PMT-e? Ha nem, akkor irány a következő sor, ellenben megnézzük, hogy van-e feldolgozó mező. Ha van, akkor átugorjuk úgy, hogy az aktuális pozíciónkat megnöveljük a feldolgozómező hosszával. Utána beolvassuk a következő két bájtot, melyek értéke állandó (0,2). Aztán elmentjük a PMT hosszát és a program azonosítót is beolvassuk. Ha a beolvasott program azonosító megegyezik a PID-hez rendelt program number-rel, és az érvényességjelző is aktív, akkor folytathatjuk a feldolgozást. Ami abból áll, hogy elmentjük az ajánlott PCR PID-jét, és beolvassuk a segédinformáció hosszát. Megnézzük, hogy ez eléri-e a 188-as csomag végét. Ha nem, akkor beolvassuk a következő információkat, ellenben meghívjuk a PMT_folyt nevű eljárásunkat, ami megkeresi a következő PMT táblát, ahol rááll a következő információ kezdetére. Ezzel elérkeztünk a következő információ kezdetére, levonjuk a PMT hosszából a 9-et (beolvasott adatok), 4-et (CRC), és az info hosszát. A következő információ a típust definiálja, tehát meg kell hívnunk a típusvizsgálatot, mely a megfelelő számra állítja a k értékét. Ezután beolvassuk a PID címet és elmentjük az adott tömb megfelelő helyére. Már csak annyi a dolgunk, hogy beolvassuk a jelhez tartozó info hosszát, megnézzük, hogy meddig tart, sorvéget ellenőrzünk, ha kell meghívjuk a PMT_folyt eljárásunkat, a végén a hosszt is megfelelően csökkentjük. Ezután megnézzük, hogy a hossz nulla-e, mert ha nem, akkor beolvassuk a következő jel típusát, PID-jét, infohosszát…

Miután a hossz nulla lesz, kényelmesen hátradőlhetünk, hiszen már csak ki kell íratnunk a képernyőre. Persze ez nem olyan egyszerű, a szintaktika kibogarászása nem egy egyszerű dolog. Én először a megfelelő formában kiírtam egy fájlba, majd ezt betöltöttem a kívánt számformátumban (HEXA vagy DEC). Miután megvalósítottam azt, hogy bárhonnan be lehet tölteni a TS-ünket, ez a módszer nem lett jó, mert mindig az aktuális könyvtárba létrehozott egy újabb fájlt. Mivel ez megengedhetetlen, így hosszas keresgélés után rájöttem a legmegfelelőbb megoldásra. A kiíratási eljárást így átalakítottam a Delphi7 által kezelhetőbb formára, mellyel már hibamentesen működik a kiíratási struktúra.

A program megírásánál természetesen előkerültek különböző problémák, melyeket szeretnék itt bemutatni okulás gyanánt. Az első az volt, hogy csak két hangra terveztem, így

több hang esetén csak az első és az utolsó hangot mentette el az eljárásom. Mostani felépítés mellett 1 video, 5 hang és 1 TXT címének a kinyerésére alkalmas a program.

A második probléma az volt, hogy a program nem tudott a megfelelő helyre állni, mikor a további információ másik csomagban volt. Itt először azzal volt a probléma, hogy nem vettem észre, hogy a csomag csak 188 bájtig hasznos, utána csak hibajavító kód van. Miután ezt kijavítottam, már két csomagot tökéletesen tudott feldolgozni, de tovább nem tudott lépni. Így be kellett vezetnem egy hely nevű változót, melybe elmentettem az éppen aktuális i értéket. Erre azért volt szükség, mert a PMT_folyt eljárás meghívásához az aktuális csomagszámra is szükség van. S mivel én ezt mindig i-vel hívtam meg, így mindig csak a második PMT csomagra ugrott. Viszont a hely változóba mindig az aktuális csomag számát tárolom, így már mindig a rákövetkező csomagra ugrik. A hibák ily módon kijavításra kerültek, már csak le kell futtatnunk a programunkat és le kell ellenőriznünk a működését. Lássuk most ezeknek az eredményét.



5.5 A program által nyújtott végeredmény

26. ábra A programfa

Nos így nézne ki egy TS programfája. A fájl, ami tartalmazza a feldolgozott TS-t, megtalálható a lemezmellékleten 20041018ts.dat és 20041018se.dat neveken megtalálható. Az első tartalmazza a TS jelfolyamot, míg a másik a paralel-Ethernet átalakító által elküldött segédadatokat. Hogy milyen adatok ezek, és hogy mire használhatjuk őket, arról a későbbiekben még beszélni kell.

A jelet, mint arról a fájlnév is árulkodik, 2004. október 18-án vettem. Pontosabban reggel 7.30-kor az Astra 1 műholdcsoport (19,2˚K-i pozíció) egyik műholdjától.

A fa megmondja a TS-ünk azonosítószámát, majd a PMT tábláink azonosítószámához hozzárendeli az összetartozó PES csomagok PID-jét. Megtehettem volna azt is, hogy a PMT táblák PID-

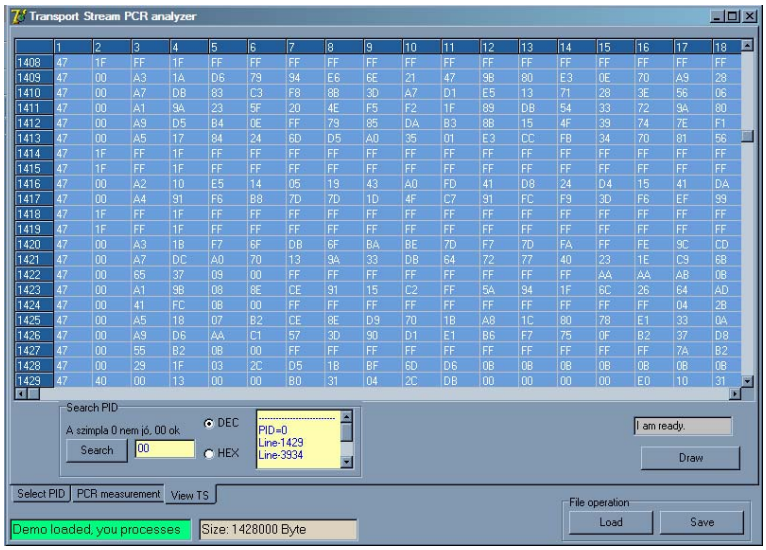
jét íratom ki, de mivel engem a PCR mérésénél nem érdekel a PMT tábla PID-je, és a programokra legtöbbször az azonosítószámukkal hivatkozunk, így inkább ezt írattam ki. Egy gomb segítségével viszont megjeleníthetjük a PMT táblák PID értékeit. A programom működéséről később részletesen fogok még írni.

Először nem akartam létrehozni a csomagok táblázatos megjelenítését, hiszen ezt a CableWorld Kft fejlesztői megcsinálták, mégpedig nagyon szuperül. A megoldásunk ellenőrzésére és a TS felépítésének jobb megismerésére szükségünk van egy ilyen megjelenítőre. Ezen okból mégiscsak bevettem ezt a feladataim közé. Habár nem lett annyira tökéletes, mint amilyen a CableWorld Kft által készítetté, de a mi céljainknak megfelelő. Használata igen egyszerű, hiszen a beolvasott információt a „Draw" nevű nyomógombbal egyszerűen kirajzolhatjuk. Kiegészítettem egy PID szelektorral is, amely a megadott PID-ű csomagok sorszámát megadja a lelkes felhasználónak. Ez a kis program volt segítségemre az ellenőrzés alatt. Térjünk is rá az ellenőrzés főbb momentumaira.

5.7 Az ellenőrzés főbb lépései

5.7.1 Az előkészületek

Első lépésként keressük meg, hogy mely sorokban találhatunk PAT csomagokat. Ehhez a programunk view TS oldalára kell lépnünk, be kell olvasnunk a jelfolyamot és meg kell keresnünk a 00-ás PID-ű sorokat.



27. ábra A TS táblázatos megjelenítése

Mint az látszik a 27. ábrán is, a vett minta 2 darab PAT-ot tartalmaz, mégpedig az 1429. és a 3934. sorban. Vizsgáljuk meg most az 1429. sort. Nézzük meg bitről bitre a tartalmát.

5.7.2 A PAT csomag feldolgozása

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1429	47	40	00	13	00	00	B0	31	04	2C	DB	00	00	00	00	E0	10	31	B2	F1	B2	6D	63	ED	63	6F	69	EF

	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
1429	69	6F	6A	EF	6A	6F	6B	EF	6B	6F	6D	EF	6D	6F	6E	EF	6E	6F	71	EF	6F	6F	D8	EF	D8	0F	C8	A5

	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
1429	0B	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
1429	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
1429	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
1429	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196
1429	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	BE	EB	00	98	1B	C2	60	F9

	197	198	199	200	201	202	203	204
1429	05	5A	35	D3	F8	DE	DC	C7

28. ábra Az 1429. csomag

Kiszedve az 1429-es sort, láthatjuk egybe a PAT csomagot. Látszik, hogy 47_h-val kezdődik, azaz a szinkronbájttal. A második és harmadik bájt együttes értéke 4000_h=0100 0000 0000 0000_b. Az egyesről még nem beszéltem, ha az első két bit 01, akkor PES vagy

PSI kezdődik. Azaz ez nem egy félbehagyott csomag folytatása, hanem az információ itt kezdődik. Mivel az utolsó tizenhárom bit 0, a csomag PID-je 0, így ez a csomag egy PAT.

A 4. bájt értéke 13_h=0001 0011_b, azaz az első két bit 00, így nincs lekódolva a csomagunk, a következő két bit 01, tehát feldolgozó mező sincs, és a számláló értéke 3-on áll.

5. bájt 0, tehát a kezdetnél járunk, a 6. bájt is nulla, így tényleg PAT-ról van szó. A 7. és 8. bájt árulja el nekünk a csomagunk hosszát. Mivel ennek értéke $B031_h$=1011 0000 0011 0001_b, és ebből csak az utolsó 12 bit tartozik a hossz megadására, így csomagunk hossza 49 bájt. Azaz 49bájt + 4bájt(fejrész) + 4bájt(CRC) az 57bájt. Tehát a csomag 58. bájtjától kezdve FF_h kitöltő bitekkel kell rendelkeznie. Ha megnézzük, ez teljesül is 188-as számú bájtig. Utána viszont nem az… Ennek egyszerű oka van, hiszen innen kezdődik a Reed-Solomon kód. Tehát eddig minden úgy alakul, ahogy elvárjuk.

A 9-10. bájt árulkodik a TS azonosítószámáról. Ennek értéke $042C_h=1038_d$. Térjünk csak egy pillanatra vissza a 26-os ábránkra, és hasonlítsuk össze a kapott eredményünkkel. Szemünk felragyoghat, lelkünk megnyugodhat, egyezik a programunk által mutatott értékkel. Már csak a 11. bájtot kell megnéznünk, hogy érvényes-e a csomagban található információ. Az érték DB_h=1101 1011_b, és mivel az utolsó bit egyes, így csomagunk hasznos információkat tartalmaz. A 14-15. bájt értéke 0, és a 16-17. bájt értékéből ($E010_h$=1110 0000 0001 0000_b) a NIT értéke 16_d.

Most már csak a program azonosítókat és a PID értéküket kell kinyernünk.
Lássuk ezeket:

18-19. bájt $31B2_h=12722_d$, ez az első program azonosítója,
20-21. bájt $F1B2_h$=111*1 0001 1011 0010*$_b$, ebből a PID értéke: $4530_d=11B2_h$,

22-23. bájt $6D63_h=28003_d$, ez a második program azonosítója,
24-25. bájt $ED63_h$=111*0 1101 0110 0011*$_b$, ebből a PID értéke: $3427_d=D63_h$,

26-27. bájt $6F69_h=28521_d$, ez a harmadik program azonosítója,
28-29. bájt $EF69_h$=111*0 1111 0110 1001*$_b$, ebből a PID értéke: $3945_d=F69_h$,

30-31. bájt $6F6A_h=28522_d$, ez a negyedik program azonosítója,

32-33. bájt $EF6A_h$=111*0 1111 0110 1011*$_b$, ebből a PID értéke: 3946_d=$F6A_h$,

34-35. bájt $6F6B_h$=28523_d, ez az ötödik program azonosítója,
36-37. bájt $EF6B_h$=111*0 1111 0110 1010*$_b$, ebből a PID értéke: 3947_d=$F6A_h$,

38-39. bájt $6F6D_h$=28525_d, ez a hatodik program azonosítója,
40-41. bájt $EF6D_h$=111*0 1111 0110 1101*$_b$, ebből a PID értéke: 3949_d=$F6D_h$,

42-43. bájt $6F6E_h$=28526_d, ez a hetedik program azonosítója,
44-45. bájt $EF6E_h$=111*0 1111 0110 1110*$_b$, ebből a PID értéke: 3950_d=$F6E_d$,

46-47. bájt $6F71_h$=28529_d, ez a nyolcadik program azonosítója,
48-49. bájt $EF6F_h$=111*0 1111 0110 1110*$_b$, ebből a PID értéke: 3951_d=$F6F_h$,

50-51. bájt $6FD8_h$=28632_d, ez a kilencedik program azonosítója,
52-53. bájt $EFD8_h$=111*0 1111 1101 1000*$_b$, ebből a PID értéke: 4056_d=$FD8_h$.

Ezzel a végére is értünk a PAT csomagunknak, és az előre elvárt értékeket is kaptuk, így a PAT feldolgozását végző programrészletünk nem lehet annyira rossz. Menjünk tovább, és dolgozzunk fel néhány PMT táblát is. Hiszen ezek működése bonyolultabb, mivel át is léphetünk egy másik csomagra is.

5.7.3 A 12722_d című PMT feldolgozása

Használjuk fel programunk segítségét, hogy megtudjuk, mely sorokban keressük a PMT táblánkat. Eredményül azt kapjuk, hogy a 447., a 2958. és az 5480. sorban leledzik a vizsgálni kívánt PMT-nket. Vizsgáljuk meg ennek tudatában a 447-es sort.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
447	47	51	B2	13	00	02	B0	A7	31	B2	F1	00	00	E0	A7	F0	00	02	E0	A7	F0	13	09	11	01	00	E6	23

	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
447	00	64	FF	00	08	00	00	00	00	80	02	FF	FF	03	E0	70	F0	19	09	11	01	00	E6	23	00	64	FF	00

	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
447	08	00	00	00	00	80	02	FF	FF	0A	04	73	70	61	01	C0	E1	2D	F0	41	C2	38	43	53	45	5F	50	49

	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
447	4C	31	43	53	45	5F	43	48	41	4E	43	53	45	5F	50	49	4C	32	43	53	45	5F	53	45	53	00	43	53

	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
447	45	5F	45	50	47	00	43	53	45	5F	46	55	54	31	43	53	45	5F	46	55	54	32	C6	05	01	00	05	05

	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
447	FF	C1	E1	2C	F0	0A	C2	08	50	49	4C	4F	54	45	00	00	C0	E0	DE	F0	0A	C2	08	43	48	41	49	4E

	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196
447	45	00	00	E9	A9	B9	6E	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	07	AF	42	DE	6D	25	5C	83

	197	198	199	200	201	202	203	204
447	75	CA	52	D0	EC	DB	A1	65

29. ábra A 447. sorban lévő TS csomagunk

A szinkronbájt után a 2. és 3. bájt értéke $51B2_h$=100*1 0001 1011 0010*$_b$, melyből a csomagunk PID-je $11B2_h$, a megfelelő csomagra mutatott a PID keresőnk. A 4. bájt értéke $1B_h$=0001 1011$_b$, azaz nincs kódolva ez a program, és feldolgozó mező sincs ebben a csomagban. Az 5. bájt értéke 0, tehát ez a kezdeti csomag, míg a 6. bájt értéke 02, azaz PMT csomagot találtunk. A 7. és 8. bájt értéke $B0A7_h$=1011 *0000 1010 0111*$_b$, ebből a PMT hossza 167 bájt, hozzáadva 8-at, megkapjuk a csomag végét, ami a 175. bájt. Ha megnézzük az ábrát, akkor látjuk, hogy utána már csak kitöltő bájtok vannak, ez a PMT tábla csak egy csomagban található. A 9. és 10. bájt értéke $31B2_h$=12722_d, tehát tényleg a PID-hez rendelt azonosítóval rendelkezik a csomagunk. A 11. bájt értéke $F1_h$=1111 000*1*$_b$, a csomagunk érvényes. A 12-13. bájt értéke 0, így a következő kettő bájt árulkodik a program PCR-jét tartalmazó csomag PID-jéről. Mivel ennek értéke $E0A7_h$=111*0 0000 1010 0111*$_b$, a PCR PID 167_d, $A7_h$. Mivel a következő két bájtból a programinfo értéke 0 ($F000_h$=1111 *0000 0000 0000*$_b$). Azaz a következő, számunkra hasznos, bájt a 18., melynek kettes értéke azt mutatja, hogy az itt kinyerhető PID egy video PES csomagot azonosít. Értékét megkapjuk, ha a 19-20. bájtot elemezzük ($E0A7_h$=111*0 0000 1010 0111*$_b$), akkor látjuk, hogy szintén 167_d a video PID értéke. Dolgozzuk fel a 21-22. bájtokat, melyek értéke $F013_h$=1111 *0000 0001 0011*$_b$, a segédinformációk 19 bájton vannak ábrázolva. Így a következő PES-ről a 22+19+1=42. bájt árulkodik. Ennek értéke 03, tehát audio típusú az innen kinyert PID-ű PES csomagok. A PID-je viszont 70_h, 112_d. A program információjának hossza $F019_h$=1111 *0000 0001 1001*$_b$, 25 bájt hosszan találhatunk segédinformációkat. Így a következő PES-ről az információkat a 46+25+1 művelet

elvégzése után kapott 72. bájton kapjuk. Mivel ennek a bájtnak az értéke $C0_h$, ami nem egy előre meghatározott tulajdonsággal rendelkező érték, hanem szabadon felhasználható, és általában szerviz információk jelzésére használják. Az itt kinyert PID értéke 301_d, melyre rákeresve a PID keresőnkkel, azt tapasztalhatjuk, hogy megtalálható az elmentett csomagunkban, mégpedig kb. 5500 db közül 9db ezzel a PID-del rendelkezik. Segédinformációk hossza 65 bájt, a következő információt 76+65+1=142. bájton kapjuk. A típusra C1-et kapunk, melyre ugyanazok az elmélkedések érvényesek, mint az előzőre. PID-je 300_d, melyből 52 is található az elmentett jelfolyamban. A következő információt 142+10+1, azaz 153. bájton található. A típusra azt kapjuk, hogy C0, kicsit már ismerős, PID-je 222_d, és a segédinformációk 10 bájtot foglalnak el. Így a következő hasznos információt a 172. bájton találhatjuk. Ez pedig már csak a CRC. Így ezt a csomagot teljesen feldolgoztuk, és a kapott eredmények is egyeznek az elvárttal. Mint láthatjuk, igen sok munkától kímél meg minket a programunk. Viszont még nem dolgoztunk fel olyat, ami nem ér véget egy csomagon belül.

5.7.4 A 28521_d azonosítóval rendelkező csomag feldolgozása

Ennek az azonosítónak a hozzárendelt PID-je az $F69_h$, nézzük azt meg, hogy hol találhatunk ilyeneket. A kapott eredmény a következő: 166, 2256, 2394, 2535, 2674, 4772, 4912, 5055 és az 5194. Az egymáshoz közel lévő értékek arra engednek következtetni, hogy 4 csomag együttesen írja le az adott programot. Nézzük is meg rögtön a 2256. csomagot.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
2256	47	4F	69	10	00	02	B2	D8	6F	69	F7	00	00	E0	A1	F0	00	02	E0	A1	F0	8A	09	3E	01	00	E6	35

	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
2256	00	80	FF	00	01	00	00	00	00	00	02	00	21	E6	44	00	84	FF	00	00	00	10	00	00	00	00	00	21

	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
2256	E6	40	00	86	FF	00	00	00	80	00	00	00	00	00	21	E6	48	00	85	FF	00	00	00	00	00	00	00	22

	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
2256	00	21	09	04	0D	03	E6	37	09	0F	05	00	E6	36	10	01	00	13	01	20	14	03	00	78	00	09	0F	05

	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
2256	00	E6	33	10	01	00	13	01	20	14	03	00	80	00	09	0F	05	00	E6	4B	10	01	00	13	01	20	14	03

	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
2256	00	81	00	09	0F	05	00	E6	46	10	01	00	13	01	20	14	03	00	83	00	03	E0	54	F0	90	09	3E	01

	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196
2256	00	E6	35	00	80	FF	00	01	00	00	00	00	00	02	00	21	E6	44	00	84	47	38	0D	70	71	2C	6B	95

	197	198	199	200	201	202	203	204
2256	98	77	B1	21	8D	52	5C	C3

30. ábra A 2256. csomag

Ezt a PMT-t már nem fogom olyan részletesen körülírni, megmagyarázni mindent, hiszen az előbbi lépéseket kell végrehajtani a megfelelő sorrendben. Lássuk ennek függvényében az adatokat.

2-3. bájt $4F69_h$, a 4-es miatt PES vagy PSI kezdődik és a PID cím $F69_h$,

4. bájt 10_h, nincs lekódolva, és feldolgozómező sincs,

5-6. bájt 0 és 2, ez a kezdőcsomag, és PMT is,

7-8. bájt $B2D8_h$, 728db bájt hosszú a PMT, azaz 4 csomag tartalmazza,

9-10. bájt $6F69_h$, egyezik a program azonosító,

11. bájt $1111\ 011I_b$, érvényes a csomag,

12-13. bájt 0,0, így a következő a PCR PID lesz,

14-15. bájt $E0A1_h$, $A1_h=161_d$ a PCR PID-je,

16-17. bájt 0, azaz nincs kiterjesztett info,

18. bájt 02, video PID következik,

19-20. bájt $A1_h=161_d$ a videojel PID-je,

21-22. bájt $F08A_h$, 138 a prg. Info hossza, a következő hasznos info 22+138+1=161,

161. bájt 03, audio PES PID-je következik,

162-3. bájt $54_h=84_d$ az audio PID-je,

164-5. bájt $F090_h$, azaz 144, így a következő hasznos info 165+144+1=310, ez már csak a következő csomagban található meg, mégpedig a 310-188+4(fejrész)=126. bájtja. Most érkeztünk a fordulóponthoz, hiszen a következő PMT PID-ű csomag 126. bájtjára kell ugranunk a következő PES csomag információjáért.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
2394	47	0F	69	11	FF	00	00	00	10	00	00	00	00	00	21	E6	40	00	86	FF	00	00	00	80	00	00	00	00

	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
2394	00	21	E6	48	00	85	FF	00	00	00	00	00	00	00	22	00	21	09	04	0D	03	E6	37	09	0F	05	00	E6

	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
2394	36	10	01	00	13	01	20	14	03	00	78	00	0A	04	66	72	65	01	09	0F	05	00	E6	33	10	01	00	13

	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
2394	01	20	14	03	00	80	00	09	0F	05	00	E6	4B	10	01	00	13	01	20	14	03	00	81	00	09	0F	05	00

	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	13
2394	E6	46	10	01	00	13	01	20	14	03	00	83	00	03	E0	55	F0	90	09	3E	01

31. ábra A2394. csomag egy részlete

Mint azt az ábra is jól mutatja, az értéke 03, tehát ismét egy audio-t tartalmazó PES-ről nyerhetünk további információkat. Ne szaladjunk annyira előre, tekintsük meg a 2. bájtot, ami nem 4_h-val kezdődik, ez a csomag egy előző folytatása. Most lássuk a további információkat:

127-8. bájt $E055_h$, ebből a PID értéke: $055_h=85_d$,

129-30. bájt $F090_h$, így a következő hasznos info pozíciója 130+144+1=275. Tehát megint meg kell keresnünk a következő $F69_h$ címmel rendelkező csomagot, és annak is a 275-188+4=91. bájtját.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
2535	47	0F	69	12	00	00	00	22	00	21	09	04	0D	03	E6	37	09	0F	05	00	E6	36	10	01	00	13	01	20

	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
2535	14	03	00	78	00	0A	04	65	6E	67	01	09	0F	05	00	E6	33	10	01	00	13	01	20	14	03	00	80	00

	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
2535	09	0F	05	00	E6	4B	10	01	00	13	01	20	14	03	00	81	00	09	0F	05	00	E6	46	10	01	00	13	01

	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	1
2535	20	14	03	00	83	00	06	E0	23	F0	1B	56	19	65	6E	67	09	00	66	72	65	10	59	66	72	6

32. ábra A 2535-ös csomag egy részlete

Mint láthatjuk, egy új típust találhatunk, mégpedig a 06-ost, ami egy TXT jelről ad tanúbizonyságot. Ennek PID-je $23_h=35_d$, és tovább boncolgatva PMT-nket, C0 és C1 típusú adat PID-jét találhatjuk meg. Mivel ennek értéke számunkra nem fontosak jelenleg, így nem is elemzem tovább a PMT-nket.

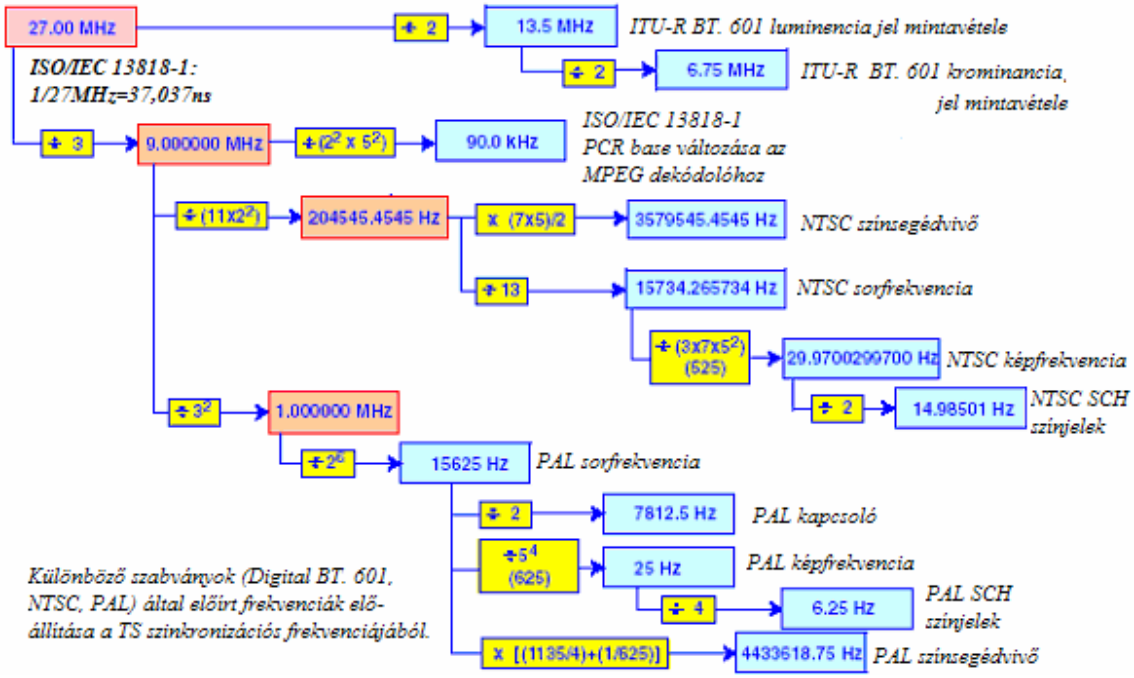
Ezzel befejezettnek tekintem a programom ezen részének tesztelését. Persze én végigboncolgattam nem is egyszer ezt és más TS jelfolyamokat, hiszen ez segített a programfelépítés megismerésében, és engem is jobban megnyugtatott a kapott eredmény. Mivel rendelkezésre áll a program, és a vett minta elmentett változata, így bárki ellenőrizheti programom helyes, vagy helytelen működését. Ha valami hiba lenne, akkor az észrevételeket szívesen venném, ha el tud jutni hozzám.

6 A PCR

6.1 A PCR lényege a dekódolás során

Sokan tesszük fel a kérdést, hogy mi is az a PCR? Miért van rá szükség? Mi a lényegi feladata? Ennek hiánya, vagy hibás értéke mit eredményez? Az első két kérdésről már beszéltem az „adaptation field” tárgyalásánál. Most elevenítsük fel ezt!

A vétel helyességéhez elengedhetetlenül szükséges az adó és a vevő oszcillátorainak azonos frekvencián történő rezgése. Ha gyorsabban rezeg a vevő oszcillátora, amitől függ a vett jel feldolgozása is, akkor egyszercsak elfogy a jel a bemenetről. Ha viszont lassabban rezeg, akkor pedig sorbaállás lenne a bemenetnél, hiszen lassabban dolgozná fel a vevőberendezésünk a jeleket. Viszont a helyzet nem ily drasztikus, mivel ennek kiküszöbölésére találták ki a vevőkben a tároló buffert. Ez egy ideiglenes átmeneti tár, ami kezeli a vétel „rendezetlenségét”. Ha mindig ugyanakkora frekvencián rezegnének, akkor nem lenne szükség sem a bufferre, sem pedig a PCR jelenlétére. Azonban mindkettőre szükség van, nézzük meg tehát, hogy milyen okra vezethető vissza.

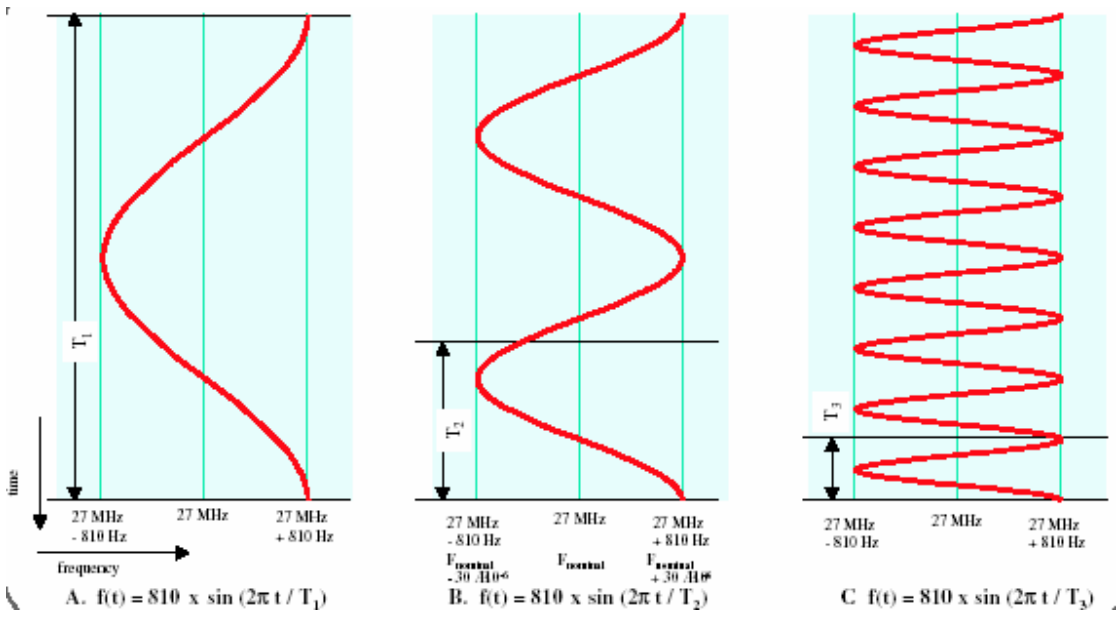


33. ábra A 27MHz-ből származtatható frekvenciák

Előtte elemezzük a 33. ábrát [11]. A set-top-box nem tartalmaz különböző, nagy pontosságú órajeleket. Elő kell állítani neki ezeket. Az előállítást a 27MHz-es órajel különböző leosztásával végzi el. Látszik, hogy miképp hozza létre a PAL, NTSC, és a digitális jel dekódolásához szükséges frekvenciákat. Az egész PCR ingadozásával járó gondunkat, az analóg jelek esetére, elfeledhetnénk, ha a set-top-box-unk tartalmazna szabványos PAL, NTSC dekódereket.

6.1.1 Az adó oszcillátorára vonatkozó megkötések

Az adó oszcillátora igen nagy pontosságú, és stabilitású, amire szigorú előírások vannak. Az egyik az az, hogy a 27MHz-es oszcillátor frekvenciaeltérése mindkét irányban 810Hz lehet. Ez 30 ppm. [1]



34. ábra Az adóoszcillátor frekvenciaingadozása [11]

Az ábrán jól látszik, hogy nem elég megadni a maximális frekvenciaeltérést, hiszen ennek gyors változási sebessége a vételhez használt set-top-box rászinkronizálását meghiúsíthatja. Ezért az oszcillátor változási sebességére, a „drift rate” értékére is

megkötést kell tenni. 75mHz/s-ban állapították meg a 27MHz-hez képest. Ebből következik, hogy a változási sebességnek 75mHz/s/27MHz=2,7778*10-9 1/s, 10ppm/h. [4]

Az adó oszcillátora így 27MHz-810Hz-es értékről 27MHz+810Hz-es értékre 6 óra alatt érne el, ha a változási sebessége éppen 10ppm. A változási sebesség is ingadozhat, ennek értéke sem állandó. Mivel a változási sebesség a frekvenciaeltérés első deriváltjának felel meg, a csúcsértéke (810*2π)/T1=0,075Hz/s. Ebből az összefüggésből kiszámolhatjuk a változás periódusát, amire 18,85 órát kapunk. Ennyi idő kell az adó oszcillátorának, hogy a változási sebesség mellett ugyanarra a frekvenciára visszatérjen. Láthatjuk, hogy igen erős megkötések ezek, melyeket a vevőeszközöktől nem lehet elvárni. Ha viszont el is várnánk, akkor is ott lenne az a gond, amikor mindkettő ugyanolyan pontosan rezeg, csak éppen más frekvenciával. Tehát a szinkronizálásra mindenképpen szükség van. Így viszont felesleges a szigorú megkötés a vevőre. Hiszen ha változási sebessége nagyobb, akkor gyorsabban tud ráállni az adó frekvenciájára. Már csak azt kell megtudnunk, hogy mi módon kerül bele a szinkronozójel a TS-ünkbe, és milyen gyakran van jelen. Ennek tudatában először erre keressük meg a választ, és csak utána jöjjön a hibaforrás keresése és a mérési módszer kiválasztása.

6.2 A PCR beültetési eljárása

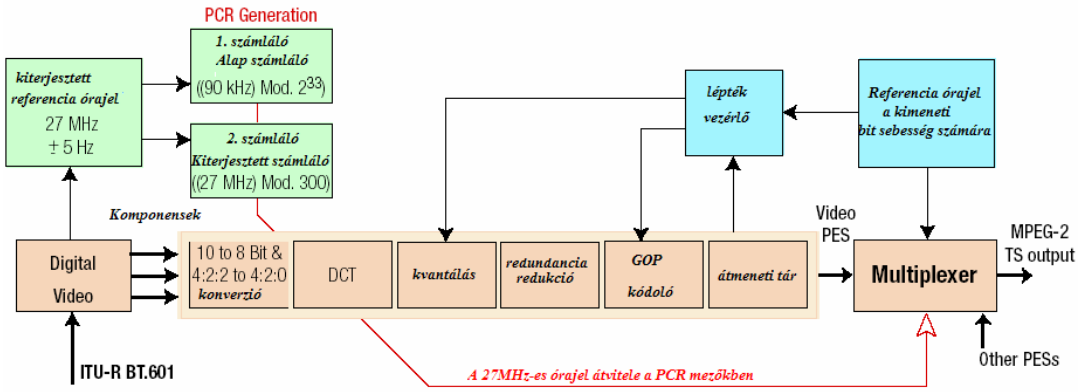
Eme kérdésre a választ megtalálni nem olyan könnyű, hiszen a gyártók nem árulják el saját módszerüket. Így azt sem tudhatjuk, hogy a beültetési eljárás egy referenciajellel történik TS-enként, vagy pedig minden programhoz külön tartozik egy oszcillátor.

Mielőtt erre keresnénk a választ, nézzük meg a dekódolást elvét. Ha nézünk egy műsort, akkor a vevőkészülékünk annak a programnak a PCR értékeit figyeli és arra szinkronizál rá. Ennek az az előzménye, hogy az egy TS-ben található műsorokat nem egy referencia-órajelre készítik, nem is készíthetik, ennek következtében arra kell gondolnunk, hogy a PCR beültetéshez programonként kell egy-egy oszcillátor. Jobban belegondolva lehet, hogy a Colombo-t 26 999 190Hz-es oszcillátor frekvenciával kódoljuk, míg az ugyanabban a pillanatban futó BL meccset 27 000 810Hz-en kódolják. Egy háztartásban mindkét jelet tudni kell egyidejűleg venni, mert lehet, hogy az anyu a Colombóra, apu meg a meccsre vágyik. A dekódolás során a vevőnket hol a Colombo-ra, hol pedig a meccsre

kell hangolnunk. Ezek a műsorok viszont lehetnek egy TS-ben. Ebből következtethetünk arra, hogy a PCR beültetéshez külön kell egy-egy oszcillátor. Ezen megállapításunkat viszont alá kell támasztanunk. Egy egyszerű kísérlettel meggyőződhetünk erről. Csak ki kell szednünk a TS-ből az összes PCR-t, és meg kell néznünk az értékeket. Ha a számok egymás után jönnek, akkor egy oszcillátor tartozik egy TS-hez, ellenben az előző feltevésünk a helyes. Erre egy egyszerű programot írtam, amely segítségével megkerestem azokat a csomagokat, amelyekben feldolgozómező volt található, azon belül is PCR értékek. Ezeket megjelenítve azt tapasztaltam, hogy a számlálók állása különböző értékben álltak programonként, mely az első feltevésemet igazolta.

Ennek tudatában már megfogalmazhatjuk a PCR beültetés eljárását. A tektronix műszergyártó cég egyik ismertetőjében [11] találtam erre egy megfelelő ábrát is, mely itt is látható.

Először nézzük meg a digitális jelhez tartozó PCR beültetési mechanizmust.



35. ábra PCR beültetése digitális videojelbe

Látszik, hogy a programokhoz külön oszcillátor tartozik, plusz a mátrixolást vezérléséhez is használnak még egy oszcillátort. Ez az oszcillátor nem 27MHz-es, hanem a kimeneti jelsebességhez igazodik. A programokhoz tartozó oszcillátorok vezérelnek egy-egy számlálót, amely értékeit adott időközönként beültetik az adott programok feldolgozómezőjébe. Az adott időközre a DVB szabvány 40ms-ot, az MPEG pedig 100ms-ot ír elő [1]. Ezek tudatában rendszerünket úgy kell elkészítenünk, hogy 40ms-onként várja a PCR-eket, de 100ms-ig tolerálja hiányukat. A gyakorlat azt mutatja, hogy a videojelekhez

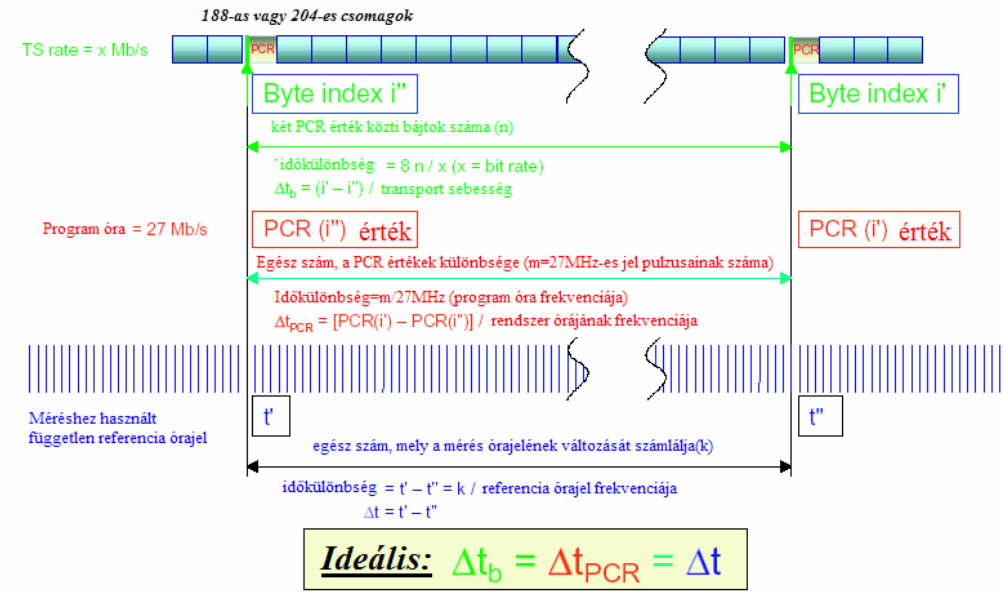
38-39ms-onként rendelnek PCR értékeket, míg rádiójelekhez, azaz csak audio műsorokhoz, 100ms körüli időköz a bevett megoldás.

Az mpeg kódolást vezérlő 27MHz-es órajelből generáljuk a PCR alap, és kiterjesztett értékeit. Miután a digitális jelből előállítottuk a video PES csomagunkat, beillesztjük a TS-ünkbe. Kb. 40ms-os időközönként az éppen aktuális PCR értéket beépítjük a video PES-ünk adaptation field mezőjébe.

Az analóg PAL jel átvitele, és PCR beültetési eljárása kicsit másképp működik. A PAL jel átvitele úgy történik, hogy a PAL jelet az adóoldalon dekódolják, előállítják az RGB színjeleket, majd ezeket kódoják át digitális jellé, és viszik át az átviteli csatornán. A 27MHz-es referenciajelet is a PAL jel színsegédvivőjével vezérlik, tartják a megfelelő határértékek közt. Innentől viszont a PCR beültetés mechanizmusa megegyezik az előbb leírtakkal.

A beültetett PCR értéket sok mindenre felhasználják a vételi oldalon. A két egymást követő PCR érték különbségét használják fel az órajel kinyeréséhez, a szinkronizáláshoz, számomra a mérés megvalósításához ennek értékeit kell kiszámolnom. A szinkronizáláshoz nem lenne szükség 42 biten tárolni a PCR értékeket, és szétszedésére (base és extended értékek) sem kényszerít semmi. A PCR base 33 biten kerül átvitelre, ugyanúgy, mint a PTS, DTS értékek. Ennek oka az, hogy a megjelenítés és dekódolás időbeni vezérlését a PCR base értéke vezérli. Tehát a különbség kell az órajel szinkronizálásához, az abszolútérték pedig az mpeg jel dekódolásához.

6.3 A PCR mérésének lehetőségei



36. ábra PCR mérési lehetőségek [10]

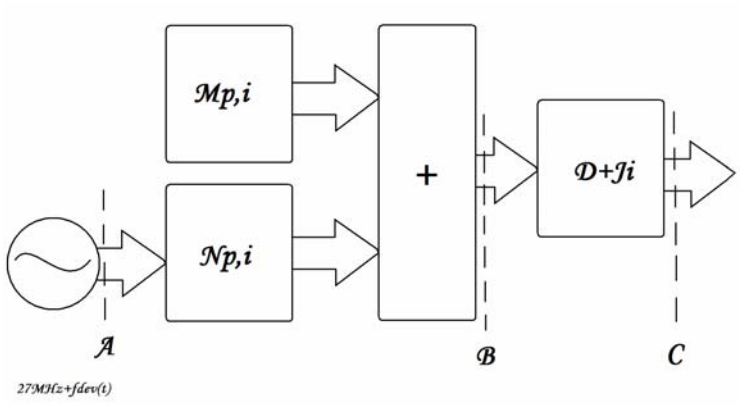
Ezen az ábrán nyomon követhetjük a PCR értékének és az órajelnek az összetartozását. Két egymást követő PCR értékek különbségének átszámítása megad egy időt. Ha elmentjük a PCR értékek bájtpozícióját és ezek különbségét megszorozzuk a TS sebességével, akkor megkapjuk a két PCR között ténylegesen eltelt időt. A harmadik módszer, hogy egy külső referencia órajelet használunk, amit elindítunk az első PCR értékénél, és leállítjuk a második érkezésénél. Ebből ugyancsak mérhetünk egy időt. Sikeres beültetésnél ezek értéke egyező. Ez lenne az ideális eset.

A PCR beültetési hibája sohasem lehet 0. Ennek egyik oka az, hogy a 27MHz-es órajel által vezérelt PCR érték 37ns-onként változik, nem pedig folyamatosan. Ebből következik, a PCR érték 1 bittel való eltérése 37ns-os fázishibát eredményez. Mivel a PCR érték, és a beültetési pozíció közti fáziseltérés még más dolgoktól is függ (pl. a másik beültetni kívánt jelünk érkezési fázisa), ezért ezt a hibát 500ns-ban állapították meg. Ez nagyon pontos beültetést kíván. El is nevezték PCR accurancy-nak.

Mikor a megszerkesztett TS-ünk az átviteli csatornára kerül, az ott fellépő hálózati jitterek, esetleg a remultiplexálás közben keletkező bájtcsúszások következtében a PCR

hiba jelentősen megnőhet. A vételi minőség biztosítása érdekében az itt keletkező újabb hibák, és a beültetési fázishiba összegére egy újabb kritériumot tettek. Ez a kritérium az ISO/IEC13818-9-es ajánlás szerint nem lépheti túl a ±25µs-ot.

A PCR hiba tehát nem egy forrásra vezethető vissza, hanem többre. Ezeket szét kell tudnunk választani egymástól. Nézzük meg a következő ábrát. [4]



37. ábra PCR hibák keletkezése

Láthatjuk, hogy a hiba forrásai hol találhatóak. Az első hiba az a 27MHz-től való eltérést jelenti. A szakmában ezt frequency offsetnek nevezték el, a rövidítése PCR_FO. Mint azt már megírtam, ez 810Hz lehet mindkét irányban. Ezzel szorosan összefügg a frekvenciaváltozás sebessége, a drift rate, PCR_DR. Ezeket az adóoldalon az "A" pontnál észlelhetjük legelsőként. A számlálóból adódó hibát $N_{p,i}$-vel jelölhetjük. Ennek értéke ±37ns. A beültetés során keletkező fázishiba, $M_{p,i}$, accurancy, PCR_AC értéke ±500ns lehet. Ezt a "B" pontnál mérhetjük legelőször, tehát az adó kimenetén. Ezután a jelünk a vevőkészülékig sok csatornán, remultiplexeren átmehet, ahol a csatorna adattovábbítási sebessége változhat, lehet jitteres. Ne feledkezzünk meg a remultiplexálásnál keletkező csomagcsúszásokról sem. Mivel a remultiplexer több TS-ből csinál egyet, a bejövő TS-ek különböző sebességben érkeznek, és az is megeshet, hogy a két remultiplexálni kívánt csomag egyszerre érkezik, de egyszerre nem kerülhetnek tovább. Az egyiknek várnia kell a sorára. Itt keletkezik egy nagyobb jitter érték. Ha mindig az egyiket késleltetjük, mégpedig ugyanannyival, akkor csak egy hibás PCR keletkezik, mivel a többi PCR érték ehhez képest

már jól lesz beültetve. Nem ez a gond a remultiplexált jeleknél, hanem a gond ott keletkezik, ha mindig más fázisban érkeznek a jelek a remultiplexáláshoz. Nagyon nagy lehet a kimeneten a PCR_OJ, hiszen itt már ezt mérhetjük könnyedén, ilyenkor szoktak olyan remultiplexert alkalmazni, amely képes a PCR-t újra beültetni. Vagy rászinkrozinálnak a remultiplexálni kívánt programokra, és akkor azzal a frekvenciával végzik a PCR újragenerálást, vagy pedig amit én is tapasztaltam, egyszerűen vesznek egy 27MHz-es oszcillátort, és azzal ültetik be az új PCR-t. Ez PCR_FO-t fog eredményezni, mivel nem a programhoz eredetileg tartozó oszcillátorfrekvenciával keletkezik az új referenciajel, hanem attól akár teljesen független órajel hatására. Ha ez túlságosan eltér, akkor lehet, hogy a vevő bufferje már nem tudja lekezelni az ebből adódó hibákat.

6.4 A PCR méréséhez tartozó elméleti tudnivalók

Az ETSI TR 101 290-es [4] ajánlása foglalkozik a digitális műsorátvitel esetén keletkező mérési problémákkal. Ez már az ETR 290-es módosított változata, ami 2001. májusában készült el. Az első ajánlás csak rendszermérésekkel foglalkozik, azaz RF mérésekkel, BER-rel, MER-rel, C/N-nel stb. Ez a változat viszont már az adatátvitel ellenőrzésére is használható méréseket szolgáltat. Ezek közül az egyik a PCR mérésére vonatkozó mérések készítéséhez támpontot adó ajánlás.

Mielőtt megnéznénk a mérési módszereket, meg kell értenünk a méréshez kiadott maszkot. Az összes, előbb említett, PCR hibát fel lehet fogni szinuszos függvények kombinációjaként az órajel frekvenciájának bevonásával. [4]
Az órajelre felírhatjuk a következő egyenletet:

$$A \cdot \sin\left[\omega_c \cdot t + \Theta(t)\right] = A \cdot \sin\left[\omega_c \cdot t + \Theta_p \cdot \sin\left(\omega_m \cdot t\right)\right] \qquad (1)$$

$\omega_c = 2\pi * f_c$

$\Theta(t)$-fázismoduláció függvénye

Θ_p-fázismoduláció csúcsértéke

ω_m -fázismoduláció körfrekvenciája

A $\Theta(t)$ érték jelzi, tartalmazza az órajel ingadozásából eredő hibát. Az órajel értékének

változását jelzi a $\Theta_i(t)$:

$$\Theta_i(t) = \omega_c \cdot t + \Theta(t) = \omega_c \cdot t + \Theta_p \cdot \sin(\omega_m \cdot t) \quad (2)$$

Ennek első deriváltja megadja a frekvencia pillanatnyi értékét leíró függvényt:

$$\omega_i(t) = \Theta_i'(t) = \omega_c + \Theta_p \cdot \omega_m \cdot \cos(\omega_m \cdot t) \quad (3)$$

Ha a kapott eredményt még egyszer deriváljuk, akkor megkaphatjuk a változás sebességét:

$$r_i(t) = \omega_i'(t) = \Theta_i''(t) = -\Theta_p \cdot \omega_m^2 \cdot \sin(\omega_m \cdot t) \quad (4)$$

Θ_p a maximálisan elszenvedhető fázishiba értékével kapcsolatban van. Ennek nagysága az előző egyenletekből, és az ISO/IEC 13818-1-es ajánlásban [1] megfogalmazott maximális fázishiba értékéből a következő:

$$\Theta_p = \omega_c \cdot T_{\max} = 2\pi \cdot 27\ MHz \cdot 500\ ns = 84{,}823\ rad \quad (5)$$

A (3)-as egyenletből. és az ISO/IEC 13818-1-ban előírt maximális frekvenciaeltérésből a Θ_p értéke:

$$\Theta_p = 2\pi \cdot 810 / \omega_m = 5089{,}4 / \omega_m \quad (6)$$

A (4)-es egyenlet, illetve az ISO/IEC 13818-1-ban előírt maximális változási sebesség (75mHz/s) figyelembevételével a Θ_p értéke:

$$\Theta_p = 2\pi \cdot 0{,}075 / \omega_m^2 = 0{,}4712 / \omega_m^2 \quad (7)$$

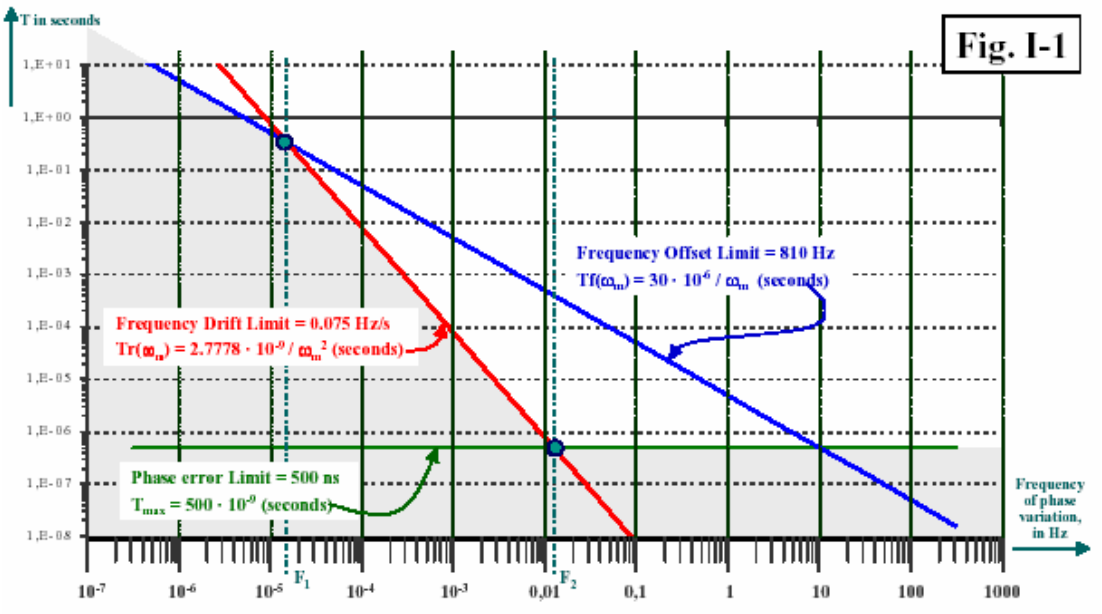
Ezen (5,6,7) egyenleteket rendezzük most a Tmax értékre, hiszen ezt ábrázolva kapjuk meg a maximálisan megengedett változás nagyságát. A kapott eredmények:

$$T_{\max} = \Theta_p / (2\pi \cdot 27\ MHz) = 500\ ns \quad (8)$$

$$T(\omega_m) = 2\pi \cdot 810 / (2\pi \cdot 27\ MHz \cdot \omega_m = 30 \cdot 10^{-6} / \omega_m \quad (9)$$

$$T(\omega_m) = 2\pi \cdot 0{,}075 / (2\pi \cdot 27\ MHz \cdot \omega_m^2 = 2{,}7778 \cdot 10^{-9} / \omega_m^2 \quad (10)$$

Ha a (8), (9), (10) alatti függvényeket ábrázoljuk fm (ez a rendszerünket megtámadó fázis moduláció frekvenciája) függvényében, akkor megkapjuk a 38. ábrát. [4]



38. ábra PCR jitter komponensei

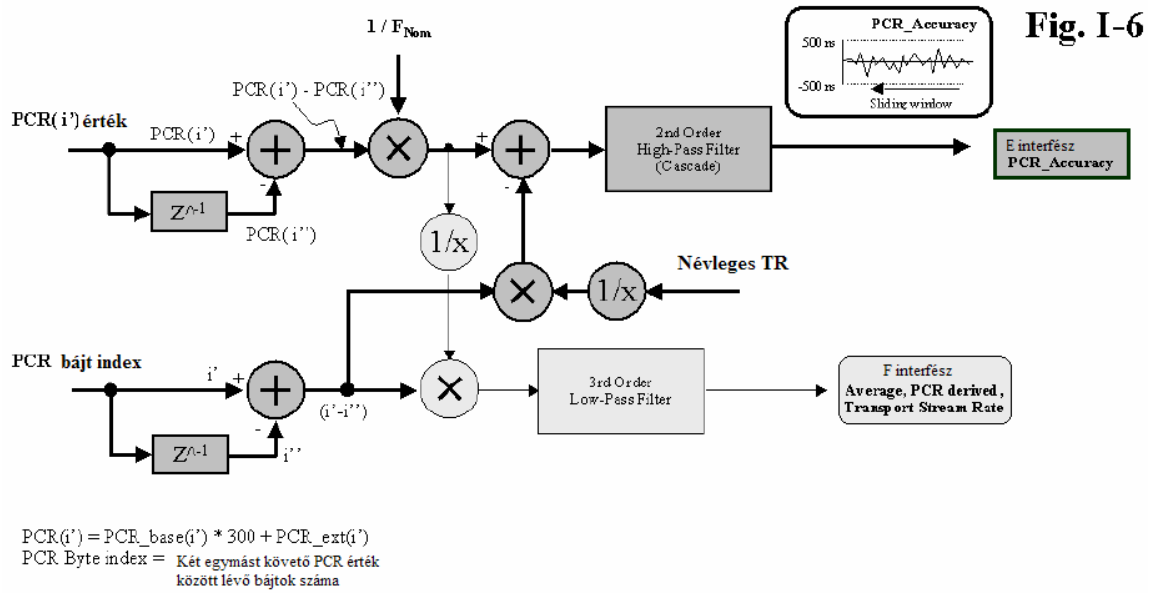
Látható, hogy a frekvencia-változás sebességét ábrázoló görbe a fázishiba görbéjét kb. 10mHz (11,86mHz) fm-nél metszi. Azaz, a PCR hiba frekvenciaösszetevőkre bontható. 10mHz alatti összetevők a PCR hiba drift rate-jét határozza meg, míg a 10mHz feletti komponensek főképp a jitter hibát tartalmazzák. Tehát, ha a kapott PCR értékeinket egy digitális szűrőn engedjük át, akkor szét tudjuk választani a frekvencia-változás sebességéből eredő hibát és a rendszerből eredő jitter hibát.

6.5 Hardveres úton történő mérési módszerek

A TR 101 290-es szabvány a PCR méréséhez hardveres konfigurációt javasol. Nézzük meg ezeket.

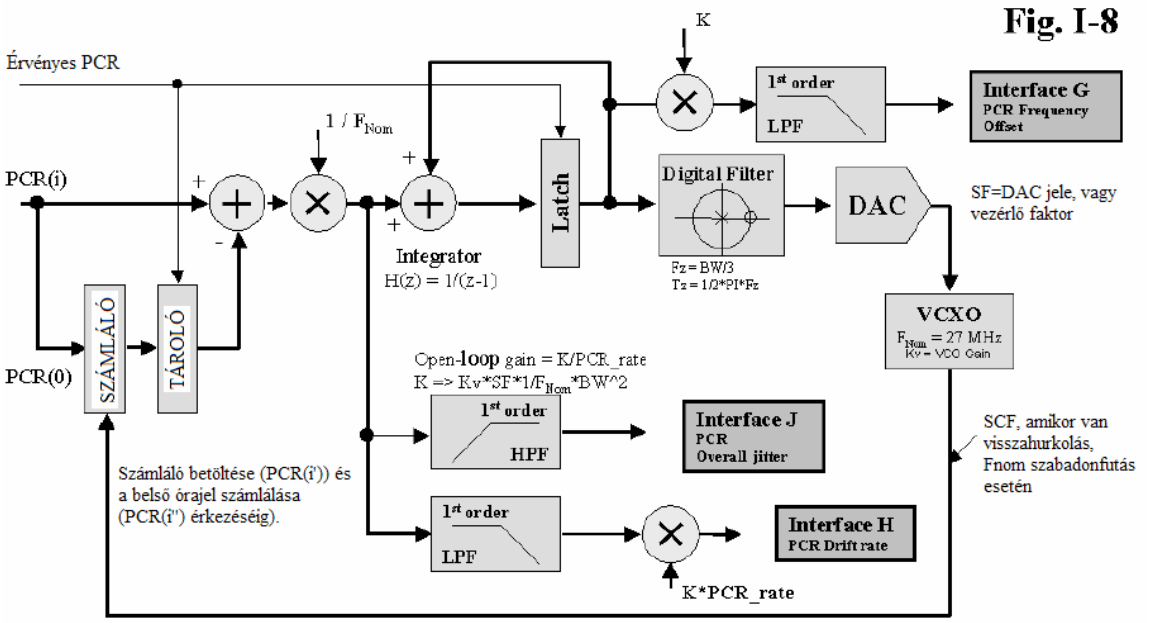
Kezdjük először a PCR_AC-vel. A két egymást követő PCR különbségét elosztjuk a 27MHz-es frekvenciával, akkor megkapjuk a 2 PCR között eltelt időt. A két PCR között

lévő bájtok számát elosztjuk a TS sebességével, ezáltal megkapjuk a ténylegesen eltelt időt a két érték között. Ezek különbségét át kell ereszteni egy szűrőn, ami a hibának időbeli átlagát veszi. Ezáltal nem a csúcsértéket mérjük, hanem valamilyen átlagos hibát.



39. ábra PCR_AC mérési módszerei [4]

A többi PCR méréshez már szükségünk van egy saját oszcillátorra, ami követi az adó frekvenciáját. Ez látszik a 40. ábrán is. [4]

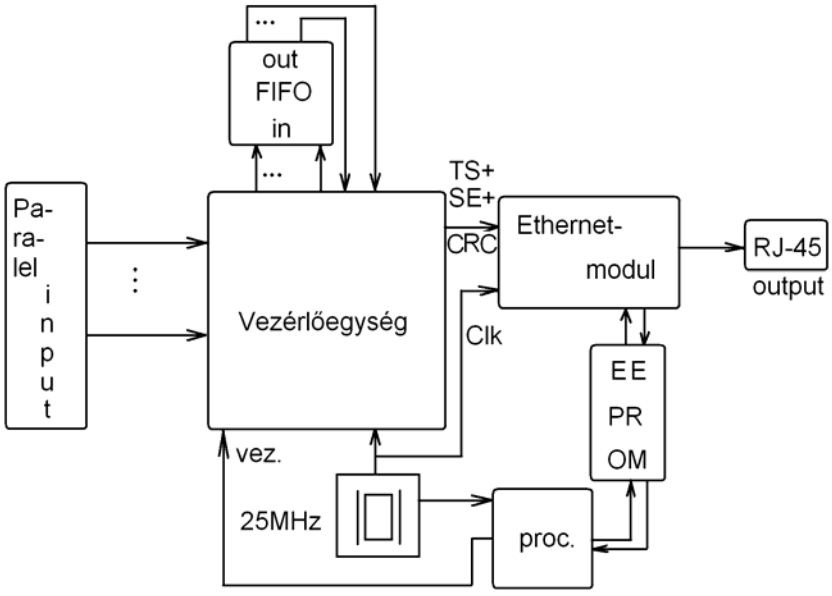


40. ábra A többi PCR hiba mérése

Az első PCR érték elindít egy számlálót, amely a belső VCO frekvenciáját számlálja a következő PCR értékig. A számláló által mutatott érték a két PCR közti időt számlálja. Ebből kivonva a két PCR érték különbségét megkapjuk a PCR hibát. Ez a hiba sokféle forrásból származhat. Ha a sorba jövő PCR értékeket 10mHz-es felüláteresztő szűrőn engedjük át, akkor megkapjuk a hálózatot, és a beültetést jellemző PCR_OJ értékét [4]. Mivel a PCR_DR lassú változás, ezért ezt aluláteresztő szűrő segítségével kaphatjuk meg [4]. A hibajelet kiintegrálva megkapjuk azt a hibafeszültséget, ami szabályozza a VCO frekvenciáját. Ebből a hibajelből nyerhető ki a PCR_FO is [4].

A mérési eljárások egyenes utat biztosítanak a mérések megvalósításához. Viszont ezek hardveres utat mutatnak számunkra, amit jobb volna szoftveres úton megvalósítani. Napjainkban a számítógép használata mindennapossá vált, ezért lenne hasznos ezen mérések átültetése számítógépre.

6.8 Az Ethernet átalakító működése



41. ábra Az Ethernet-átalakító működése

A CableWorld Kft egyik tervezőmérnöke, Tóth Miklós fejlesztette ki az itt bemutatott párhuzamos TS-Ethernet, röviden ethernet-átalakítót. Nem szeretném érdemeit

learatni, de a mérésem könnyebb megértéséhez rövid ismertetése elengedhetetlen. A műholdról érkező jelet a CableWorld Kft által tervezett készülékek belső feldolgozásra párhuzamosítják. Ezt a jelet kell rákötnünk a paralel bemenetére. Innen a jel átmegy egy Xilinx IC-n, ami képes párhuzamos feladatokat egyszerre megoldani. Ezen keresztül másolódik a minta egy FIFO-ba. Nem lenne szükségszerű a "Vezérlőegység"-en átmennie a TS-nek, de a FIFO reset műveletének elvégzése közben a FIFO bemenetét be kell állítani alapértékre, addig nem kerülhet rá TS adat. Ez a kapuzás az egyik feladata a "Vezérlőegység"-nek. Emellett, ha az ethernet-modul kéri a következő 7*204 méretű csomagot, akkor ezt kiolvassa a FIFO-ból, hozzáteszi a segédbájtokat, SE-ket (1 bájtos folyamatosságjelzőt és 4 bájtnyi időbélyeget) és 4 bites (Ethernet 4 bites rendszerben dolgozik) formában elküldi az ethernet-modulnak. Minden egyes csomag (7*204=1428) kiküldésekor a folyamatosság számlálót növeli 1-gyel. Ezt figyelve a vételi oldalon, megállapíthatjuk a vétel hibamentességét. Az időbélyeget is ez a Xilinx áramkör állítja elő, mégpedig a TS-re rászinkrozinálva. A számlálót egy 25MHz-es órajel vezérli. A számláló állását 1428 beérkező bájt után olvassuk ki, és tesszük a kiküldendő SE közé. Az ethernet-modulnak a csomagkészítéshez szüksége van még az UDP címre, a kiküldendő adat CRC értékére (szintén a Xilinx számolja) és még több más segédadatra. A CRC-n kívül a többi adatot egy beépített PIC processzor tárolja, ami bekapcsolás után egy belső EEPROM-ba írja, ahonnan az ethernet-modul minden küldésnél kiolvassa az adatokat. A PIC-nek jelen kivitelben nincs több funkciója, de a jövőbeni egyik felhasználása lesz a Xilinx vezérlése.

Számunkra ennyi tudás elég is, hiszen az ethernet-modul az RJ-45-ös csatlakozón keresztül már tud kommunikálni a számítógéppel.

6.9 PCR_AC mérése

Ennek mérése a legegyszerűbb. Ha a számítógép megkapja a TS jelet és a jelfolyam sebességét, akkor a PCR hiba már számolható és kijelezhető. Persze felmerül az a gond, hogy a Windows nem párhuzamos működésű, azaz egyszerre csak egy dologgal képes foglalkozni. Ha csak a saját programunkat szeretnénk rajta működtetni, hardveres háttér nélkül, akkor a mérést nem végezhetjük folyamatosan, csak mintákat tudunk venni amit

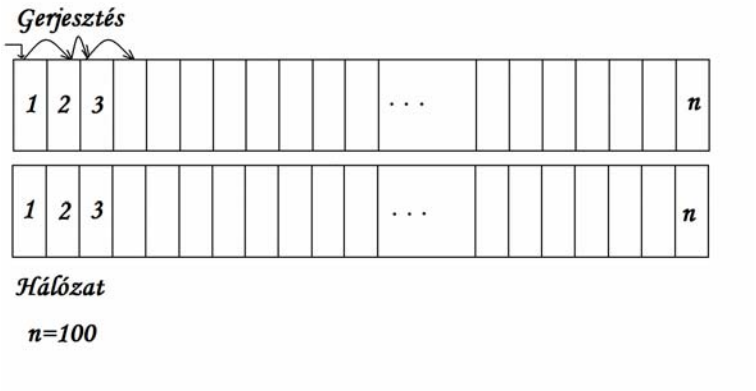
feldolgozhatunk. Feldolgozás alatt hiába üzen az Ethernet átalakítónk, a számítógép nem veszi a jeleket, amelyek felülíródnak egy kis idő múlva.

6.10 A többi mérés megvalósítása

6.10.1 A digitális szűrő megvalósítása szoftveresen

A többi mérés alapja a szűrés. Ennek módszerét ismerni kell ahhoz, hogy egyáltalán elgondolkodjunk a mérési módszerek átültetésén. Ennek alapja a szűrőkarakterisztika átviteli függvényének leírása. Ehhez a karakterisztika inverz Fourier transzformáltját kell vennünk. Ha használjuk a matlab nevű matematikai programot, akkor onnan megkaphatjuk kellő közelítési pontosságnak megfelelő számhalmazt, amellyel modellezhetjük a szűrőnk viselkedését. Azaz, ha szeretnénk meredekebb szűrőt megnalósítani, akkor sűrűbb mintavételt kell vetetnünk a programmal, és több számot is kapunk végeredményként. A szűrőkarakterisztika közelítéséhez szükséges mintaszám nagysága függ a határfrekvenciától, a zárócsillapítástól és az átmeneti tartomány meredekségétől. 1Hz-es aluláteresztő szűrő tervezése során 100 minta elegendő, míg 10mHz esetén 1000 minta is kevésnek bizonyul.

Mit is kell kezdenünk a kapott halmaz elemeivel? Ha a gerjesztést megszorozzuk az adott rendszer átviteli függvényével, akkor eredményül a rendszer válaszát kapjuk. Tároljuk el a szűrő átviteli függvényét leíró számokat egy tömbben (Hálózat). Elemszáma legyen 100 (1Hz-es szűrő). A vett PCR mintákat tároljuk el egy másik 100 elemű tömbben (Gerjesztés).



42. ábra A két 100 elemű tömbünk

Ha érkezik egy új minta, akkor léptessük a Gerjesztés tömbünket eggyel úgy, hogy az első helyiértékre kerüljön az újonnan érkező minta. Ezután az alábbi eljárást kell végigfuttatnunk, hogy megkapjuk az 1Hz-es aluláteresztőn átmenő PCR értékét:

```
procedure 1HzHighPassFilter(Halozat, Gerjesztes :Integer);
var Valasz, i:integer
begin
        Valasz:=0;
        for i:=1 to 100 do begin
                Valasz:=Valasz+Gerjesztes[i]*Valasz[i];
        end;
        for i:=0 to 98 do begin
                Gerjesztes[100-i]:=Gerjesztes[99-i];
        end;
end;
```

Azaz 100db PCR érték kell ahhoz, hogy megkapjuk a szűrt PCR értéket. A bemenő PCR a tömb felénél áll rendelkezésünkre, akkor érezteti hatását. 38ms-os PCR érkezések mellett az éppen érkező PCR érték szűrt eredményét 38ms*100/2=1,9s-os késleltetéssel kapjuk meg.

A digitális szűrés egyszerű megvalósítása így történhet szoftveres úton.

6.10.2 A VCO megvalósítása

Ennek részleteibe nem szeretnék belemenni. Gyors utánagondolás során rájöttem a digitális szűrő nehézségeire, ezért nem is mélyedtem bele oly mélységekbe. Az elv viszont nem oly nehéz. Bevezetek egy 27MHz-es változót. A két PCR érték különbségéből következtethetek a két érték között eltelt időre. Megnézem, hogy ebben az időintervallumban a 27MHz-es "belső órajelem" számlálója mennyit számlál. Összevetem a kettőt, a különbségből számíthatom az órajelem új frekvenciáját. Persze az integráláshoz (40.ábra) is több minta kell, tehát itt is be kell vezetni egy tömböt, ahol az előző mintákat

tárolom.
Legközelebb már az új órajelemmel számolom a két PCR elvárt különbségét.

Összességében ez a két egység szoftveres megvalósítása esetén elhagyhatnánk a hardveres mérés megvalósítását. Gondolnunk kell arra, hogy a szükséges eljárásokat belepréseljük a két érkező PCR érték közti szünetbe. Mielőtt erre rátérnénk, tekintsük át az átültetés elvét.

6.10.3 PCR_OJ mérési elve

Az előbb leírt eljárások segítségével könnyen átvihetnénk szoftveres méréssé, hiszen csak annyi a dolgunk, hogy vesszük a két PCR értékeket, és elmentjük. Az első érkezési idejét lejegyezzük, majd kivonjuk a másik érkezési idejéből, és megnézzük, hogy ez az általunk rezegtetett VCO mennyit számol. A kettő különbsége lesz egy PCR hiba. Ezt elmentjük a Gerjesztés nevű tömbbe és lefuttatjuk a szűrési ciklust. Így megkapjuk PCR_OJ-t, persze egy bizonyos késleltetéssel. Hátra van még a VCO-nk módosítása, tehát a Gerjesztés nevű tömböt ki kell integrálnunk, a megfelelő mértékben meg kell változtatnunk a VCO frekvenciáját.

6.10.4 PCR_DR mérési elve

Veszünk két PCR értékeket, és elmentjük. Az első érkezési idejét lejegyezzük, majd kivonjuk a másik érkezési idejéből, megnézzük, hogy ez az általunk rezegtetett VCO mennyit számol. A kettő különbsége lesz egy PCR hiba. Ezt elmentjük a Gerjesztés nevű tömbbe, és lefuttatjuk a felüláteresztést megvalósító ciklust. Így megkapjuk PCR_DR-t, persze egy bizonyos késleltetéssel. A VCO megváltoztatását itt is el kell végezni, persze ha egyszerre mérjük a kettőt (OJ-t és DR-t), akkor csak egyszer kell megváltoztatni.

6.10.5 PCR_FO mérési elve

Lényegi különbség csak annyi, hogy az integrátor után kapott eredményeket kell szűrnünk, azaz ezeket az értékeket tömbbe kell elmentenünk a digitális szűrés elvégzéséhez..

6.10.6 Összegzés

A szoftveres mérés megvalósítása nem oly nehéz, csak egy nagyon lényeges dologtól függ a működése. Ez pedig a sebessége, az általam megadott eljárások lefuttatási ideje. Emellett az operációs rendszerem működési folyamata sem elhanyagolható, hiszen a Windows működéséből következik, hogy minden részprogrammal foglalkozik egy kicsit, amit mi párhuzamos működésnek feltételezünk. Viszont sok művelet kiadása esetén már nem tudja a párhuzamosság látszatát kelteni. Szoftverünket tehát úgy kell megírnunk, hogy a bejövő adatokat fel tudja dolgozni addig, amíg a következő adathalmaz meg nem érkezik, mindamellett el tudja látni a háttérfeladatokat. Folyamatos mérést csak úgy érhetünk el, ha minél kevesebb feladatot bízunk a számítógépre. Ha ezt nem szeretnénk, akkor le kell mondanunk a folyamatos mérés elkészítéséről. Ha viszont nem folyamatosan nézzük a PCR alakulását, akkor a szűrő beépítése értelmetlenné válik. Ebből következik, hogy azt a hibafajtát mérhetjük, amelyiknél nem oly fontos a szűrő. Ez pedig a PCR_AC. A többi szétválasztásához szükségünk van több szűrőre és a folyamatos feldolgozásra. Diplomamunkám következő néhány fejezete a PCR_AC mérésének megvalósításáról fog szólni, az erre elkészített programmal végzek méréseket különböző TS jeleken.

6.11 A PCR_AC mérésének rövid leírása

A mérés elvének leírása előtt ismételjük át a PCR megtalálásának főbb lépéseit. Ha veszünk egy mintát, és kirajzoljuk a programfát (lásd. 28. ábra), akkor azt tapasztaljuk, hogy a PCR PID értéke megegyezik az azt követő első elementary adatfolyam PID értékével. Azaz minden programnak külön van PCR beültetése, ezt pedig a legnagyobb prioritással rendelkező adatfolyam tartalmazza. Tévé jel esetén a video adat, rádiójel esetén az audio adat tartalmazza. Ha egy program PCR beültetését szeretnénk leellenőrizni, akkor meg kell adnunk a PCR-t tartalmazó adatfolyam PID-jét, ki kell keresnünk ezeket, meg kell néznünk, hogy tartalmaz-e PCR mezőt.

6.11.1 A PCR mező megtalálása

Vegyünk egy példát. Mi kíváncsiak vagyunk a 12722-es programszámmal ellátott programunk PCR értékeire. Ekkor annyi a teendőnk, hogy megkeressük a PCR-PID értékét a PMT csomagból, ez egyezni fog a videojel PID-jével. Azaz a program videocsomagjai tartalmazhatnak beültetett PCR értékeket. Nem mindegyik csomag, hanem egy bizonyos időközönként megszakítva a videojelet, beszúrják a PCR értékeket. Ennek meglétét külön jelzik, amit nekünk figyelni kell.

TS packet layer - Adaptation field

Bájt	Bit	Megnevezés	Jellemzõk	Érték	Megjegyzés
1.	8	Synchron byte		h47	
2.	1	Transport error indicator	0-jó, 1-hiba van	0	
	1	Payload unit start indicator	1-ha PES vagy PSI kezdõdik 0 -ha nem a kezdõ bájt következik		1-pointer field
	1	Transport priority	nem használják, mindig 0		
	5	PID felsõ 5 bitje			21.old.
3.	8	PID alsó 8 bitje			PID Table 2-4.
4.	2	Transport scrambling control	00-not scrambled 01, 10, 11 user defined		null packet is 00
	2	Adaptation field control	01 payload only 10 adaptation field only 11 adaptation field + payload		hasznos adat, feldolgozó mezõ vegyes
	4	Continuity counter	mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		elõzõ mezõ 00, 10 nem nõ
Ez volt a packet header, eddig valamennyi packet felépítése azonos A következõ rész a packet payload. Ezen belül az adaptation field felépítését láthatjuk					
5.	8	Adaptation field length	feldolg. mezõ ? byte		
6.	1	Discontinuity indicator	1, ha csak AF van		Folytonosság állj!
	1	Random access indicator	1, akkor köv. PID csom. 1. Byte =tartalom info		
	1	Elementary stream priority	'1' video, '0' más adat		
	1	indicator	'1' prog. óra ref. érték található		
	1	PCR flag	'1' original prog. ref. érték van		
	1	OPCR flag	'1' splice countdown található		
	1	Splicing point flag	'1' 1 vagy több bizalmas adatbyte		
	1	Transport private data flag	'1' adatmezõ kiterjesztés		
		Adaptation field extension			
7.	8	Program clock reference base	Elsõ byte, összesen 33 bitnyi.		Ha PCR flag=1
8.	8	Program clock reference base	Következõ 8		
9.	8	Program clock reference base			
10.	8	Program clock reference base			
11.	1	Program clock reference base			
	6	reserved			
	1	Program clock reference extension	A programok referencia órájának		
12.	8	Program clock reference extension	kiterjesztése		
13.	8	Original program clock reference	A „fõ" referenciaórajel		Ha OPCR

43. ábra Részlet a függelékből

A 167-es PID című csomagokat kell néznünk, és ha találunk ilyeneket, akkor meg kell vizsgálnunk a következő bájtokat:

4. bájt 4-5. bitje jelzi nekünk, hogy van-e a csomagban adaptation field. Ugyanis ebben a feldolgozómezőben helyezik el többek között a PCR értékeket. Ha nincs, akkor keresnünk kell a következő csomagot, hátha abban megtaláljuk, ellenben a 6. bájt 4. bitjének 1-es értéke jelzi a PCR jelenlétét. Ha ez nem 1-es, akkor igaz, hogy találtunk feldolgozó mezőt, de ez nem tartalmaz PCR bájtokat.

6. bájt 4. bitjének 1-es értéke esetén a 7-8-9-10. bájt és a 11. bájt 7. (legfelső) bitje tartalmazza a PCR base (alap) értékét. A PCR base 90 kHz-es lépésekben változik. 7. bájt a

legnagyobb helyiérték. A 11. bájt 0. bitje és a 12. bájt tartalmazza a PCR extension (kiterjesztett) értéket. Ez 27MHz-es lépésekben változik. Ezeket kiolvasva megkapjuk az elküldött információt, ami már csak a feldolgozásra vár.

6.11.2 A mérés menete

A mérés első lépése, hogy beolvasunk egy akkora adatfolyamot, amelyben biztosan van 2 egymást követő PCR érték. Ez azért fontos, mert az ethernet átalakítónk 7*204bájtot, azaz 7 TS csomagot küld a számítógép felé egy lépésben. Viszont a PCR mezők videojel esetén kb. 38ms-onként vannak jelen. Ha vesszük a műholdról jövő TS csomagunkat, mely esetén egy csomag ideje 1/v_adat*204=42,93075us, akkor 38ms/42,93075us=894,4, azaz kb. 900 csomag választja el a két PCR értéket egymástól. Ez a távolság rádiócsatorna esetén (PCR-ek távolsága kb. 100ms) 2,5-szerese lesz. Ebből adódóan minimum 2500 csomagot kell vennünk ahhoz, hogy egy PCR hibát ki tudjunk számolni. Ezután meg kell mondanunk, hogy melyik PID-ű csomag tartalmazza a PCR értékeket. Ha ez megvan, akkor kikeressük a megadott PID-ű csomagokat, és megnézzük a fent említett bitek végigvizsgálásával, hogy melyik csomag tartalmaz PCR értékeket. PCR találása esetén elmentjük a PCR base-t, a PCR extension-t és a PCR kezdetének pozícióját. Miután ezzel megvagyunk, akkor ugyanezt meg kell ismételnünk, amíg a következő PCR értéket el nem mentjük. Ezután a következő számításokat kell csak elvégeznünk:

PCRint[1]:=(PCRbase[1]*300+PCRext[1])/27 000 000 [s]
PCRint[2]:=(PCRbase[2]*300+PCRext[2])/27 000 000 [s]
Ez a két érték másodpercben adja meg a PCR mezők által adott bájtok értékét. Megadja, hogy melyik másodpercben történt a beültetés.

DPCR:=(PCRint[2]-PCRint[1])*1 000 [ms]
Ez a számérték megadja, hogy mennyi a két érték közti távolság.

REF:=(pos[2]-pos[1])/v_adat[Bps]*1 000 [ms]
A REF ezáltal tartalmazza a ténylegesen eltelt időt a két PCR érték között.

PCR_AC:=(DPCR-REF)*1 000 000 [ns]

A kettő eltérése adja meg az általunk mérni kívánt hibát. Ha ez a két érték egyezik, akkor helyes volt a beültetés az adóoldalon.

Ebben a számításban a v_adat méréséről nem ejtettem még szót. Nekünk az ethernet átalakítónk 1-4. bájtja tartalmazza egy számláló azon értékét, ami a 7*204-es adat megérkezését jelzi. Ezt a számlálót egy nagy pontosságú 25MHz-es kristályoszcillátor vezérli. Ebből a sebességet a következő képlettel könnyedén kiszámíthatjuk:

v_adat=(két tetszőleges SEClk közt érkező bájtok száma)/((SEClk2-SEClk1)/25 000 000)

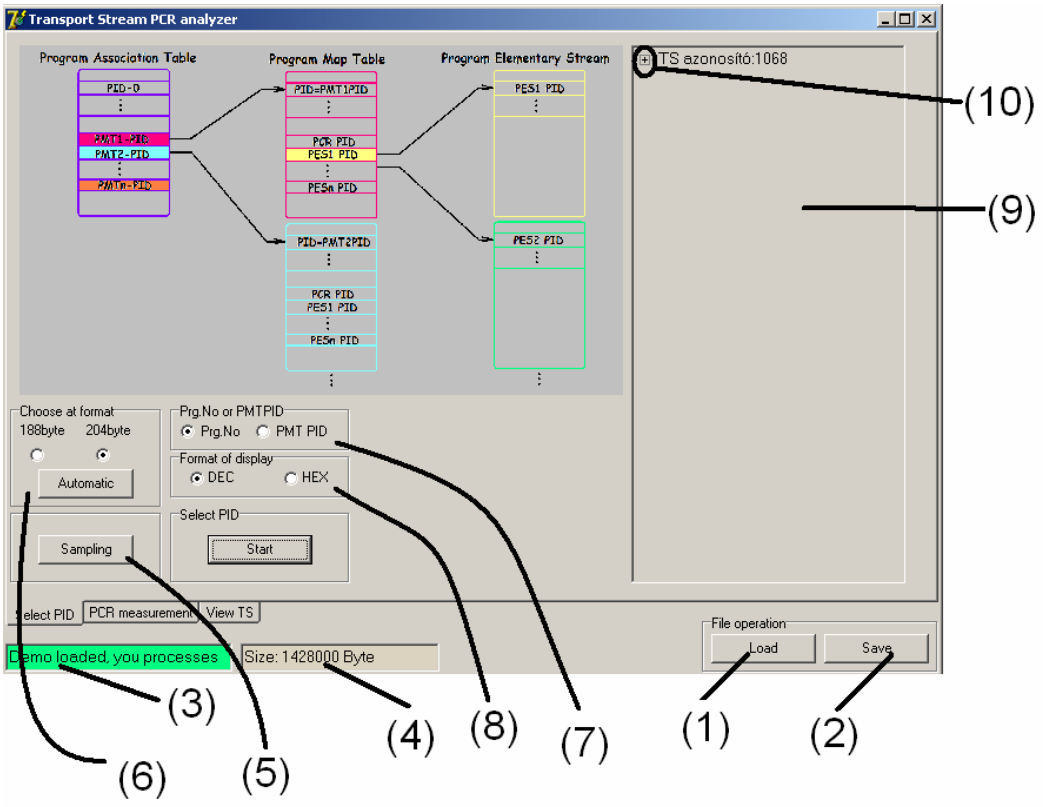
Veszünk két, egymástól minél messzebb lévő (az 1 bites hiba így egyre kevésbé okoz mérési hibát) számlálóértéket, amelyet a 25MHz-es órajel vezérel. Ebből számíthatjuk a két SEClk között eltelt időt. Ha ezzel elosszuk a két számlálóérték között érkező bájtok számát, akkor megkapjuk az adatok érkezési sebességét.

Ez a mérés elvben tökéletes, de gyakorlatban hibás eredményt szolgáltat. Nem teljesen hibásat, de a PCR méréséhez nem elegendően pontos értéket nyújt. Ebből következik, hogy nem használható eme sebességmérő mérésünkhöz mostani állapotában. Ennek kijavítása folyamatban van, de a mérésünket ettől függetlenül még elvégezhetjük. Az általam készítendő mérőszoftver tervezését nem befolyásolhatja eme hiba. Ezt azért merem ennyire nyíltan kijelenteni, mert a TS-jelet küldő hálózatok sebessége legtöbbször állandó. A műhold állandó sebességgel sugároz (nagyon kicsi az eltérés), nem is beszélve a DVB-T adásokról. Ezeket könnyűszerrel kiszámíthatjuk a megadott paraméterek segítségével. Egy TS jelet előállító mérőgenerátor, illetve egy remultiplexer kimeneti sebessége is könnyűszerrel beállítható, amit nagyon pontosan betart az adott készülék. Labori körülmények között ezen mérések elvégzése lehetséges, melyekhez a referencia-órajel nem létszükség. Persze a kapott eredmény kiértékelésekkor ezt nem szabad elfelednünk.

A következő három fejezetben tekintsük át az általam elkészített program használatát.

7 Az elkészített program működése

Elsőként tekintsük meg azokat a gombokat, kijelzőegységeket, amelyek mindhárom panelen azonosak. Ezután jöjjenek a panelek, és azok működtetése.



44. ábra A "Select PID" panel

8.1 Az állandó részek

(1)- Elmentett adatfájlok betöltése az újbóli feldolgozás céljából. Mivel programomat úgy írtam meg, hogy bármikor sebesség mérésére átalakítható legyen (az átalakító modul hibájának feltárása és kijavítása után), ezért nem csak a vett TS jelet kell betölteni, hanem az elmentett SE-t, azaz a segédadatokat, tartalmazó fájlt is. A gomb megnyomása után megjelenik egy Windows-os környezetű fájlnyitási ablak, ahol ki kell választani a feldolgozni kívánt TS-t tartalmazó fájlt. Ezután újból megnyílik ez az ablak,

ahol most a segédadatokat tartalmazó fájlt kell kiválasztani. A program a megnyitni kívánt fájlok adatait elmenti egy-egy adattömbbe, és ellenőrzi az SE adatait, hogy az elmentett minta hibátlan volt-e. Ha igen, akkor a (3)-as kijelzőn az alábbi szöveg jelenik meg zöld háttérrel: "Demo loaded, you processes!"

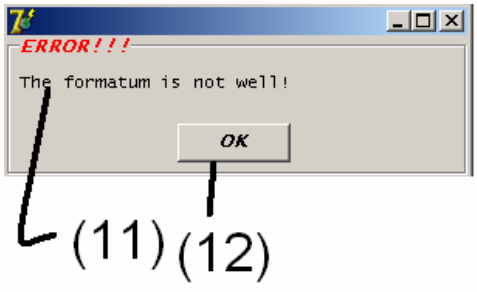
(2)- A vett minta elmentése. Az éppen aktuális, számunkra feldolgozás szempontjából hozzáférhető, adatok elmentése fájlba. Először a TS jelet menti el az általunk megadott helyre, az általunk megadott fájlnévvel, majd ugyanezt teszi a segédadatokkal. Ebben megintcsak egy előugró ablak van segítségünkre. A fájlok alapértelmezés szerint dat kiterjesztéssel kerülnek elmentésre.

(3)- Ez nem egy vezérlőgomb, hanem egy kijelző. Itt kerülnek kijelzésre az éppen folyamatban lévő, vagy aktuális adatműveletek.

Zöld háttérrel: "Demo loaded, you processes"-ha a betöltött fájl hibátlan,
"Data sampling ready"-ha a frissen vett jel hibátlan,
Sárga háttérrel: "Data sampling..."-ha a mintavétel folyamatban van,
Piros háttérrel: "The sample is not ready!"-ha a folyamatosságjelző hibázik.

(4)- Itt kerül kijelzésre az éppen aktuális (betöltött, vagy nemrég vett) TS tömbünk mérete bájtban megadva.

7.2 Az előugró hibaablak



45. ábra A hibaüzenet

Ha valami akadályozza az általunk kijelölt művelet elvégzését, akkor megjelenik az itt látható hibamenü, amely egyszerű működéssel rendelkezik.

(11)- A hibaüzenet itt jelenik meg, a hiba okára egy angol nyelvű mondattal utal.

(12)- Az OK gombra kattintva eltűnik az ablak, visszatérhetünk a programunkhoz.

7.3 A "Select PID" menü

(5)- Mintavétel gomb a fakirajzolás műveletének elvégzéséhez. A gomb megnyomása esetén 1s-on keresztül fogadja a hálózati kártyán keresztül a TS és az SE adatokat, melyeket elment két különböző helyre, majd ellenőrzi az SE-ben található folyamatosságjelzőt, és a (3)-as, ill. (4)-es kijelzőket frissíti a megfelelő módon.

(6)- Kézzel is kiválaszthatjuk a vett TS minta formátumát, de lehetőségünk van automatikus keresést kérni a gomb megnyomásával.

(7)- A megrajzolt fa struktúra megjelenítheti a program nevét, vagy pedig a programot leíró PMT tábla PID címét. Ezek közti váltást teszi lehetővé az itt található két rádiógomb.

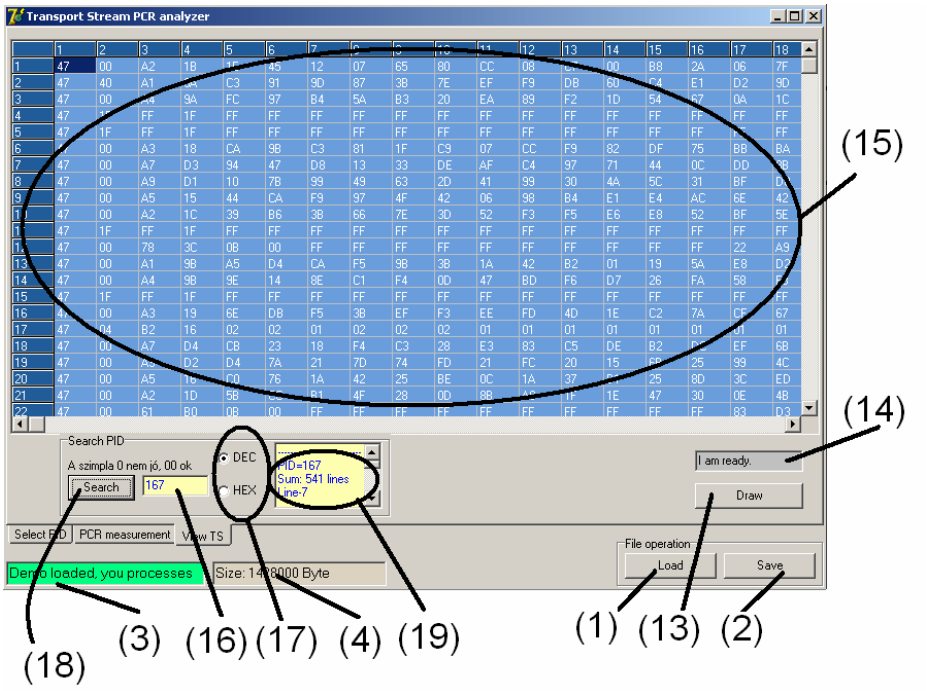
(8)- A feldolgozott TS csomagunkból kinyert értékeket megjeleníthetjük decimális illetve hexa formátumban is.

(9)- Ebben az ablakban kapjuk meg az elvégzett műveletek eredményét.

(10)- Eme + jelre rákattintva jeleníthetjük meg az alsóbb rétegeket, majd a - jelre történő ismételt kattintás tünteti el az adott részeket.

7.4 A "View TS" menü

Szakítsunk most a sorrendiség szabályaival, és mielőtt rátérnénk a PCR mérést megvalósító panel működésének elemzésével, tekintsük át ezt, hiszen ennek működése egyszerűbb.



46. ábra A "View TS" panel

(13)- A "Draw" gombbal elindíthatjuk az épp aktuális TS csomag táblázatos megjelenítésének procedúráját.

(14)- Ez a kijelző jelzi a művelet folyamatát. Ha végzett a felrajzolással, szürke háttérrel az "I am ready." üzenet jelenik meg.

(15)- Ebben a táblázatban található a felrajzolt TS bájtjai a szinkronbájtra rendezve.

(16)- Ezen mezőbe kell beírni a keresni kívánt csomagok PID címét.

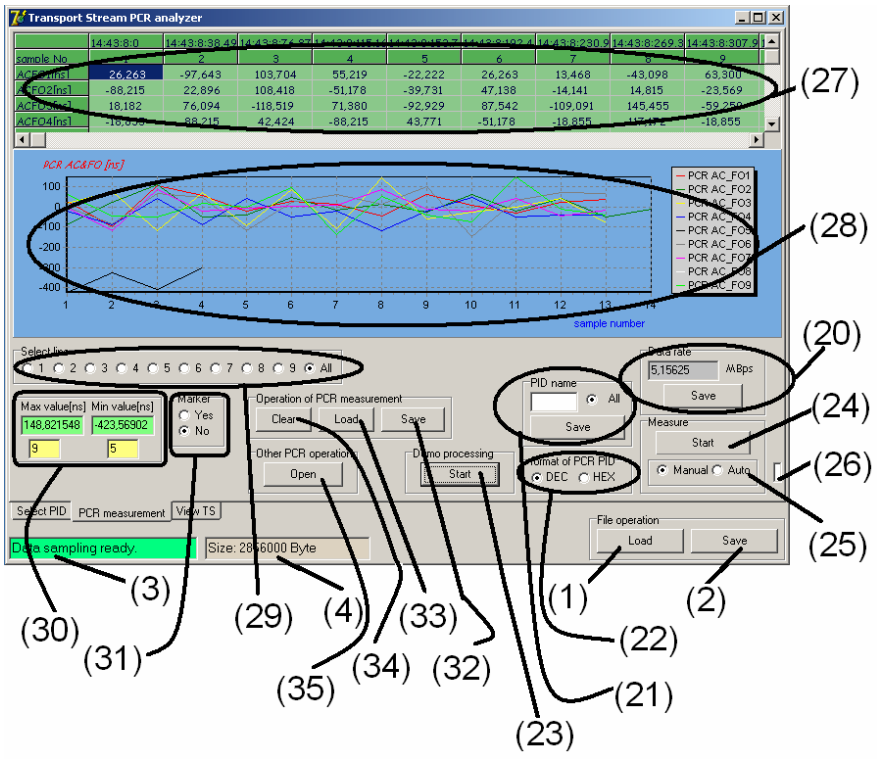
(17)- Választhatunk decimális, illetve hexa formátumú megadási módot is.

(18)- Eme gombot megnyomva elkezdődik a keresés, a végeredmény a (19)-es "Memo"-ban keletkezik.

(19)- A keresés végeredménye itt lelhető meg. Feltüntetjük a keresett PID értékét és azt, hogy mennyi csomagot talált a felrajzolt mintában. Ezek után megadja azokat a sorokat, melyekben megtalálhatjuk a keresett csomagokat.

7.5 A "PCR measurement" menü

Már csak ennek a panelnek a működését kell megértenünk ahhoz, hogy méréseinket elkezdhessük. Első ránézésre sok gombot tartalmaz, de ezek csak a kényelmünket szolgálják, nem pedig a működést bonyolítják. Ezek segítségével könnyen, gyorsan végezhetjük el a méréseket.



47. ábra A "PCR measurement" panel

(20)- Itt kell az előre kiszámított sebességet a mérésünkhöz betáplálni. Működése egyszerű, MBps formátumba várja az értéket, hibás karakter megléte esetén az alapból bennlévő sebességet tartja meg, miközben hibaüzenetet ad. A sebességet olyan pontosan kell megadni, amilyen pontosan csak tudjuk, minimum 5 tizedesjegy pontossággal. Ellenkező esetben ez a mérési hibát jelentősen növeli.

(21)- Lehet választani, hogy a vett TS jelfolyamból az összes programnak a PCR-jét mérjük, vagy csak egy, általunk itt kiválasztott program PCR hibáját jelenítsük meg. Ez esetben a (22)-es rész is értelmét nyeri, hiszen itt adhatjuk meg, hogy milyen formátumba kerüljön bevitelre a PCR PID értéke.

(22)- PCR PID bevitelének formátuma.

(23)- Ha fájlból betöltött jelet szeretnénk feldolgozni, akkor ezzel a gombbal tehetjük meg.

(24)- Az Ethernet hálózaton keresztül történő mérés indítása.

(25)- Automata-Kézi mérés indítása. Kézi mérés esetén mintavétel csak akkor történik, ha a (24)-es gombot megnyomjuk. Automata mérés esetén vesz a program egy mintát, azt feldolgozza, majd egy idő múlva megint vesz egy mintát az Etherneten keresztül.

(26)- Automata mérés esetén ez a lámpa mintavétel esetén sárga, feldolgozás esetén kék. Ezzel figyelhetjük programunk folyamatos működését. Ha ez nem villog, akkor a program valószínűleg megfagyott, újraindítása ajánlott. Legalább ebben hasonlít a programom a klasszikus Windows programokra.

(27)- A kapott eredmények táblázatos megjelenítése. A felső két sor tartalmazza a mintavétel idejét, melyet a számítógép órájából nyerjük és a mintaszámot. Mivel a mintavétel nem folyamatos, ezért nem a mintavétel ideje a grafikon vízszintes tengelye, hanem a mintaszám. Ha kíváncsiak vagyunk a hiba idejére, akkor ezt könnyen visszakereshetjük a mintaszám segítségével. Összes program vizsgálata esetén csak a "PCR_AC"-t tartalmazza, míg egy program vizsgálata esetén megjelenítjük a két egymást követő PCR értéket [s], a DPCR-t [ms] és a "PCR_AC"-t [ns] is.

(28)- A grafikus megjelenítés megoldása. Nagyítani úgy lehet, hogy a nagyítandó területet kijelöljük a bal felső sarokból a jobb alsó sarok irányában. Az alapállapotot ellentétes irányban, kijelölést prezentáló mozdulattal, történő egérművelettel történik.

(29)- Ennek a gombsornak igazán csak az "összes program mérése" esetén van lételeme, ezekkel csak az éppen látni kívánt mérés grafikonját választhatjuk ki.

(30)- Kikeresi a látható grafikonok min és max hibaértékét. Ezzel láthatjuk mérés során, hogy melyik program hibája a legnagyobb, illetve a legkisebb.

(31)- A grafikonokra rátehetjük az y értéket jelző markereket. "All prg" esetén hatástalan.

(32)- A táblázatot el lehet menteni pcr kiterjesztésű fájlba, ezt tehetjük meg a gomb megnyomásával.

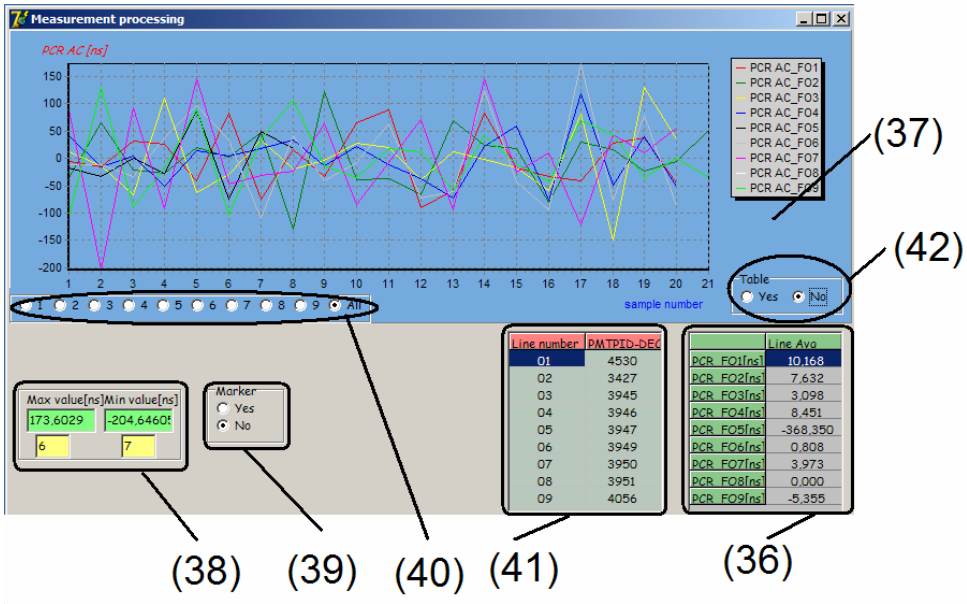
(33)- Az elmentett táblázatot lehet betölteni, mely művelet során a táblázat újból

kitöltésre kerül, és a grafikon is újrarajzolódik az elmentett táblázatból.

(34)- Ennek megnyomásával töröljük a táblázatot, és a grafikont, mindkettő alapállapotra áll vissza.

(35)- A mérések elvégzése közben merült fel az az igény, hogy a mért eredmények átlagát kiszámítsuk, és kivonjuk a mért eredményekből. Erről akkor majd bővebben fogok írni, most csak annyit szeretnék megjegyezni, hogy ezzel egy új, kisegítő ablakot kapunk, ahol a mért eredményeket számoljuk tovább.

7.3 A kiterjesztett ablak részei



48. ábra A PCR mérés feldolgozása

(36)- A mért értékekből számolunk egy átlagot, amit itt, táblázatosan jelenítünk meg.

(37)- Az adott grafikon értékeiből kivonjuk a hozzá tartozó átlagot, és újrarajzoljuk a grafikonokat.

(38)- Működése azonos a (30)-assal, csak a műveleteket erre a grafikonra értelmezi.

(39)- Működése azonos a (31)-essel, csak a műveleteket erre a grafikonra értelmezi.

(40)- Működése azonos a (29)-essel, csak a műveleteket erre a grafikonra értelmezi.

(41)- Itt látható a PMT PID-ek és a hozzá rendelt grafikonok számértéke. Ez a funkció csak akkor működik, ha érvényes TS adat van betöltve. Ha csak PCR mérést töltünk be, akkor ezen értékek nullák lesznek, vagy pedig az előző értékeket megtartja, ami lehet teljesen más PMT PID értékek is.

(42)- Ha kíváncsiak vagyunk az átlagszámítás után keletkező hibaértékekre, akkor megjeleníthetjük az így keletkező értékeket táblázatosan. A táblázat az ablak alsó részén lévő adatokat eltakarja, kikapcsolás esetén ezek újra megjelennek.

Most, hogy átnéztük a program működési folyamatát, elkezdhetjük méréseinket elvégezni először egy műholdról lejövő TS jelfolyamon, majd a magyarországi DVB-T adást teszteljük. A Rohde-Schwartz mérőgenerátor vizsgálata kihagyhatatlan, hiszen érdekes eredményt hoz számunkra. Végül az általam elkészített új programcsomag tesztelését követhetjük végig, amit egy kínai remultiplexerrel készítettem egy műhold jelét és a mérőgenerátor jelét felhasználva.

8 Mérések végzése az elkészített program segítségével

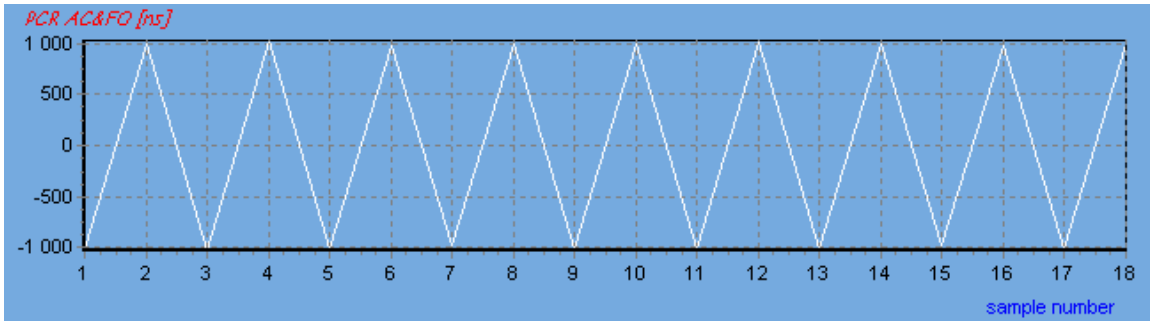
8.1 Ellenőrzés

Mielőtt az élesben mért értékeket teljes egészében elfogadjuk, ellenőrizni kell a mérési eredmények helyességét. Ehhez szükséges egy olyan generátor, amivel PCR jittert tudunk előállítani, aminek mértékét mi állítjuk be. Erre alkalmas a Rohde&Schwartz MPEG-2 mérőgenerátora. A rendelkezésemre álló készülék típusa R&S DVG. Beállítottam a generátoron egy programcsomag kiküldését 204-es formátumban, az adatsebesség 41,25Mbps. Az így kapott TS jelet egy ASI átalakítóra vezettem, ami az ethernet-átalakítónkhoz szükséges csatlakozóra tette át a TS jelet. Innen már rácsatlakozhattam az ethernet-átalakító bemenetére. A jelet átalakítás után (segédadatok hozzáadása, ethernet struktúra) egy helyi számítógép hálózati kártyáján keresztül vettem a programom segítségével.

Az összes itt bemutatott mérés táblázata és grafikonja előhívható mérőprogramom

segítségével, ezért minden grafikon előtt megadom az elérési utat, ahonnan behívható az eredmény.

A mérőgenerátoron 500ns-os jittert állítottam be (Mentések\Rohde&Schwartz\PCR \18jo500ns.pcr). Programommal viszont a következő ábrát kaptam:

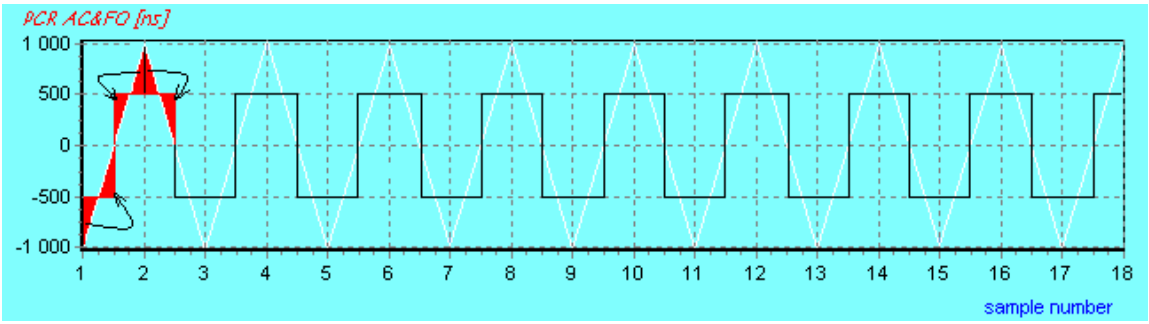


49. ábra Rohde 500ns-os jittert tartalmazó jele

Pont a kétszeresét mértem a programom segítségével. Megkezdődött a hibakeresés, hiszen egy kettes osztó beültetése nem megoldás a hiba kijavítására. Első lépésként elővettem a Rohde DVM-100-as készülékét, amelyet TS jel állandó figyelésére, úgynevezett monitoring-ra fejlesztettek ki. Ez a készülék alkalmas a PCR érték figyelésére is. Lényegében egy Windows operációs rendszerrel ellátott felügyeleti számítógép mérőműszer (19"-os) formában kiadva. Bekapcsolva, és megmérve az előbb vett jelet, arra a megállapításra jutottam, hogy 500ns-ot mér a Rohde TS analizátora. Még szerencse, hogy kompatibilis a két készülék. Hiszen elég furcsa volna, hogy egy gyártó két készüléke teljesen mást mutat. Ellenben azt észrevettem, hogy a TS felügyelőrendszere hol +500ns körüli, hol -500ns, hol pedig 0ns körüli értéket mér. Ez kíváncsivá tett, így megfigyeltem, hogy a mutatott értékek között 1 másodperc telik el. Azaz nézi a PCR értékeket, melyeknek valamilyen átlagát veszi, és azt veszi PCR értéknek. A cég nem közli a mérési módszerét, így csak következtetni tudok a különbség okára.

Mérésem során én bizonyos időközönként egy-egy értéket kapok. Az általam mutatott grafikon nem azt jelenti, hogy az egyes mérési pont és a kettes mérési pont között eltelt időben a PCR hiba így változik, hanem csak azt segíti, hogy ne egy-egy pontot keresgéljünk, lássuk a két egymást követő PCR értéket egyszerűen, gyorsan. Mégis ez a megjelenítési forma segített nekem abban, hogy meglássam a különbség egyik feltételezett

okát. Annyi biztos, hogy az én mérési eredményeim az adott pontban kiszámított értékkel egyezik. Persze ezzel a megoldásommal feladtam azt a lehetőséget, hogy mérésem teljesen folyamatos legyen. Veszek egy általam előre meghatározott mintát, és az abban található összes PCR hibát megjelenítem táblázatos és grafikus formában. A kapott értékeket egy lineáris vonallal kötöm össze. Mint látható, a mérőgenerátor 500ns-os jitter beállítás esetén egyszer +1us-os, egyszer -1us-os hibát ültet be a mérésbe körülbelül 32ms-onként. Ha a hibákat időben eloszlatjuk, akkor 500ns-os négyszögjelet kapunk. Most már legalább azt látjuk, hogy mitől mutathat 500ns-ot a készülék. Már csak arra kell megoldást adnom, hogy miért lehet +-500 illetve 0ns. Vegyük az érkezési időket (DPCR) átlagosan 32ms-nak. Attól függően, hogy a műszer mérési periódusában mennyi +1us-os, és mennyi -1us-os jitter volt, kaphatunk a területek előjeles összegére +32ms*500ns-ot, 32ms*-500ns-ot, illetve 32ms*0ns-ot. Tehát a mérés tartama alatt fogadott értékekből számított terület (integrálás) átszámítása egy átlagos PCR távolsággal (DPCR). Én így értelmezem a Rohde&Schwartz által mutatott értékek forrását.

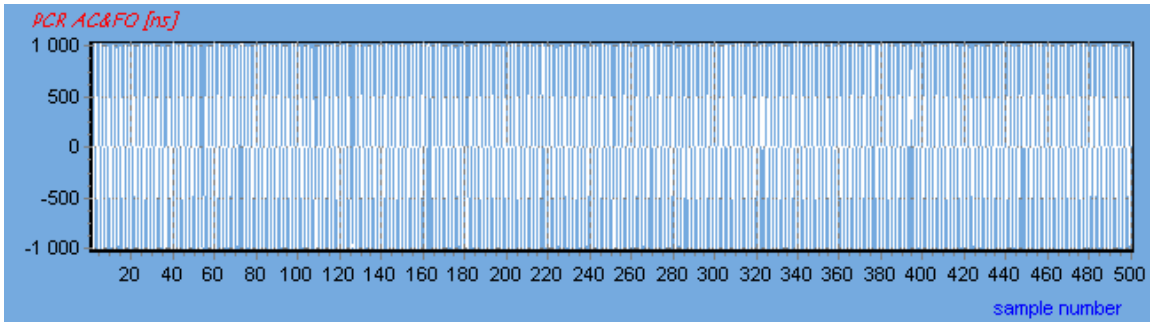


50. ábra A Rohde&Schwartz lehetséges mérési elve

Ebből is következik, hogy a PCR hiba mérése számítógépes módszerrel nem mérhető folyamatosan. A Rohde mérőkészülék nem állandó, folyamatos beültetési hibát mér, hanem egy bizonyos időközönként keletkező átlagos hibát mér. Az én programom ellenben nem tud annyira folyamatos mérést végezni, mint az előbb említett, de a vett mintában talált összes PCR hibát kiveszi, ami azt jelenti, hogy programom "csúcsértéket" mér.

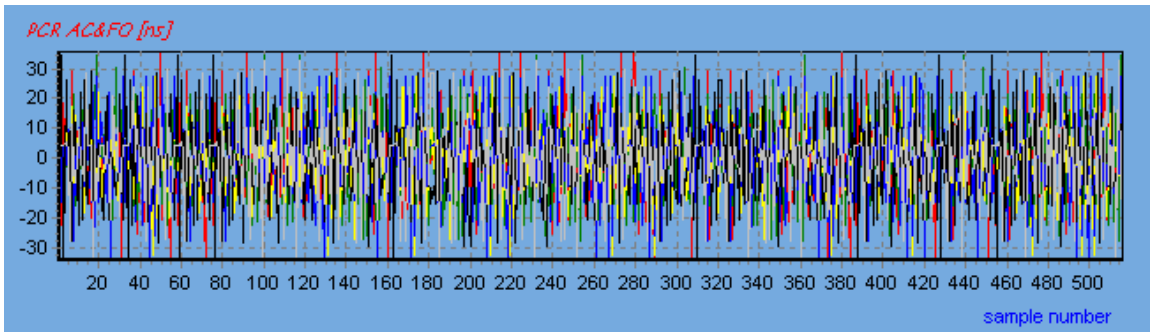
Most, hogy ezt a gondot megoldottnak tekinthetjük, láthatjuk a következő ábrán azt, hogy a mérőgenerátor folyamatosan váltogatja a beültetési hiba előjelét, viszont a

programom a két mintavétel határfelületén nem ezt jelzi. Ez abból adódik, hogy az előző mintavétel végén egy -1us-os hibát mértem, a következő mintavételem esetén, az amúgy váltakozó hibát adó generátor TS folyamából, úgy mintavételeztem, hogy a feldolgozott mintahalmazban ismét -1us-os hibát észleltem. Programom ezt a két értéket összeköti, mintha egymás után következnének, de tudnunk kell azt, hogy ez nem igaz. Viszont az itt látott eredmény mutatja, hogy a generátor tökéletesen adja a hibaértékeket, időben állandó jelleggel (Mentések\Rohde&Schwartz\PCR \500jo500ns.pcr).



51. ábra A Rohde generátor jelének mintavett mérése

Egy mérőgenerátor esetén lényegi kérdés az, hogy mennyire pontos a beültetés, ha nem kívánunk jittert a TS folyamunkba. Ezt jelzi a következő ábra. Mint azt már leírtam, a 27MHz-es órajel lépcsőssége miatt 37ns beültetési hiba bármikor keletkezhet egy bites csúszás következményeként. A Rohde maximálisan mért hibája ezzel egyenlő. Tehát a beültetés nagyon pontos. Egy rendszertől, mármint elsődleges beültetést végző, 500ns-os hiba a megengedett mindkét irányban (Mentések\Rohde&Schwartz\PCR \500jo.pcr).



52. ábra A generátor jittermentes jelének beültetési hibája

8.2 Egy műholdas rendszer vizsgálata

Műhold pozíció: 19,2Kelet, azaz az Astra 1 műholdakra pozícionáltunk.
Polarizáció: V
Frekvencia: 11,778GHz
Transzponderszám: 68

Ezen paraméterek beállításával vevőkészülékünk az Astrát fogja, ahol a 68-as transzponderen lévő TS-t veszi. Ha kíváncsiak vagyunk, akkor előveszünk egy újságot, melyben le van írva az összes fogható műhold programterve (ilyen például Európában az INFOSAT EUROPE), akkor rákereshetünk a megfogott csatornára, és megnézhetjük, hogy mit tartalmaz.

A kapott eredmény:

TVBS	162-video PID
Travel	163-video PID
TCM(FRA)	169-video PID
TCM(ESP)	164-video PID
CNN	165-video PID
Cartoon Network	161-video PID
Bloomberg TV	167-video PID
CNN Radio	101-audio PID

Ha összevetjük az itt kapott eredményeket a létrehozott programfa eredményével (26. ábra), akkor láthatjuk, hogy tényleg azt találta meg a programvisszafejtő eljárásunk, amit kellett neki. Persze az újságot nem csak azért kellett elővenni, mert nem voltunk biztosak a régebben tárgyalt program visszafejtésben, hanem ez tartalmaz olyan információkat is, amelyek segítenek kiszámítani a műholdas jel adatsebességét. Mint azt már említettem, a sebességmérő nem tökéletes, ezért kívülről kell megadnunk a program számára a mérendő TS sebességét. A mérőgenerátor esetében ez nem okozott gondot, hiszen amit beállítottunk a mérőgenerátoron, azt kellett a programunk számára is megadni. De ez esetben vagy megmérjük egy frekvenciamérővel a frekvenciát és abból számolunk vissza, vagy a megadott műholdparamétereket használjuk fel a sebesség kiszámítására.

A számításhoz szükséges adatok:

szimbólumsebesség: 27 500 000bps

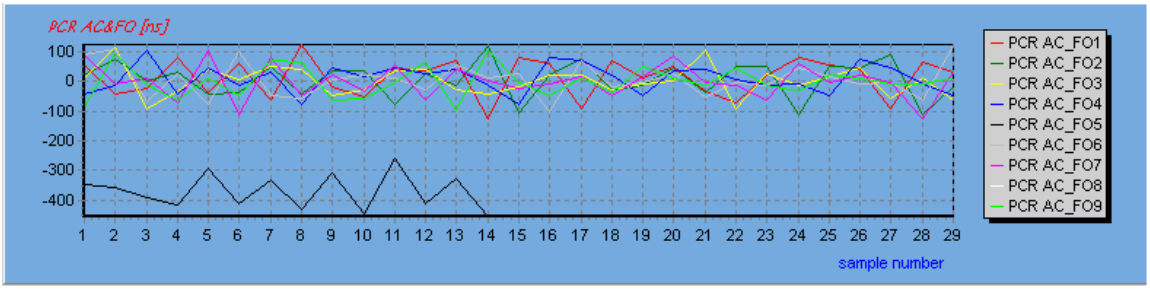
Viterbi kódarány: 3/4

szimbólum adattartama: 2

Ebből: v_adat=szimbólumsebesség*Viterbi-kódarány*adattartalom=41,25Mbps=
=5,15625MBps.

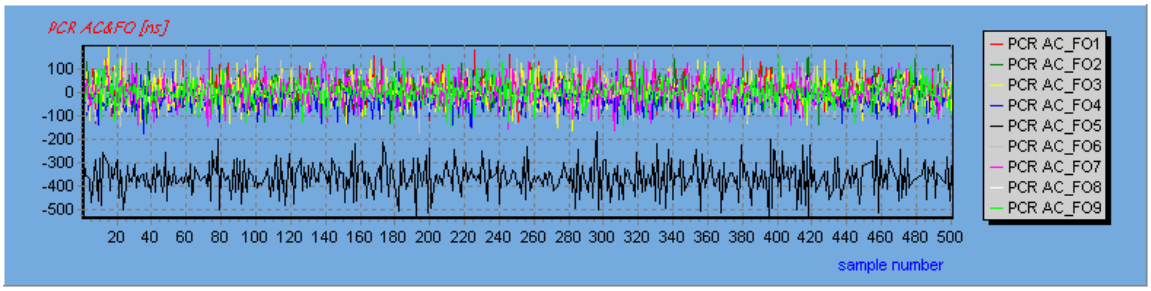
Ezt a sebességet kell megadnunk a programunk számára a mérés elvégzése érdekében. Ezt az értéket állítottam be alapértéknek, ami a program elindításakor automatikusan beíródik, illetve hibás szimbólumot tartalmazó sebesség bevitelekor keletkező sebességérték.

Most már megvan a keresett műhold, tudjuk a sebességét, kell a mérés összeállítási struktúrája. A műholdról vett jelet egy, a CableWorld Kft által kifejlesztett digitális műholdvevővel veszem, amellyel ráállok a 68-as "csatornára" és a vett TS-t rávezetem az Ethernet átalakítóra. Mérésem eredménye itt látható (Mentések\Satelit\PCR \53_abra.pcr):



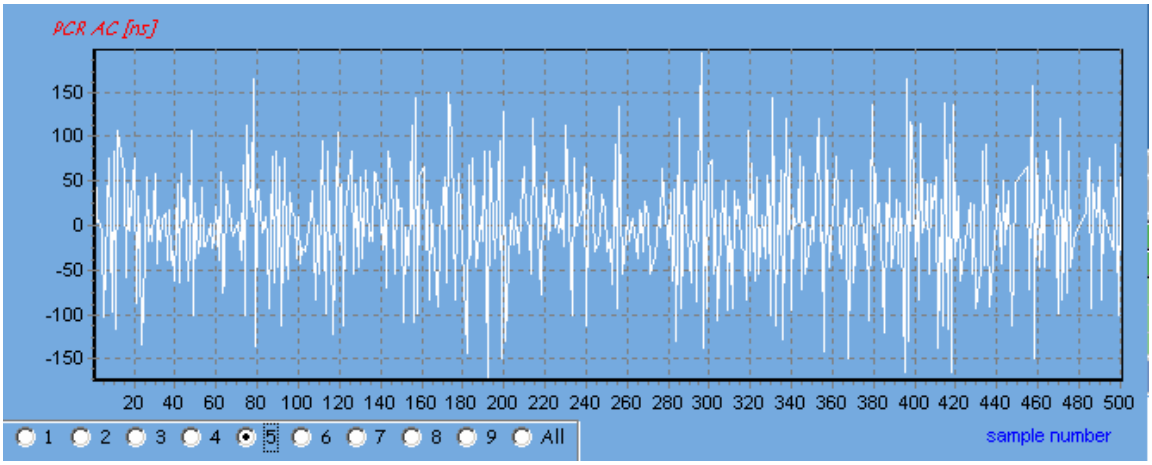
53. ábra Egy műholdas jel PCR hibája

Az ábráról leolvasható, a videojelek PCR beültetése igen pontos, nem sokkal több, mint 100ns a hiba. Audiojel esetén ez -400ns körül van. Az is látszik, hogy a vett mintába található PCR értékekből kevesebb van egy rádióprogram esetén, mint tévéprogram esetén. Ez természetes is, hiszen az átlagos táv 100ms, míg tévéprogram esetén 40ms-nyi. Mivel az átviteli úton nem keletkezhet hiba, az itt mért érték műhold esetén a PCR_AC-vel egyezik meg. Vegyünk egy nagyobb mérést, és vonjuk le a következtetéseket (Mentések\Satelit\PCR \Satelit500.pcr).



54. ábra Egy műholdas jelen végzett mérési sorozat

Az rögtön látszik, hogy az audiojel hibája 400ns körül ingadozik. Ez azt jelenti, hogy az adóoldalon a PCR beültetés nem 27MHz-es órajellel történik, hanem más frekvenciával. Ha kiátlagoljuk a mérési eredményeinket, akkor abból következtethetünk a vevőkészülék által észlelt jitterre. Ez azért nem fog egyezni az itt mértre, mert a vevőkészülék rá fog szinkronizálni a beültetés órajelére Mentések\Satelit\PCR \Satelit500.pcr kiátlagolva).

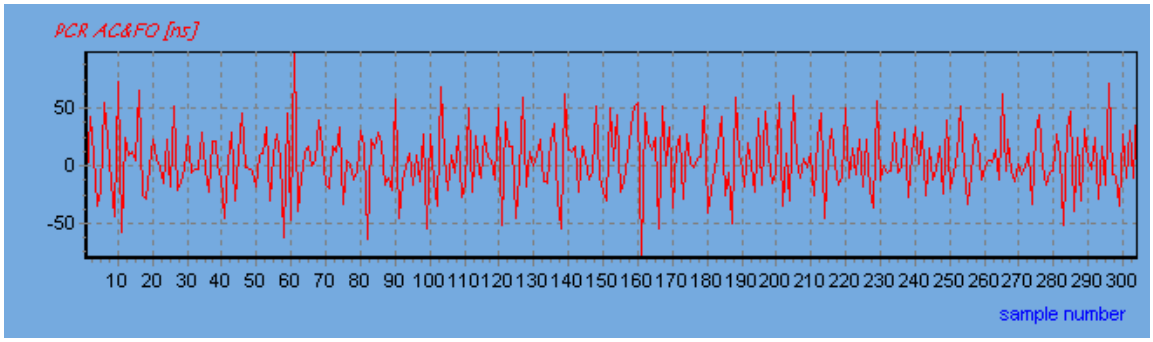


55. ábra Egy rádiójel PCR jitterének alakulása

Jól látható, hogy szinkronizálás estén a hiba már csak akkora lesz, mint normál videojel esetén. Ez viszont már nem nevezhető accurancynak, hiszen nem a beültetésből adódik (ez volt az átlagszámítás előtt), ez már az a remegés (jitter), amit a vevőkészülékünk észlel. Az átlag, amivel el van tolva a mért eredmény a 0 értéktől (54. ábra), -357,98ns. Ha vesszük azt a mérési eredményt, hogy 2 PCR érték között 100ms telik el átlagosan, akkor 100ms alatt a PCR értékek által jelzett távolság 357,98ns-mal kevesebb. Azaz 357,98ns*27MHz/100ms=96,65Hz-vel kisebb frekvencián rezeg az adóoldali oszcillátor.

Ebből az következik, hogy az átlagos frekvenciaeltérés (PCR_FO) 96,65Hz volt a mérésem idején.

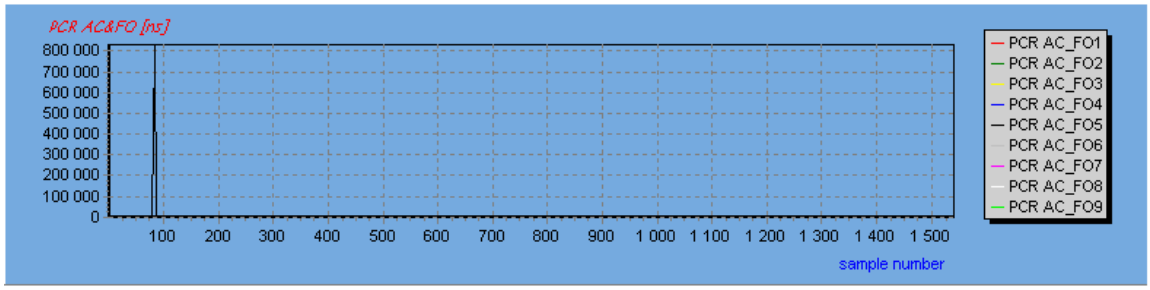
Ha megvizsgáltuk a rádiócsatorna PCR beültetését, akkor ezt tegyük meg egy videocsatorna esetén. Választásom a 167-es PID-ű Bloomberg TV vizsgálatára esett (Mentések\Satelit\PCR \pid167.pcr).



56. ábra A 167-es PID-ű videocsatorna vizsgálata

Ennek beültetési jósága vetekszik a Rohde mérőgenerátorának beültetési jóságával, hiszen a beültetés hibájának abszolútértéke nagy részében 50ns alatt van.

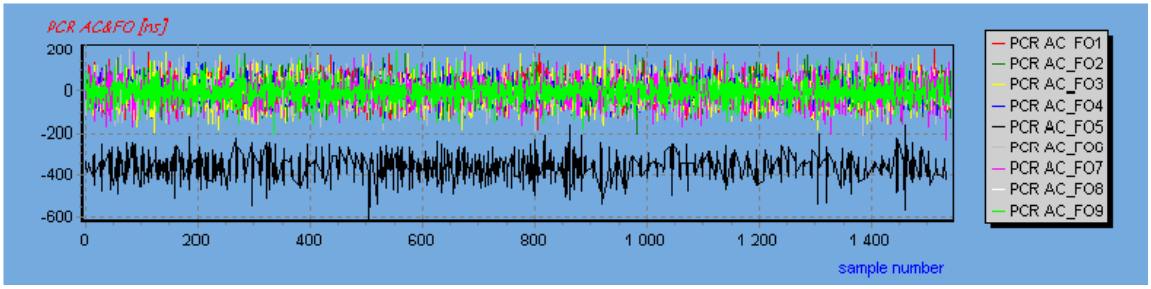
Természetesen nem mindig ilyen tökéletes a beültetés, tekintsük meg a következő ábrát (Mentések\Satelit\PCR \programváltás.pcr).



57. ábra Egy műsorváltás hibája

A nagy különbség a századik mérésszám közelében történik, ahol 2 PCR érték által megadott DPCR 800us-mal nagyobb, mint a tényleges távolság. Ez azt jelenti, hogy sokkal nagyobb a PCR2-es érték az előtte lévő PCR1-es értéktől. Viszont az utána következő PCR3-as érték már átlagos távolságra van az előtte lévő PCR2-es értéktől. Ez nem azt jelenti, hogy hibás PCR érték a PCR2-es, hiszen az utána következő hozzá képest

normálisan érkezik. Itt valószínűleg egy másik műsor, reklám kezdődik, melynél a PCR kezdőérték változik. Hiszen könnyebb beültetni egy nagyobb PCR hibát, amit a vevőkészülék oszcillátora amúgy sem tud követni, mint az előre elkészített program összes PTS, DTS idejét átírni a PCR-nek megfelelőnek. Ha kinagyítjuk az alsó régiót, akkor azt láthatjuk, hogy az előbbi hibát leszámítva tökéletes a PCR beültetés. Ez az előző gondolatmenetemet látszik igazolni.



58. ábra A műsorváltás előtt és után

8.3 A DVB-T PCR hibájának vizsgálata

Magyarországon megkezdődtek a földfelszíni digitális műsorsugárzásának tesztelései. Budapesten és környékén már fogható az m1, m2 és a Duna TV, vidéken a Kab-hegyi tv állomás kezdte meg digitális kísérleti tv sugárzást. Ahhoz, hogy megvizsgáljuk a PCR beültetést, meg kell határoznunk a bejövő jel sebességét. Ez azért szükséges, mert nem tudjuk megmérni a programunk ezen változatával. Azért nem kell elkeseredünk, mert vannak módszerek a sebesség meghatározására.

Az első módszer a megadott paraméterekből történő számítási mód. Nézzük, hogy milyen műszaki adatokat ad meg az Antenna Hungária Rt a budapesti adójára [14]:

f-a sáv:	UHF 43&51-es csatorna,
ERP	1kW,
üzemmód:	8k,
moduláció:	64QAM,
hibajavítás kódaránya:	2/3,
védelmi intervallum:	1/32.

A sebesség meghatározására az alábbi képletet felhasználhatjuk [9]:

$$v_{jelfolyam} = \frac{m \cdot B}{1+\Delta} = \frac{m \cdot \frac{N_{tényleges}}{N} \cdot \frac{64}{8}}{1+\Delta}$$

Ahol m az egy szimbólummal átvitt bitek száma (64QAM esetén ez 6), B az elfoglalt sávszélesség, Δ pedig a védőintervallum nagysága. A B meghatározásához szükségünk van a vivőszámra (8k esetén ez 8192), az ebből ténylegesen felhasznált vivőszámra (6785), és a mintavételi frekvenciára, aminek meghatározása a 64/8 művelet elvégzése ad meg MHz-es dimenzióban. A 64 a 64 állapotból ered, míg a 8-as az UHF csatorna 8MHz-es sávszélességéből. Ezekkel kiszámítva a sebességet:

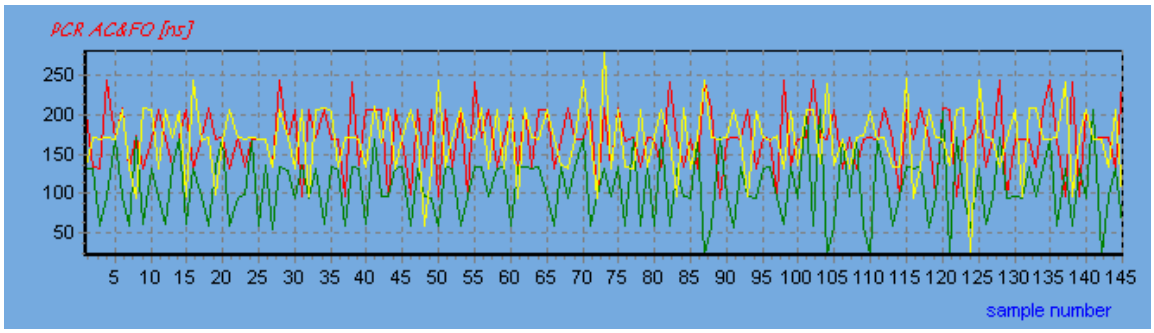
$$v_{jelfolyam} = \frac{6 \cdot \frac{6785}{8192} \cdot \frac{64}{8}}{1+\frac{1}{32}} = \frac{6\,bit \cdot 6{,}6259\ MHz}{1+\frac{1}{32}} = 38{,}551136\ Mbps$$

2/3-os hibajavító kódot használnak, amit csak a PCR beültetése után tesznek hozzá, így ezzel az értékkel meg kell szoroznunk a kapott eredményünket és át kell számolnunk MBps-ra.

$$v_{jelfolyam} = 38{,}551136\ Mbps \cdot \frac{2}{3} = 25{,}7\,Mbps = 3{,}21259\ MBps$$

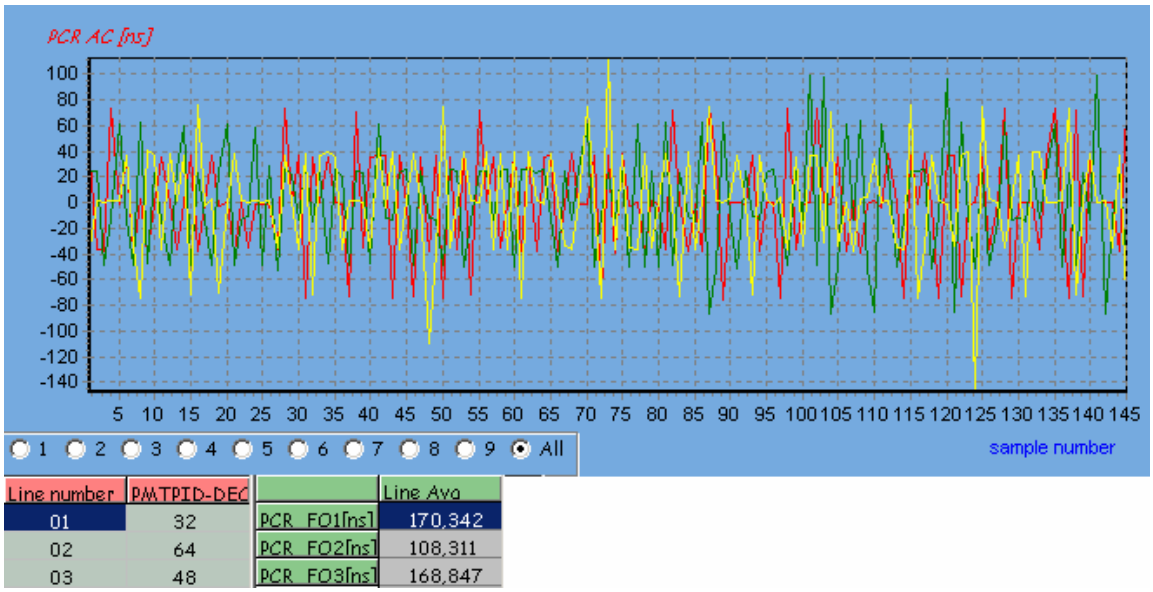
Ez a számítási mód elméletileg helyes eredményt ad, de ha meg tudnánk mérni egy mérőműszer segítségével, akkor sokkal pontosabban meghatározhatnánk. Ez azért lényeges, mert a mérés nagyon érzékeny a jel sebességének pontatlan megadására. A mérést úgy végeztük el, hogy egy digitális oszcilloszkóppal megnéztük a bejövő adatokat, melyeket lementve lekérdeztük a frekvenciájukat. Az így kapott sebesség: 3,27274MBps. Ez a tényleges sebessége a budapesti DVB-T adásnak. A sebesség tudatában már elvégezhetjük a mérésünket.

Az összeállítás itt sem bonyolult, hiszen egy meglévő rendszert kell vizsgálnunk. Az antennáról lejövő DVB-T jelet egy CableWorld-os DVB-T vevővel vettem, amelynek ASI kimenetén megkapott jelet visszaalakítottam paralel jellé, majd az ethernet-átalakítóra vezettem a TS jelet, ahonnan egyenesen a számítógép hálózati kártyájára került a segédadatokkal ellátott TS jelem (Mentések\DVB-T\PCR \dvpt.pcr).



59. ábra A DVB-T PCR_AC hibája

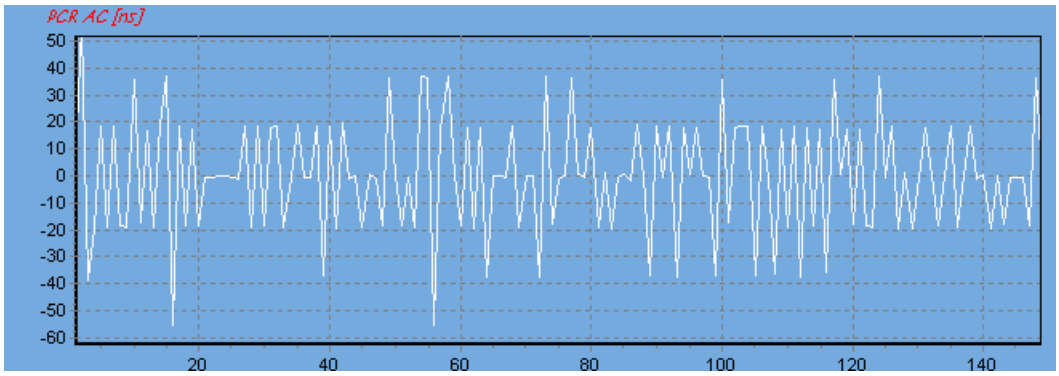
Az ábrán mindhárom műsor PCR hibája látható. Mivel az átviteli úton nem történik olyan esemény, ami a PCR beültetést megrontaná (csomag eltolások, beszúrások), így a mért PCR hiba az adóoldali beültetésből származik egyedül. Ebből az következik, hogy a mért eredmény a PCR_AC-vel egyezik meg, aminek abszolútértéke nem lépheti túl az 500ns-ot. Látható, hogy a kapott eredmények nem a 0 tengely körül ingadoznak, azaz a kapott eredményben van egy kevés PCR_FO is. Ha a kapott eredményt kiátlagoljuk, akkor abból következtethetünk a PCR_FO átlagos értékére is.



60. ábra A DVB-T kiátlagolt PCR hibája

A PCR-ek átlagos távolsága 34ms, a 32-es program átlagos hibája 170ns, azaz az adóoldali oszcillátor átlagosan 170ns*27MHz/34ms=135Hz-el nagyobb frekvencián rezgett

a mérésünk idején. Megnézve ennek a programnak a kiátlagolás után kapott beültetési hibáját:



61. ábra A 32-es program PCR_AC-je a vevőkészüléknél

Mint az ábrán látszik, a vevőkészülékünk csak +50 és –60ns között ingadozó PCR értéket veszi észre, mivel belső oszcillátora rászinkrozinálódott az adóoldali oszcillátorra (27MHz+135Hz) és az ábrán már a tényleges jittert kapjuk meg.

A DVB-T mérése közben szerzett tapasztalataimat szeretném megosztani a mérés befejeztével. A DVB-T TS-ében megtalálható táblák a PAT, a PMT-k, a PES csomagok és az EIT. Az EIT-ben „műsorújságot” visznek át, azaz elküldik az aznap nézhető műsorok tartalmát, időpontját. Mérés közben észrevettem, hogy a PAT és PMT táblák küldése sokkal ritkábban történik, mint műholdas átvitelnél. Kiszámolva kiderült, hogy 472ms körüli két PAT közti távolság (műholdnál átlagosan 60ms körüli). A szabvány 500ms-ot jelöl meg maximális távolságnak [3], ami azt eredményezi, hogy programváltásnál (más TS-t akarunk venni) fél másodperc a minimális átkapcsolási idő. Mivel csak egy TS-t sugároznak pillanatnyilag, így ez nem zavaró egy néző számára, de az általam készített program nem mindig volt képes kirajzolni a programfát. Ezt kiküszöböltem azzal, hogy nagyobb mintát vettem a TS-ből, amibe biztosan megtalálható a feldolgozáshoz szükséges tábla. Érdekes probléma volt, ami segítette programom tökéletesítését is.

8.4 Egy remultiplexer vizsgálata

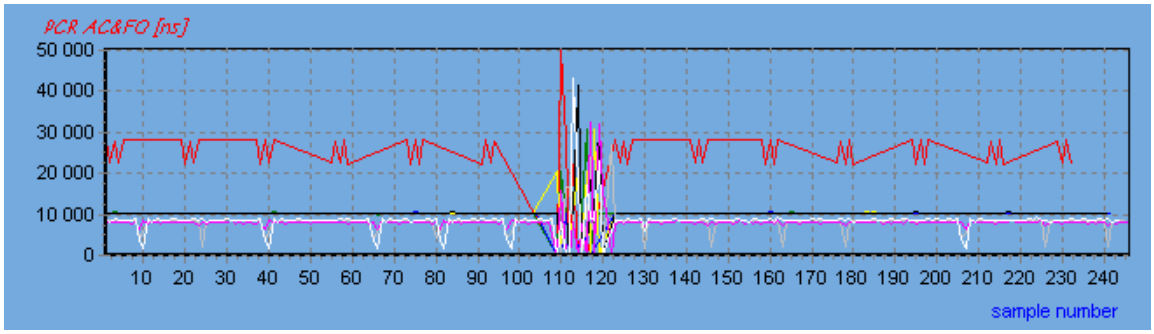
A méréseket egy igen érdekes berendezés vizsgálatával fejeztem be. Ez pedig egy remultiplexer vizsgálata. Nem mindig megoldható az, hogy a PCR beültetés után elkészült jel azonnal az előfizetői végpontokhoz eljusson. A kábeltelevíziósok eddig úgy készítették programcsomagjaikat, hogy több helyről vették a programcsomagokat (műholdak jelei, földfelszíni műsorsugárzás jelei), majd ezekből kiválasztották az előfizetőiknek szánt programokat, melyeket analóg módon tovaküldték az előfizetők felé. Ezt az analóg jelet titkosítani, ezáltal több programcsomag kiválását megvalósítani nagyon nehéz és könnyen feltörhető (egyszerű szűrő). Remultiplexer segítségével viszont készíthetünk saját digitális programcsomagokat, melyeket könnyen titkosíthatunk. Ezáltal akár 8-szor több programot leszünk képesek továbbítani koaxiális vagy optikai hálózatunkon, és a programcsomagokat tetszés szerint kódolhatjuk. Egy ilyen remultiplexert volt szerencsém megvizsgálni. Ez egy kínai gyártmányú készülék, azaz ára elfogadható, minősége ellenben nem olyan kiválló, mint egy nyugati remultiplexernek. A követelmények teljesítését ezen tökéletesen meg lehet vizsgálni.

Első lépésként össze kellett állítanom a saját programcsomagom, amihez az általam sokat vizsgált Astra 1 műholdcsoport egyik csatornájának jelét és a Rohde&Schwartz mérőgenerátorát használtam fel. A programcsomagom így nézett ki:

Műholdról vett programjaim:	CNNradio
	CNN
	Cartoon Network
	Travel
	TCM
Rohde&Schwartzról:	Bounce.gts
	HSweep1.gts
	CCIR17.gts

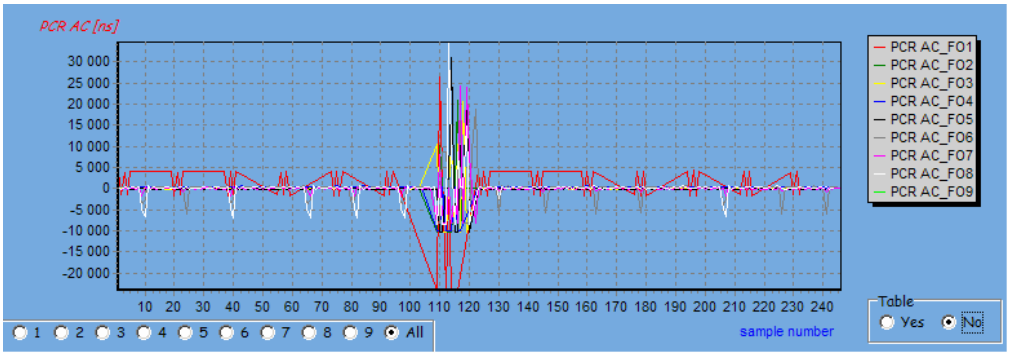
A műholdról vett és a Rohde által nyújtott TS jeleket a remultiplexer ASI bemenetére adtam, ahol kiválasztottam a remultiplexálni kívánt programokat, megadtam a TS_ID értékét, beállítottam a kimeneti adatsebességet, a formátumot (188 vagy 204) és az újonnan elkészült programjaim PID értékeit. Ezáltal a remultiplexerem ASI kimenetén már meg is

jelent az elkészített programcsomag. Ezt először egy QAM modulátorra vezettem, amely kimeneti jelét egy, a szabadteret modellező, csillapítón keresztül egy DVB-T-s set-top-boxra vezettem és megnéztem a TV-n látható képminőséget. Az eredmény megnyugtató volt, hiszen tiszta, hibamentes képet kaptam vissza. Tesztelés során néha, órás nagyságrendre tehető időközönként a képet egy pillanatra elvesztette a vevőkészülék, ami a képernyőn alig látható képkimaradást okozott. Ezen tapasztalat után kíváncsian vártam a mérésem eredményeit (Mentések\Remultiplexer\PCR \hiba31.pcr).



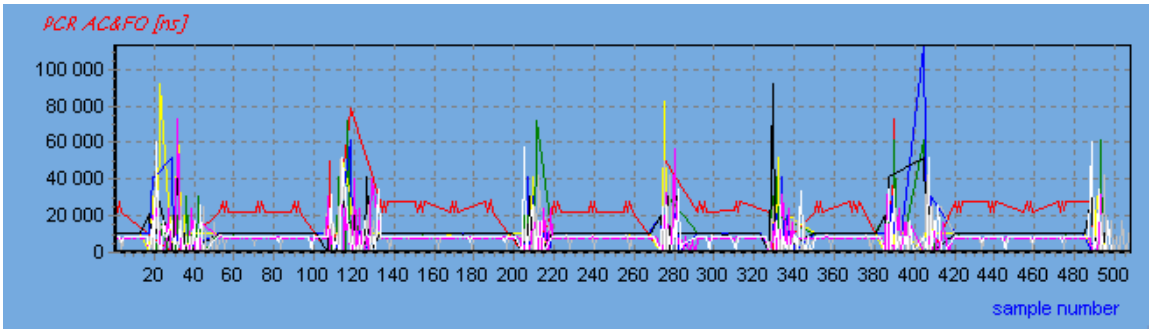
62. ábra A remultiplexer által okozott PCR hiba

Egy remultiplexer nem tekinthető elsődleges programcsomag-készítőnek, hiszen már meglévő programcsomagokat szed szét, és ezekből kiszedve egy vagy több programot készíti el a saját programcsomagját. Így a kimenetén már nem mérhető a PCR_AC, hanem csak a PCR_OJ, azaz a PCR_AC és a hálózati úton (remultiplexálás) keletkező hiba összege. Persze nem csak ez a hiba van jelen, a mérési módszerem magába tartalmazza a PCR_FO-t is. Ezt kiszűrni csak úgy tudjuk, ha a kapott eredményből kivonjuk a hibák átlagát.



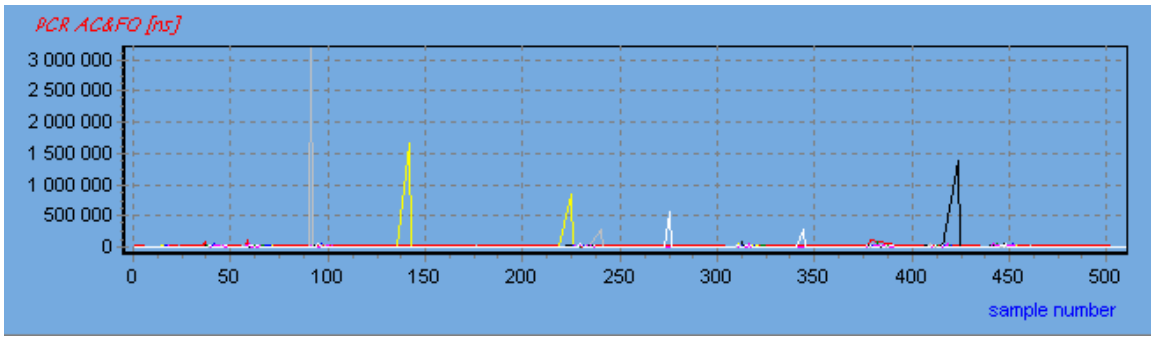
63. ábra Az előző mérés kiátlagolva

A PCR beültetés átlagosan jónak mondható, hiszen ha az ábrát kinagyítjuk (a programmal csak be kell töltenünk a …\Mentések\Remultiplexer\PCR\hiba31.pcr-t, meg kell nyitnunk az „other PCR operation” ablakot, ahol megkapjuk a 63. ábrát. Ezt kinagyítva láthatjuk, hogy a PCR beültetés abszolútértéke 500ns alatt van. Ez audiocsatorna esetén nem igaz, annál 2us környéki. De az audiocsatorna ilyen szempontból érzéketlenebb. Egyes programoknál (jelen esetben a Rohde programjainál) rendszeresen bekerül egy negatív maximum. Ennek az az oka, hogy a Rohde egy rövid műsort ismétel rendszeresen, így az ismétlés határán nagyobb PCR hiba keletkezik (műsorváltási hiba). Még ez sem lenne gond, hiszen a szabvány a 62. ábrán látott karakterisztika abszolút értékeire adja meg a 25us-os maximálisan megengedhető értéket. Ezt csak a rádió-műsor PCR hibája lépi túl rendszeresen, de erre nem oly kötöttek az előírások. Gondot okoznak viszont a remultiplexálás által bevitt zavarjelek, amiből egyet bemutat a 62. ábra, és egy csoportot a 64. ábra (Mentések\Remultiplexer\PCR \500gas.pcr).



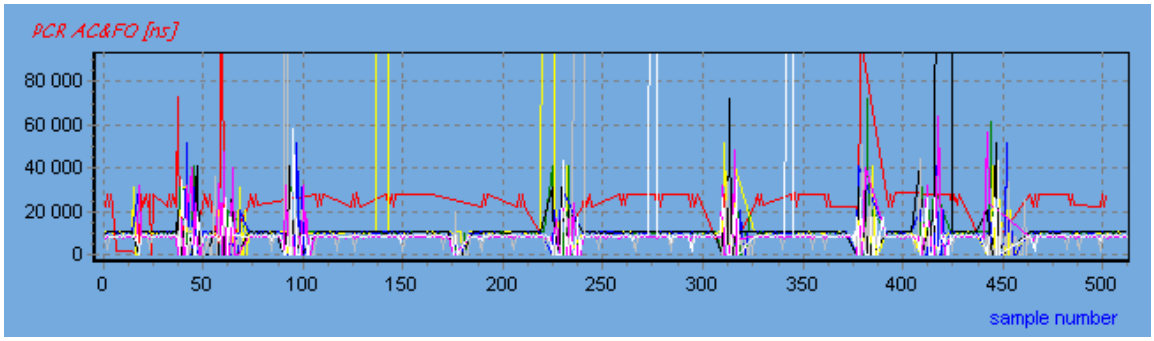
64. ábra A remultiblexer által bevitt hibajelek

Ezek a hibák már túllépik a 25us-os, a hálózaton keletkező jitter értékére megadott maximális értéket. Azaz ez már nem teljesíti a szabványban leírt követelményt. Viszont méréseim azt igazolták, hogy a digitális MPEG jelet még gond nélkül tudja kezelni a vevőkészülék még ekkora hibatartalomnál is.



65. ábra Műsorváltás jelleggű hibák

(Mentések\Remultiplexer\PCR \500rossz.pcr). Ezen az ábrán műsorváltás jellegű hibát láthatunk, egy-egy PCR ugrás figyelhető meg a 90., 140., 225., 270., 340., 450. mérési pontok körül. Ezek a hibák nagy gondot nem okoznak az órajel kinyerésében, hiszen egy-egy ekkora hiba nem tudja 38ms-nyi idő alatt annyira elhangolni az órajelet 27MHz-ről, hogy azt a következő, jó érték, ne tudná visszairányítani. Nagyítsuk ki az alsó régiót:



66. ábra A 65. ábra nagyítva

Ahogy az várható volt, a remultiplexer hibázásai megmaradtak, viszont okai közt nem említhetjük meg a műsorváltozást, hiszen a műsorváltozás során keletkező nagyobb hibaértéket nem kíséri minden esetben teljes összeomlási jelenség. A hiba okára más megoldást kell találnunk. Ez az ok lehet hardveres, illetve (ami valószínűbb) szoftveres hiba. Ez az összeomlási jelenség hamar lezajlik, utána minden rendesen folytatódik, de a jelenség kihat az összes remultiplexált csomagra.

Méréseim azt igazolták, hogy ez a készülék még nem tökéletes, de digitális MPEG videojelek átvitelére már alkalmas. Méréseket nem végeztem arra vonatkozóan, hogy a PAL jelekre miként viselkedik, de mivel a PAL jel érzékeny a színsegédvivő frekvenciájának pontos regenerálására, nem biztos, hogy ennek dekódolása teljesen tökéletes lesz egy ilyen remultiplexeren történő áthaladás során.

9 Összegzés

Most jön a szakdolgozatírók második legnehezebb feladata. Egy ilyen munkát elkezdeni nagyon nehéz, az elejét én is többször átírtam, de befejezni talán még nehezebb. Ezért nem szeretném az időt húzni, tekintsük át gyorsan, hogy mit is tanulhattam meg ezalatt a három hónap alatt, és mit sikerült átadnom eme szakdolgozat által ennek olvasói számára.

Röviden áttekintettem a digitális televíziótechnika előzményeit, az analóg televíziótechnika történetét. Biztos több oldalt is megérdemelt volna, de mivel ebben nőttünk fel, így egy-egy gondolat egész történeteket juttat mindenki emlékezetébe. Én ezért csak ezeket a gondolatokat szerettem volna felélesztени egy-egy főbb momentum megemlítésével.

Ezután következett a diplomatémám fő része, ahol először megpróbáltam egy egyszerű TS készítéséhez szükséges táblákat bemutatni, és a programcsomag-készítéshez szükséges táblák részletes magyarázásával megtudtuk, hogy hogyan is kell összerakni egy TS-t. Ehhez rengeteg szabványt kellett átböngésznem, ahol meghatározták számunkra a szükséges szintaktikát. Érdekes volt látni, hogy pillanatnyilag csak néhány táblát használnak, pedig a szabvány sokkal többet meghatározott. Eközben megismerkedtem a Delphi 7 nevű programmal, mellyel elkészítettem egy olyan programot, amellyel visszafejthetjük az összerakott TS összetartozó PID-jeit. Bármely set-top-box első és legfontosabb művelete, hiszen eme információ elengedhetetlen a dekódoláshoz. Persze nem ez volt témám lényegi része, de ugyanúgy, mint egy set-top-box számára, számomra is elengedhetetlen a jelfolyamban található PID-ek jelentése.

Miután a PID kigyűjtés megvalósult, már csak a PCR vizsgálata maradt hátra. Megismerkedtem az egyes mérési módszerekkel, és láttam a számítógépes mérés korlátait, megterveztem a saját mérési elvemet, melyet Delphi 7 alatt meg is írtam. A mérés megvalósításához elengedhetetlen volt a CableWorld Kft fejlesztői által kifejlesztett ethernet-átalakító.

Az elkészített programom segítségével méréseket végezhettem különböző műsorszórási rendszereken, ebben sok segítséget kaptam az ottani dolgozóktól.

Röviden talán így foglalnám össze eme 88 oldal tartalmát, hátterét. Remélem olvasása nem volt nagyon fárasztó, próbáltam egyszerű, érdekes, nem pedig szakmailag száraz művet létrehozni.

10 Függelék

Bájt	Bit	Megnevezés	Jellemzõk	Érték	Megjegyzés
1.	8	Szinkron bájt		h47	
2.	1 1 1 5	Transport error indicator Payload unit start indicator Transport priority PID felsõ 5 bitje	0-jó, 1-hiba van 1-ha PES vagy PSI kezdődik 0 -ha nem a kezdő bájt következik nem használják, mindig 0	0	
3.	8	PID alsó 8 bitje			
4.	2 2 4	Transport scrambling control Adaptation field control Continuity counter	00-not scrambled 01, 10, 11 user defined 01 payload only 10 adaptation field only 11 adaptation field + payload mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		null packet is 00 hasznos adat, feldolgozó mezõ vegyes elõzõ mező 00, 10 nem nő
Ez volt a packet header, eddig valamennyi packet felépítése azonos A következõ rész a packet payload. Ezen belül Program Specific Information (PSI) egyik táblájának, a Program Association Table (PAT) nevû táblának a felépítését láthatjuk					
5.	8	A section-ig lévõ bájtok száma, a payload kezdete	Section kezdetnél nem 0, a továbbiaknál 0.		
6.	8	Table Id	A PAT-nál ez mindig 00	h00	
7.	1 1 2 4	Section syntax indicator Mindig 0 Reserved Section lenght felsõ 4 bitje	Ez mindig 1 Ez mindig 0 Mindig 00xx értékû lehet csak	1 0 11 00xx	 max. 1021
8.	8	Section lenght alsó 8 bitje	A 12 bites hossz az innen (8+....) következõ bájtok száma CRC-vel		
9.	8	Transport stream id felsõ 8 bitje	**A TS "neve", ez a TS**		
10.	8	Transport stream id alsó 8 bitje	**legfontosabb azonosítója**		
11.	2 5 1	Reserved Version number Current next indicator	 Minden változás 1-gyel növeli 1-akkor aktuális, érvényes	xx 1	 0-elõzetes, próba
12.	8	Section number	00-tól kezdve az aktuális számú		
13.	8	Last section number	Ez az utolsó szekció száma		
14.	8	Program number 0 -bájt			
15.	8	Program number 0 -bájt			
16.	3 5	Reserved Network PID-0. A PID felsõ 5 bitje	 A NIT tábla, Network Information	 h00	
17.	8	Network PID-0. A PID alsó 8 bitje	Table PID-je (feleslegesen, ...16!)	16	
18.	8	Program number 1	Ez egy kát bájtos tetszõleges		
19.	8	Program number 1	azonosító		
20.	3 5	Reserved Program map PID -1	 Az elsõ program PMT táblájának		
21.	8	Program map PID -1	a PID-je van itt.		
		Ismétlõdõ strukturában folytatódik			

n-4	32	CRC	lezáró 4 bájt		

TS packet layer - CAT					
Bájt	**Bit**	**Megnevezés**	**Jellemzõk**	**Érték**	**Megjegyzés**
1.	8	Szinkron bájt		h47	
2.	1 1 1 5	Transport error indicator Payload unit start indicator Transport priority PID felsõ 5 bitje	0-jó, 1-hiba van 1-ha PES vagy PSI kezdõdik 0 -ha nem a kezdõ bájt következik nem használják, mindig 0	0	
3.	8	PID alsó 8 bitje			
4.	2 2 4	Transport scrambling control Adaptation field control Continuity counter	00-not scrambled 01, 10, 11 user defined 01 payload only 10 adaptation field only 11 adaptation field + payload mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		null packet is 00 hasznos adat, feldolgozó mezõ vegyes elõzõ mezõ 00, 10 nem nõ
Ez volt a packet header, eddig valamennyi packet felépítése azonos A következõ rész a packet payload. Ezen belül Program Specific Information (PSI) egyik táblájának, a Conditional Acces Table (CAT) nevû táblának a felépítését láthatjuk					
5.	8	A section-ig lévõ bájtok száma, a payload kezdete	Section kezdetnél nem 0, a továbbiaknál 0.		
6.	8	Table Id	A CAT-nál ez mindig 01	h01	
7.	1 1 2 4	Section syntax indicator Mindig 0 Reserved Section lenght felsõ 4 bitje	Ez mindig 1 Ez mindig 0 Mindig 00xx értékû lehet csak	1 0 xx 00xx	 max. 1021
8.	8	Section lenght alsó 8 bitje	A 12 bites hossz az innen (8+....) következõ bájtok száma CRC-vel		
9.	8	Reserved		xx	
10.	8	Reserved		xx	
11.	2 5 1	Reserved Version number Current next indicator	 Változás 1-gyel növeli modulo 32 1-akkor aktuális, érvényes	xx 1	 0-elõzetes, próba
12.	8	Section number	00-tól kezdve az aktuális számú		
13.	8	Last section number	Ez az utolsó szekció száma		
14.	8	Descriptor tag			
15.	8	Descriptor lenght			
16.	8	CA system id			
17.	8	CA system id			
18.	3 5	 CA PID			
19.	8	CA PID			
20.	8				
21.	8				
n-4	32	CRC	Lezáró 4 bájt		

TS packet layer - PMT					
Bájt	**Bit**	**Megnevezés**	**Jellemzõk**	**Érték**	**Megjegyzés**
1.	8	Szinkron bájt		h47	
2.	1 1 1 5	Transport error indicator Payload unit start indicator Transport priority PID felsõ 5 bitje	0-jó, 1-hiba van 1-ha PES vagy PSI kezdõdik 0 -ha nem a kezdõ bájt következik nem használják, mindig 0	0	
3.	8	PID alsó 8 bitje			
4.	2 2 4	Transport scrambling control Adaptation field control Continuity counter	00-not scrambled 01, 10, 11 user defined 01 payload only 10 adaptation field only 11 adaptation field + payload mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		null packet is 00 hasznos adat, feldolgozó mezõ vegyes elõzõ mezõ 00, 10 nem nõ
Ez volt a packet header, eddig valamennyi packet felépítése azonos A következõ rész a packet payload. Ezen belül Program Specific Information (PSI) egyik táblájának, a Program Map Table (PMT) nevû táblának a felépítését láthatjuk					
5.	8	A section-ig lévõ bájtok száma, a payload kezdete	Section kezdetnél nem 0, a továbbiaknál 0.		
6.	8	Table Id	A PMT-nél ez mindig 02	h02	
7.	1 1 2 4	Section syntax indicator Mindig 0 Reserved Section lenght felsõ 4 bitje	Ez mindig 1 Ez mindig 0 Mindig 00xx értékû lehet csak	1 0 xx 00xx	 max. 1021
8.	8	Section lenght alsó 8 bitje	A 12 bites hossz az innen (8+....) következõ bájtok száma CRC-vel		
9.	8	Program number felsõ 8 bitje	**A program azonosító száma, a**		max. 1016
10.	8	Program number alsó 8 bitje	**Progr. legfontosabb azonosítója**		
11.	2 5 1	Reserved Version number Current next indicator	 Minden változás 1-gyel növeli 1-akkor aktuális, érvényes	xx 1	 0-elõzetes, próba
12.	8	Section number	Mindig 00	h00	
13.	8	Last section number	Mindig 00	h00	
14.	3 5	Reserved PRC PID. A PID felsõ 5 bitje			
15.	8	PRC PID. A PID alsó 8 bitje			
16.	4 4	Reserved Program info lenght	 innen kezdve hány bájt a szöveg,	 00xx	 elsõ két bit 00
17.	8	Program info lenght	00+10 biten kifejezve		
18.	8	Stream type	Lásd a 2.36 tábl szerinti kiosztást		
19.	3 5	Reserved Elementary PID			
20.	8	Elementary PID			
21.	4 4	Reserved ES info lenght	 00xx értékû lehet csak		
22.	8	ES info lenght			
		Ismétlõdõ strukturában folytatódik			
n-4	32	CRC	Lezáró 4 bájt		

TS packet layer - Private section					
Bájt	**Bit**	**Megnevezés**	**Jellemzõk**	**Érték**	**Megjegyzés**
1.	8	Szinkron bájt		h47	
2.	1 1 1 5	Transport error indicator Payload unit start indicator Transport priority PID felsõ 5 bitje	0-jó, 1-hiba van 1-ha PES vagy PSI kezdõdik 0 -ha nem a kezdõ bájt következik nem használják, mindig 0	0	
3.	8	PID alsó 8 bitje			
4.	2 2 4	Transport scrambling control Adaptation field control Continuity counter	00-not scrambled 01, 10, 11 user defined 01 payload only 10 adaptation field only 11 adaptation field + payload mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		null packet is 00 hasznos adat, feldolgozó mezõ vegyes elõzõ mezõ 00, 10 nem nõ
Ez volt a packet header, eddig valamennyi packet felépítése azonos **A következõ rész a packet payload. Ezen belül Program Specific Information (PSI) egyik táblájának, a Private section nevû táblának a felépítését láthatjuk**					
5.	8	A section-ig lévõ bájtok száma, a payload kezdete	Section kezdetnél nem 0, a továbbiaknál 0.		
6.	8	Table Id	Néhány foglalt.	h40 - hFE	Table2-27
7.	1 1 2 4	Section syntax indicator Private indicator Reserved Private section lenght felsõ 4 bitje	Ez mindig 1 Mindig 00xx értékû lehet csak	1 xx 00xx	 max. 1021
8.	8	Private section lenght alsó 8 bitje	A 12 bites hossz az innen (8+....) következõ bájtok száma CRC-vel		
9.	8	Table id extension			
10.	8	Table id extension			
11.	2 5 1	reserved version number current next indicator	 számlál, ha van private section 0, version.no. nem hiteles		
12.	8	section number	h00 elsőnél, utána 1-256		
13.	8	last section number			
14.	8	private data	ism. struktúra		
n-4.	32	CRC 32			
15.					
16.					
17.					
18.					
19.					
n-4	32	CRC	lezáró 4 bájt		

TS packet layer - Adaptation field					
Bájt	**Bit**	**Megnevezés**	**Jellemzõk**	**Érték**	**Megjegyzés**
1.	8	Szinkron bájt		h47	
2.	1 1 1 5	Transport error indicator Payload unit start indicator Transport priority PID felsõ 5 bitje	0-jó, 1-hiba van 1-ha PES vagy PSI kezdõdik 0 -ha nem a kezdõ bájt következik nem használják, mindig 0	0	 21.old.
3.	8	PID alsó 8 bitje			PID Table 2-4.
4.	2 2 4	Transport scrambling control Adaptation field control Continuity counter	00-not scrambled 01, 10, 11 user defined 01 payload only 10 adaptation field only 11 adaptation field + payload mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		null packet is 00 hasznos adat, feldolgozó mezõ vegyes elõzõ mezõ 00, 10 nem nõ
Ez volt a packet header, eddig valamennyi packet felépítése azonos **A következõ rész a packet payload. Ezen belül az adaptation field felépítését láthatjuk**					
5.	8	Adaptation field length	feldolg. mezõ ? byte		
6.	1 1 1 1 1 1 1 1	Discontinuity indicator Random access indicator Elementary stream priority indicator PCR flag OPCR flag Splicing point flag Transport private data flag Adaptation field extension	1, ha csak AF van 1, akkor köv. PID csom. 1. Byte =tartalom info '1' video, '0' más adat '1' prog. óra ref. érték található '1' original prog. ref. érték van '1' splice countdown található '1' 1 vagy több bizalmas adatbyte '1' adatmezõ kiterjesztés		Folytonosság állj!
7.	8	Program clock reference base	Elsõ byte, összesen 33 bitnyi.		Ha PCR flag=1
8.	8	Program clock reference base	Következõ 8		
9.	8	Program clock reference base			
10.	8	Program clock reference base			
11.	1 6 1	Program clock reference base reserved Program clock reference extension	 A programok referencia órájának		
12.	8	Program clock reference extension	kiterjesztése		
13.	8	Original program clock reference base	A „fõ" referenciaórajel		Ha OPCR flag=1
14.	8	Original program clock reference base			Ha PCR flag=1
15.	8	Original program clock reference base	Következõ 8		
16.	8	Original program clock reference base			
17.	1 6	Original program clock reference base Reserved Original program clock reference			

	1	extension			
18.	8	Original program clock reference extension			
19.	8	Splice countdown	Hátralévõ TS csomag száma		Pozitív érték
20.	8	Transport private data length	? byte bizalmas adat		Ha t.p.d.f=1
21…		Private data byte	Nem specifikus		
22.	8	Adaptation field extension length	Kiterjesztett feldolgozómezõ hossza		
23.	1 1 1 5	Ltw flag Piecewise rate flag Seamless splice flag reserved	Ltw offset van-e Összeillesztési idö info		
24.	1 7	Ltw valid flag Ltw offset	Az offset valós-e Legal time window offset		Ltw flag=1 Ltw flag=1
25.	8	Ltw offset			
26.	2 6	Reserved Piecewise rate			Ha p.r.flag=1
27.	8	Piecewise rate			
28.	8	Piecewise rate			
29.	4 3 1	Splice tyme DTS next AU[32..30] Marker bit	Összeillesztési idõ és sebesség Dekódolási unitegység a		s.s.f.=1 s.s.f.=1
30.	8	DTS next AU[29..22]	következõ access unit-ig		
31.	7 1	DTS next AU[21..15] Marker bit			
32.	8	DTS next AU[14..7]			
33.	7 1	DTS next AU[6..0] Marker bit			
34.	8	Reserved			
35.	8	Stuffing byte	lezáró byte ’1111 1111’	hFF	
36.	8	Data byte			

TS packet layer - PES packet					
Bájt	**Bit**	**Megnevezés**	**Jellemzõk**	**Érték**	**Megjegyzés**
1.	8	Szinkron bájt		h47	
2.	1 1 1 5	Transport error indicator Payload unit start indicator Transport priority PID felsõ 5 bitje	0-jó, 1-hiba van 1-ha PES vagy PSI kezdõdik 0 -ha nem a kezdõ bájt következik nem használják, mindig 0	0	 21.old.
3.	8	PID alsó 8 bitje			PID Table 2-4.
4.	2 2 4	Transport scrambling control Adaptation field control Continuity counter	00-not scrambled 01, 10, 11 user defined 01 payload only 10 adaptation field only 11 adaptation field + payload mindig 1-gyel nõ 0000-1111 (azonos PID-re !)		null packet is 00 hasznos adat, feldolgozó mezõ vegyes elõzõ mezõ 00, 10 nem nõ
Ez volt a packet header, eddig valamennyi packet felépítése azonos **A következõ rész a packet payload. Ezen belül a Prpgram Elementary Stream felépítését látjuk**					
5.	8	packet start code prefix	'0000 0000'	h00	kezdetjelzõ
6.	8	packet start code prefix	'0000 0000'	h00	
7.	8	packet start code prefix	'0000 0001'	h01	
8.	8	stream id	jelfolyamazonosító		Table 2-19, 33o.
9.	8	PES packet length	**PES packet-ünk hossza**		
10.	8	PES packet length			
11.	2 2 1 1 1 1	'10' PES scrambling control PES priority data alignment indicator copyright original or copy	 00 nincs bitkeverés, többi felhaszn. '1', akkor van 'data stream alignment descriptor' '1', akkor 'copyright descriptor' PES eredeti vagy csak másolat	10	 nem használjuk Table 2-47, 67o. Table 2-57, 72o.
12.	2 1 1 1 1 1 1	PTS DTS flags ESCR flag ES rate flag DSM trick mode flag additional copy info flag PES CRC flag PES extension flag	PES-ben '10'-PTS, '11'-PTS&DTS, '00'-DTS, '01'-tiltott '1' alap és kibőv. ESCR '1'-"ES rate field" van trick mode be-, kikapcsolása Van-e CRC ellenőrzés? Van-e PES kibővítés?		
13.	8	PES header data length	a teljes opcionális mező hossza		
14.	4 3 1	'0010' PTS[32..30] marker	 presentation time stamp '1'	h2 1	Ha PTS DTS flags='10'

15.	8	PTS[29..22]			
16.	7 1	PTS[21..15] marker	 '1'	 1	
17.	8	PTS[14..7]			
18.	7 1	PTS[6..0] marker	 '1'	 1	
14.	4 3 1	'0011' PTS[32..30] marker	 presentation time stamp '1'	h3 1	Ha PTS DTS flags='11'
15.	8	PTS[29..22]	PTS(k)=((system clock frequency		
16.	7 1	PTS[21..15] marker	*tpn(k))DIV 300)modulus2e33 '1'	 1	
17.	8	PTS[14..7]			
18.	7 1	PTS[6..0] marker	 '1'	 1	
19.	4 3 1	'0001' DTS[32..30] marker	 decoding time stamp '1'	h1 1	
20.	8	DTS[29..22]	DTS(j)=((system clock frequency		
21.	7 1	DTS[21..15] marker	*tdn(j))DIV300)modulus 2e33 '1'	 1	
22.	8	DTS[14..7]			
23.	7 1	DTS[6..0] marker	 '1'	 1	
24.	2 3 1 2	reserved ESCR base[32..30] marker ESCR base[29..28]	 elementary stream clock reference '1' alapjelfolyam referencia órajele	 1	ESCR flag=1
25.	8	ESCR base[27..20]			
26.	5 1 2	ESCR base[19..15] marker ESCR base[14..13]	 '1'	 1	
27.	8	ESCR base[12..5]			
28.	5 1 2	ESCR base[4..0] marker ESCR extension	 '1' ESCR kiterjesztése	 1	
29.	7 1	ESCR extension marker bit	 '1'	 1	
30.	1 7	marker bit ES rate	'1' definiálja a PES jelfolyam byte-	1	ES rate flag=1
31.	8	ES rate	jainak beérkezési sebességét		
32.	7 1	ES rate marker bit	a P_STD kimenetén '1'	 1	
33.	3 2 1 2	trick mode controll field id intra slice refresh frequency truncation	videojel kiolvasási sebességgére ad felvilágosítást, 000-gyors továbbküldés, 001-lassú, 010-fagyott keret, 011-gyors átkapcsolás, 100-lassúátkapcs. kijelzett részre ad felvilágosítást '1' 00-csak DC együttható nem 0, 01-első 3 nem 0, 10-első 6 nem 0, 11-egyik együttható sem 0	 1	trick mode flag=1 t.m.controll =000

33.	5	rep cntrl	? ideig tartsa az adott képet		t.m.controll =001
33.	2 3	field id reserved			t.m.controll =010
33.	2 1 2	field id intra slice refresh frequency truncation	'1'	1	t.m.controll =011
33.	5	rep cntrl			t.m.controll =100
33.	5	reserved			else
34.	1 7	marker bit additional copy info	'1' további másolási info, privát		additional copy info=1
35.	8	previous PES packet CRC			PES CRCflag=1
36.	8	previous PES packet CRC			
37.	1 1 1 1 3 1	PES private data flag pack header field flag program packet sequence counter flag P-STD buffer flag reserved PES extension flag 2	privát adat? PrStream fejrész van-e? P-STD buffer kijelzés kapcsolása PES kiterjesztett vagy összekapcsolt mező van-e		PES extension flag=1
38.	128	PES private data			PESp.d.f.=1
39.	8	pack field length	byte-ban adja meg a hosszt		pack.h.f.f.= 1
40.	1 7	marker bit program packet sequence counter	'1' hasonló, continuity counter	1	p.p.s.c.f.=1
41.	1 1 6	marker bit MPEG1 MPEG2 identifier original stuff length	'1' 1system stream, 0program stream kitöltő byte-ok a csomagfejben		
42.	2 1 5	'01' P-STD buffer scale P-STD buffer size	 audio0-128, video1-1024 BSn=P-STD buffer size*128 or	01	P-STDb.f.=1
43.	8	P-STD buffer size	1024		
44.	1 7	marker bit PES extension field length	'1' mennyi adatbit következik még	1	PESe.f.2=1
45.	8	reserved			
46.	8	stuffing byte	'1111 1111'	hFF	
47.		PES packet data byte			
48.	8	padding byte			

A DSM-CC vezérlési táblája

5.	8	packet start code prefix		h00	
6.	8	packet start code prefix		h00	
7.	8	packet start code prefix	állandó érték	h01	
8.	8	stream id	jelfolyamazonosító	hF2	
9.	8	packet length	a csomag hossza 16 biten		
10.	8	packet length			
11.	8	command id	h00- tiltott h01-felügyelés h02-nyugta h3-hFF-foglalt		a bitfolyam milyenségéről ad felvilágosítást
12.	1 1 1 5	select flag retrieval flag storage flag reserved	a bitfolyam szűrését aktivizálja újraküldést vezérli tárolás műveletének aktívvá tétele "00000"	 00000	HA h01 a command id
13.	7 1	reserved marker bit	"0000000" kezdő kód keresés kikapcs.	h01	 mindig 1
14.	8	bitstream id[31-24]	ez egy 32 bites azonosító		select flag="1"
15.	7 1	bitstream id[23-17] reserved	mely a jelfolyamot címzi meg		
16.	8	bitstream id[16-9]	egyedülálló módon, egy saját		
17.	7 1	bitstream id[8-2] reserved	hozzárendelési tábla segítségével		
18.	2 5 1	bitstream id[1-0] select mode marker bit	 0-tiltott, 1-tárolási parancs jön, 2-újraküldési, 3-31foglalt "1"	 "1"	
19.	1 1 1 1 1 3	jump flag play flag pause mode resume mode stop mode reserved	az új hozzáférési egységet jelzi a bitek küldése egy adott időtől az ismétlés szüneteltetése folytatása a bitek küldésének megállítása foglalt "000"		retrieve flag="1"
20.	7 1	reserved marker bit	"0000000" "1"		
21.	7 1	reserved direction indicator	"0000000" irányjelző (utána time_code)		jump flag="1"
	1 1 6	speed mode direction indicator reserved	sebesség skálát aktivizálja irányjelző (utána time_code) "000000"		play flag="1"
	6 1 1	reserved record flag stop mode	"000000" speciális kérés a jelfolyam felé a felvétel leállítása		storage flag="1"
n-4		time code			record flag="1"
	1 1 1 1 4	select ack retrieval ack storage ack error ack reserved	a select parancsra válaszol a kiválasztott jelfolyam ismétlésre felkészült tárolásra felkészült nem értette a kérést foglalt "0000"		HA command id = h02

	6 1 1	reserved marker bit cmd status	foglalt "000000" "1" ha ez is "1", akkor érvényes a válasza		
		time code	ha cmd status AND (retrieval OR storage ask)="1"		

time code struktúra

	7 1	reserved infinite time flag	"0000000" végtelenített idő, azaz a megadott művelet nem megfelelő		HA command id = h02
	4 3 1	reserved PTS[32-30] marker	"0000" az aktuális csomag azon idejét "1"		
	8	PTS[29-22]	tartalmazza, ahonnan az adott		
	7 1	PTS[21-15] marker	operációt el kell végezni "1"		
	8	PTS[14-7]			
	7 1	PTS[6-0] marker			

11 Irodalomjegyzék

[1] ISO/IEC 13818-1 (ITU-T Recommendation H.222.0): "Information technology - Generic coding of moving pictures and associated audio information: Systems".

[2] ISO/IEC 13818-9: "Information technology - Generic coding of moving pictures and associated audio information - Part 9: Extension for real time interface for systems decoders".

[3] ETSI EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[4] ETSI ETR 290: "Digital Video Broadcasting (DVB); Measurement guidelines for DVB systems".

[5] ETSI TR 101 290: "Digital Video Broadcasting (DVB); Measurement guidelines for DVB systems".

[6] ETSI ETR 211: "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

[7] Dr. Ferenczy Pál: Video- és hangrendszerek,
Műszaki Könyvkiadó, Budapest, 1986.

[8] Dr Ferenczy Pál: Hírközléselmélet,
Műszaki Könyvkiadó, Budapest 1974.

[9] Standeisky István: A digitális tv-műsorszórás alapjai
Széchenyi István Egyetem, Győr, 2003

[10]
http://www.tek.com/Measurement/App_Notes/25_14706/eng/25W_14706_0.pdf

[11]
http://www.tek.com/Measurement/App_Notes/25_14617/eng/25W_14617_1.pdf

[12] http://www.nhk.or.jp/strl/publica/bt/en/le0011.pdf

[13] http://www.rohde-schwarz.com/www/appnotes_files.nsf/ANFileByANNoForInternet/A93BBCCB563EAAD8C1256C6F001FC65A/$file/7BM42_0E.pdf

[14] http://www.ahrt.hu