ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA RADIOELEKTRONIKY

Výukové úlohy pro přípravek s číslicovým signálovým procesorem

2015 Diplomant: Petr Duga

Vedoucí práce: Prof.Ing. Pavel Zahradník

Poděkování

Rád bych tímto poděkoval všem, kteří mi při zpracovávání diplomové práce poskytovali cenné rady, zasvěcovali mě do tajů zpracovávané problematiky a byli mi duševní oporou. Děkuji především vedoucímu práce panu profesorovi Zahradníkovi za odborné vedení, cenné rady, připomínky a v neposlední řadě za velkou trpělivost, kterou se mnou měl. Mé díky také patří mým blízkým, rodičům a kamarádům.

Prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a používal jsem jen informační zdroje, které jsou uvedené v této práci

V Praze dne 5. ledna 2015 _____

Petr Duga

Anotace

Tato práce popisuje praktickou realizaci ukázkových výukových úloh z oblasti analýzy digitálního obrazu, radiotechniky a číslicového zpracování audiosignálu digitálním signálovým procesorem TMS320C6748 umístěným na vývojovém kitu LCDK.

Abstract

This diploma thesis describes a practical realization of educations tasks for demonstration of digital image analysis, radio technology and digital signal processing of audio signal by digital signal processor TMS320C6748 placed on evaluation board LCDK.

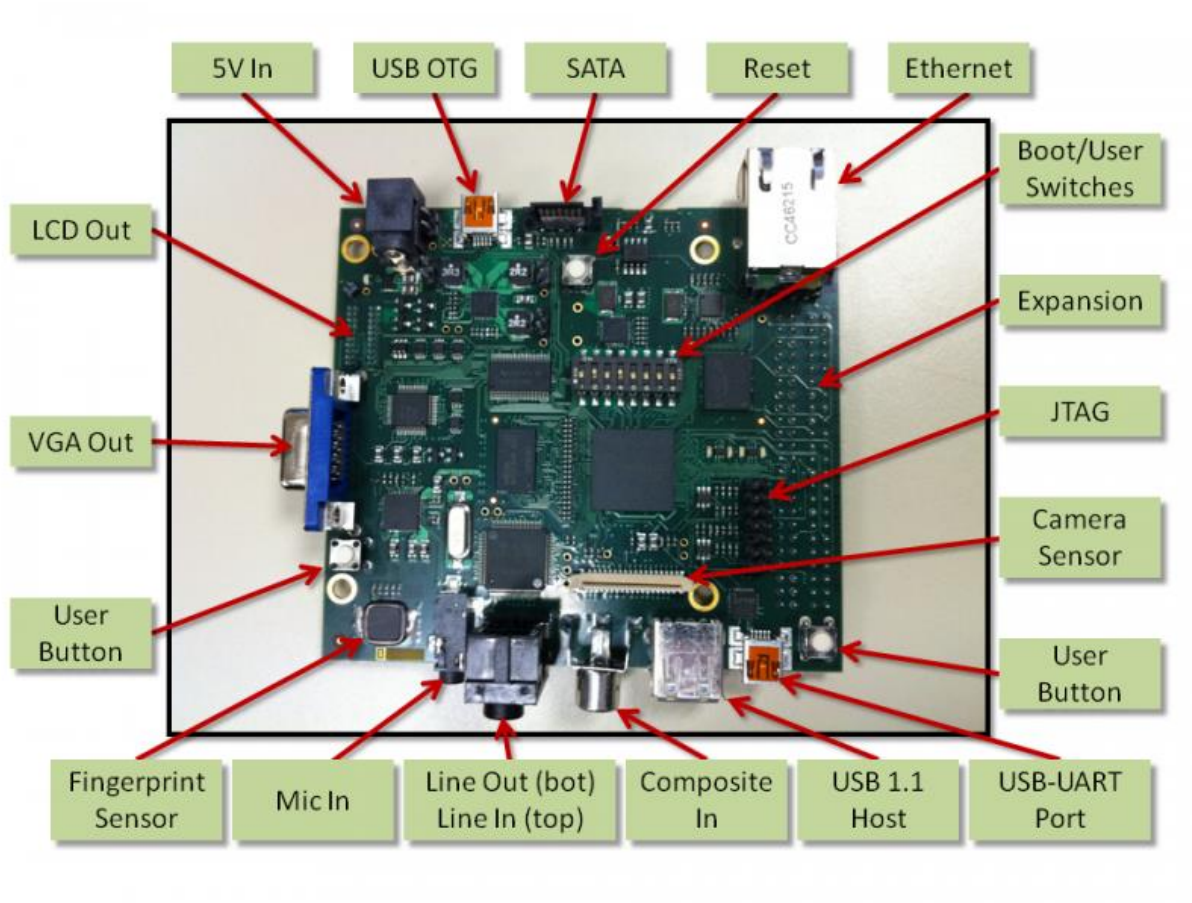
Obsah

1 Úvod

V této diplomové práci jsem se zabýval tvorbou výukových programů z oblasti zpracování audio a video signálů signálovým procesorem a vytvářením výukového přípravku s použitím vývojové desky TMS320C6748LCDK.

1. TMS320C6748 LCDK

Vývojový kit TMS320C6748LCDK obsahuje signálový procesor TMS320C6748 a další integrované obvody, tlačítka a DIP spínače, LED a konektory, které slouží k demonstraci použití TMS320C6748 pro záznam a zpracování audia, videa, pro síťovou komunikaci přes ethernetový řadič, načítání externích dat uložných na MMC/SD kartě, či na na datovém nosiči FLASH. Do hardwareové verze A6 zde byl také snímač otisků prstů, k němuž byl dodáván i demonstrační program. Vzhledem k problémům s komunikací a spolehlivostí se tento čip už neosazuje. Kit neobsahuje vestavěný emulátor, jen 14-pinový konektor pro připojení externího JTAGu. V této diplomové práci byl použit programový zavaděč JTAG XDS100v2, který kromě nahrávání programů do paměti umí také nahrávat/vyčítat do/z paměti, vyčítat hodnoty uložené v registrech a podporuje debugovací funkce vývojového prostředí Code Composer Studio (CCS) . Pro vývoj programů bylo použito vývojové prostředí Code Composer Studio 5 verze 3 (CCS5v3). Pro použití kitu do výuky vývoje aplikací na DSP hovoří zejména, jeho příznivá cena, minimální práce s hardwarem, knihovní podpora a převážně dobrá dokumentace. Proti hovoří malá bitová hloubka RGB kanálů na video výstupu a nemožnost využití všech vnitřních periferií C6748, protože některé jsou už využity v zapojení kitu.



Obr.1 Vývojový kit LCDK s procesorem C6748

1.1 Periferie LCDK

Periferie a rozhraní přítomné na LCDK shrnuje níže uvedený výčet a jejich zapojení znázorňuje diagram na obrázku 2. Pro bližší informace viz [3].

1.1.1 Rohraní

- 1x mini USB sériový port (rozhraní UART-USB)
- 1x rychlý ehternetový port (10/100) se signalizační LED
- 1x USB HOST port (USB 1.1)
- 1x USB mini OTG port (USB 2.0)
- 1x SATA port (3Gbps)
- 1x VGA port (15-pinový D-SUB)
- 1x LCD port
- 1x Vstup kompozitního videa (RCA konektor)
- 1x Vstup pro kamerový senzor od firmy Leopard Imaging (36-pinový zip konektor)
- 3x Audio port (Line in, Line out a Mic)
- 1x JTAG konektor (14- pinový)

1.1.2 Paměti

- 1x 128MB SDRAM DDR2
- 1x 128MB NAND FLASH
- 1x Micro SD/MMC slot

1.1.3 Video

- 1x 10-bitový DAC
- 1x 10-bitový dekodér digitálního videa

1.1.4 Audio

- 1x stereo audio kodek

1.1.5 Uživatelské rozhraní

- **Spínače**- Spínače mají v pozici zapnuto logickou úroveň LOW a v pozici zapnuto úroveň HIGH. SW1-4 slouží k nastavení bootovacího režimu kitu dle tabulky 1, SW 5-8 jsou uživatelské spínače, které jsou namapované na univerzální piny GPIO procesoru C6748 viz tabulka 2.

# spínače	UART2	NAND16	MMC/SD0
1	OFF	OFF	OFF
2	ON	ON	OFF
3	OFF	ON	OFF
4	ON	ON	ON
Tab.1 Bootovací režimy LCDK			

# spínače	Pin
5	GPIO0[1]
6	GPIO0[2]
7	GPIO0[3]
8	GPIO0[4]
Tab.2 Přiřazení spínačů GPIO pinům	

- **Tlačítka**- Tlačítka mají při stisku úroveň LOW, jinak mají úroveň HIGH

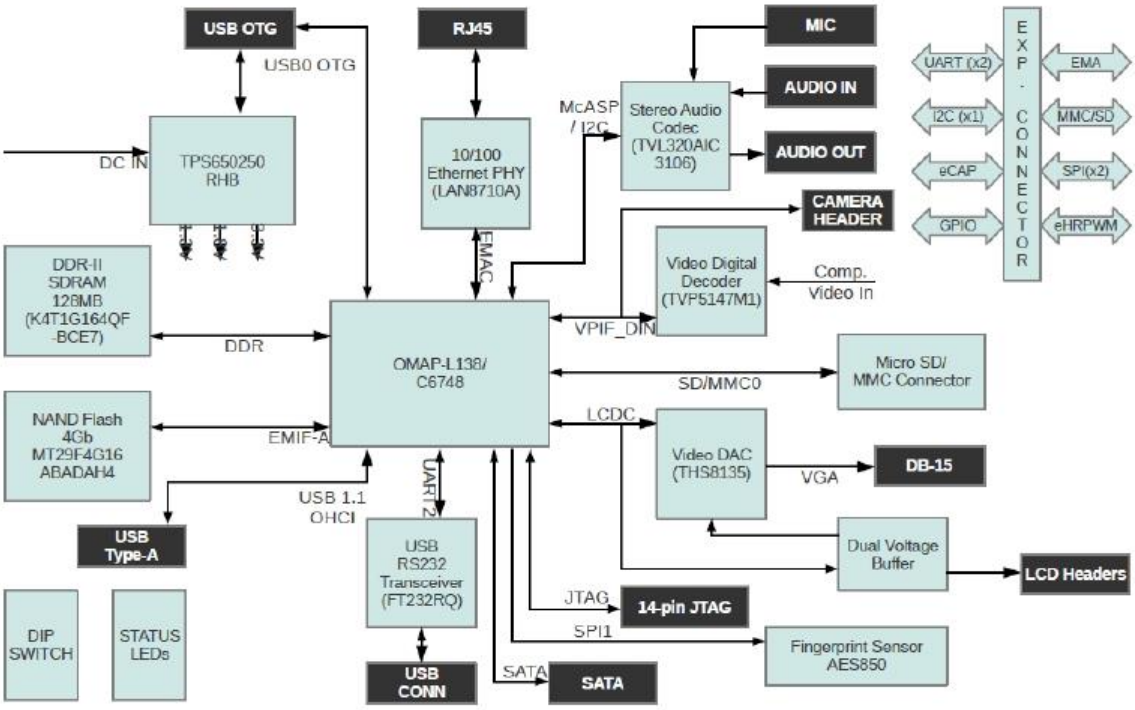
Tlačítko	Pin
S1	RESET
S2	GPIO2[4]
S3	GPIO2[5]
Tab.3 Přiřazení tlačítek GPIO pinům	

- **LED**- viz tabulka 4.

LED	Signál	Svítí když…
D1	5V_IN	je zavedené napájení do J1
D2	VOLT_ERR	je V_IN>5.8V
D3	VCC_5VD_IN	Napájení buď J1 nebo USB
D4	GPIO6[13]	Je signál v HIGH
D5	GPIO6[12]	Je signál v HIGH
D6	GPIO2[12]	Je signál v HIGH
D7	GPIO0[9]	Je signál v HIGH
Tab.4 LED na LCDK		

1.1.6 Napájení

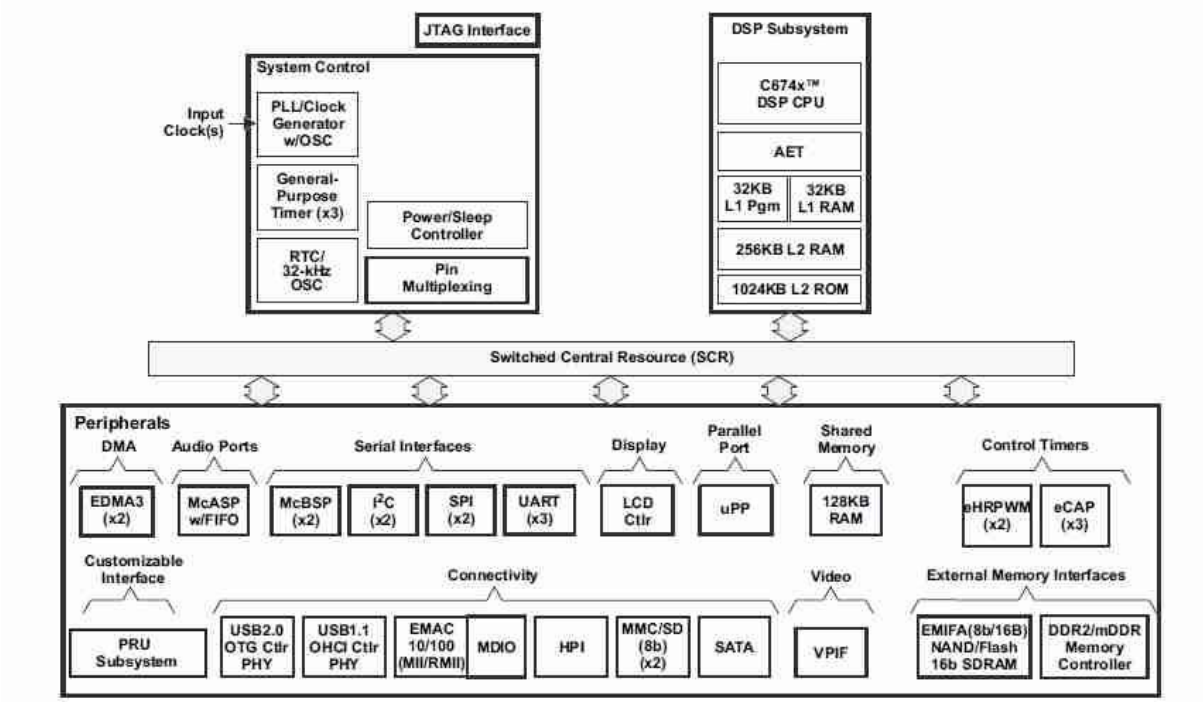
LCDK je možné napájet 5V buď přes konektor J1 nebo z USB pomocí mini-USB J2. Proudový odběr ovšem za normálních podmínek může přesáhnout 500mA, proto se nedoporučuje používat USB.



Obr.2 Blokový diagram zapojení periferií s DSP na LCDK

1.2 Procesor C6748

Procesor C6748 je 32 bitový signálový procesor s Harvardskou architekturou od firmy Texas Instruments. Implementuje architekturu VLIW a podporuje operace v plovoucí řádové čárce. Podporuje paralelní zpracovávání dat, SIMD, pipelining, dvouúrovňový systém vyrovnávacích pamětí pro programové i datové paměti, vnitřní přímý přístup do pamětí (IDMA), systém řízení zdrojů (BWM), přerušovací systém a systém správy napájení. Pro bližší informace o jádru procesoru a jeho pomocných systémech tzv. megamodulu viz [5], [1] a o CPU viz [4]. Také obsahuje zabudované řadiče pro správu periferií viz Obr.3 a [1] a obvody pro správu systemu DSP. Použité moduly pro periferie budou popsány v jednotlivých úlohách a jejich podrobný popis lze najít v referenčním manuálu [1]. Ve výchozím nastavení pracuje procesor s hodinovou frekvencí 300MHz, ale lze jej přetaktovat až na 456MHz.



Obr.3 Blokový diagram DSP

2 Cannyho hranová detekce

2.1 Teoretický popis

Hranová detekce je důležitou součástí analýzy 3D scény na základě jejího 2D obrazu. Hrany totiž mají větší vypovídací hodnotu o scéně než souvislé plochy, protože nesou informace o prostorových změnách scény, změnách v osvětlení scény (stíny, odrazy světla), změnách v odrazivosti scény (přechody barevných oblastí). Z tohoto důvodu nachází hranová detekce uplatnění v úlohách digitálního zpracování obrazu a v úlohách počítačového vidění. Pro význam hranové detekce v procesu zpracování obrazu viz [19] str. 7 obr.1.7. pro detailnější pohled viz [20] str. 14 obr. 1.14. V této úloze jsem implementoval hranovou detekci podle algoritmu Johna Cannyho, který je teoreticky podložen v [18]. O hranové detekci a digitálním zpracování obrazu jsem čerpal informace převážně ze zdrojů [19] a [20]. V dalším textu budu souhrně uvádět základní a stěžejní myšlenky a vztahy potřebné pro pochopení a popis této úlohy.

2.1.1 Popis digitálního obrazu

Digitální obraz je možno modelovat dvoudimenzionální obrazovou funkcí diskrétní jak v definičním oboru, tak v oboru hodnot, která pro daný pixel, určený souřadnicemi x a y, vrací vektor hodnot viz. rovnice 1.

$$f(x,y) = \vec{I} \tag{1.1}$$

Interpretace tohoto vektoru záleží na obrazové reprezentaci, na zvoleném barevném prostoru. V této úloze jsem pracoval s I[R,G,B] a I[Y,Cb,Cr]. Složkám vektoru I se říká kanály a rozsah jejich hodnot udává tzv. barevná hloubka. Z fyziologického hlediska je opodstatněné pracovat v úloze hranové detekce s šedotónovým obrazem, který vyjadřuje jasový kanál Y. Pro vyjádření šedotónového obrazu v RGB barevném prostoru se používá převodní vztah, který popisuje rovnice 1.2. Pro bližší informace viz [20] str. 2 a str.15.

$$Y = 0.299R + 0.587G + 0.114B \tag{1.2}$$

2.1.2 Matematický popis hran

Jak již bylo řečeno v úvodu, hrany představují prostorovou změnu obrazové funkce, která vypovídá o změně nasnímané scény. Matematicky lze změnu vyjádřit pomocí derivace, v případě 2D obrazové funkce pak pomocí gradientu této funkce v kartézské soustavě souřadnic viz vztah 1.3.

$$G_{xy} = \frac{\partial f(x,y)}{\partial x}\,\vec{\imath} + \frac{\partial f(x,y)}{\partial y}\,\vec{\jmath} \tag{1.3}$$

Kde Gxy je gradient obrazové funkce, i a j jsou jednotkové vektory určující směry v ose x, respektive y. Gradient je tedy, v případě šedotónového obrazu, vektor, který je orientován od černé barvy k bílé. Hrana je pak popsána svým normálovým vektorem, který je shodný s gradientem. Kromě složkového vyjádření rovnicí 1.3, které vypovídá o příspěvcích gradientu ve směru x a y, lze gradient obrazové funkce vyjádřit pomocí jeho modulu a úhlu. Přičemž modul vypovídá o strmosti hrany, což si můžeme představit jako velikost rozdílu jasové funkce na blízkém okolí zkoumaného bodu, a úhel o její orientaci v rámci obrazu. Viz vztahy 1.4 a 1.5.

$$\left|G_{xy}\right| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2} \tag{1.4}$$

$$ang\left(G_{xy}\right) = arg\left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right) \tag{1.5}$$

2.1.3 2D konvoluce

Jelikož se ale budeme zabývat digitálními obrazy, nelze vztahy 1.3-1.5 použít přímo. Máme dvě možnosti. Buď si obrazovou funkci v blízkém okolí daného pixelu s využitím hodnot sousedních pixelů aproximujeme spojitými funkcemi například Taylorovým rozvojem a použijeme pak přímo vztahy 1.4 a 1.5, nebo aproximujeme parciální derivace například diferencemi. Celý výpočet gradientu pak probíhá konvolucí obrazu tzv. konvolučním jádrem. Toto jádro právě aproximuje parciální derivace. Místo diferencí se potom používají jádra, která lépe vystihují změnu digitální obrazové funkce. 2D konvoluce představuje matematický přepis filtrace lineárním prostorově invariantním jádrem filtru typu FIR, přičemž toto jádro představuje jeho prostorovou impulzní odezvu. Vztah 1.6 udává přepis pro diskrétní 2D konvoluci.

$$g(i,j) = \sum_k \sum_l f(i-k, j-l) h\left(k + \frac{K-1}{2}, l + \frac{L-1}{2}\right) \tag{1.6}$$

$$k \in \langle -\frac{K-1}{2}, \frac{K-1}{2} \rangle, l \in \langle -\frac{L-1}{2}, \frac{L-1}{2} \rangle$$

Kde g(i,j) je pixel filtrovaného obrazu na pozici i,j, f(i,j) je výchozí digitální obrazová funkce, h (k,l)je konvoluční jádro o rozměrech KxL. Přičemž se většinou pracuje se čtvercovými jádry=> K=L, které jsou liché. V takovém případě má jádro střed v konkrétním pixelu.

V této úloze jsem pro aproximaci gradientu vyzkoušel postupně tyto jádra : Prewittovo a Sobelovo o velikosti 3x3. Dále je stručně uvedu pro aproximace parciální derivace ve směru x.

2.1.4 Prewittovo jádro

Základ jádra tvoří ve směru x diference viz (1.7) a ve směru y sumace viz (1.8).

$$\Delta f(i,j) = f(i-1,j) - f(i+1,j) \tag{1.7}$$

$$\sum f(i,j) = f(i,j-1) + af(i,j) + f(i,j+1) \tag{1.8}$$

Kde a je váhovací koeficient a v případě Prewittova jádra je roven jedné.

Prewittovo jádro se pak sestavuje dle rovnice 1.9

$$h_{xPrewitt}(3,3) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} (-1 \quad 0 \quad 1) = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{1.9}$$

2.1.5 Sobelovo jádro

Základ jádra tvoří ve směru x diference viz (1.7) a ve směru y váhovaná sumace viz (1.8) pro a=2.

Sobelovo jádro se pak sestavuje dle rovnice 1.10

$$h_{xSobel}(3,3) = \begin{pmatrix}1\\2\\1\end{pmatrix}(-1 \quad 0 \quad 1) = \begin{pmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{pmatrix} \tag{1.10}$$

2.1.6 DoG jádro

Jde o jádro tvořené derivací Gaussiánu ve směru x, které je převedeno na celočíselné hodnoty. Vliv hodnot „postranních“ pixelů určuje parametr sigma. Pro výpočet jader DoG pro různé vstupní parametry jsem napsal skript „Gauss.m“, který je součástí příloh této práce.

2.1.7 Cannyho přístup k detekci hran

John Canny uveřejnil roku 1986 v [18] postup pro optimální detekci 1D schodových hran s ohledem na 3 stanovená kritéria:

(1) Hranová detekce má detekovat hrany spolehlivě
(2) Hranová detekce má lokalizovat hrany co nejpřesněji
(3) Hranová detekce se má vyvarovat vícenásobné detekci hran

Tento postup byl později zobecněn pro případ 2D obrazové funkce.

Z [18] vyplývá postup pro hranovou detekci, která splňuje výše uvedená kritéria. Tento postup je shrnut v [19] str. 144. Zde si uvedeme jednotlivé kroky hranové detekce

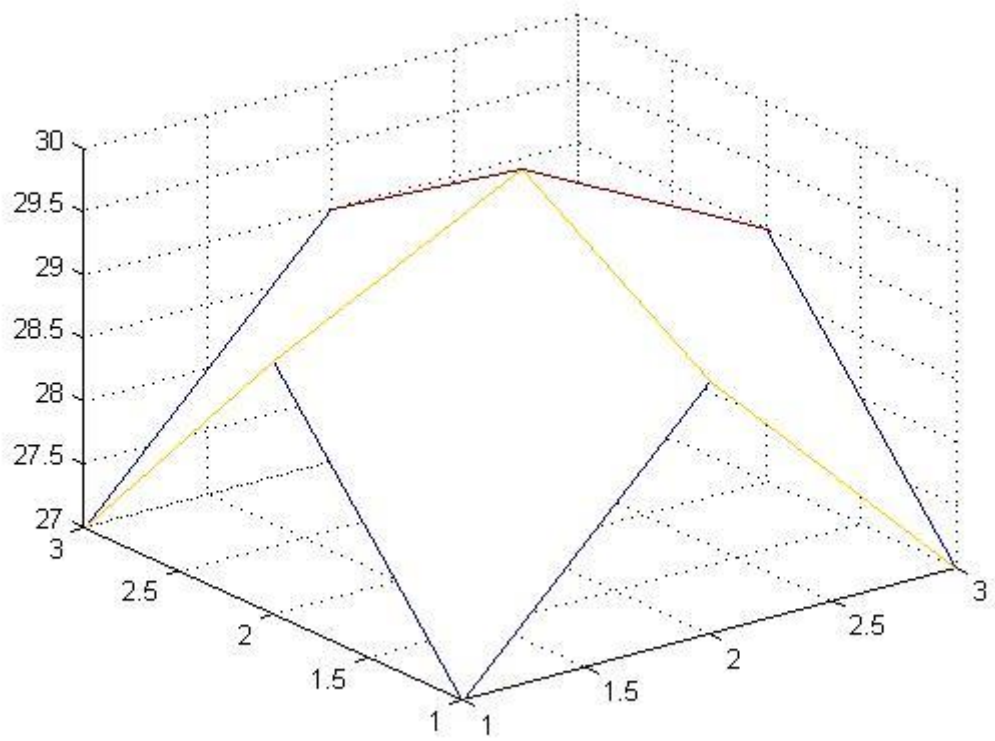
1. ***Filtrace obrazu Gaussiánem***

   V tomto kroku dochází k rozmazání obrazu Gaussiánem, což je filtr typu dolní propust, který je dán předpisem 1.11 a je určený parametrem střední kvadratické odchylky sigma.

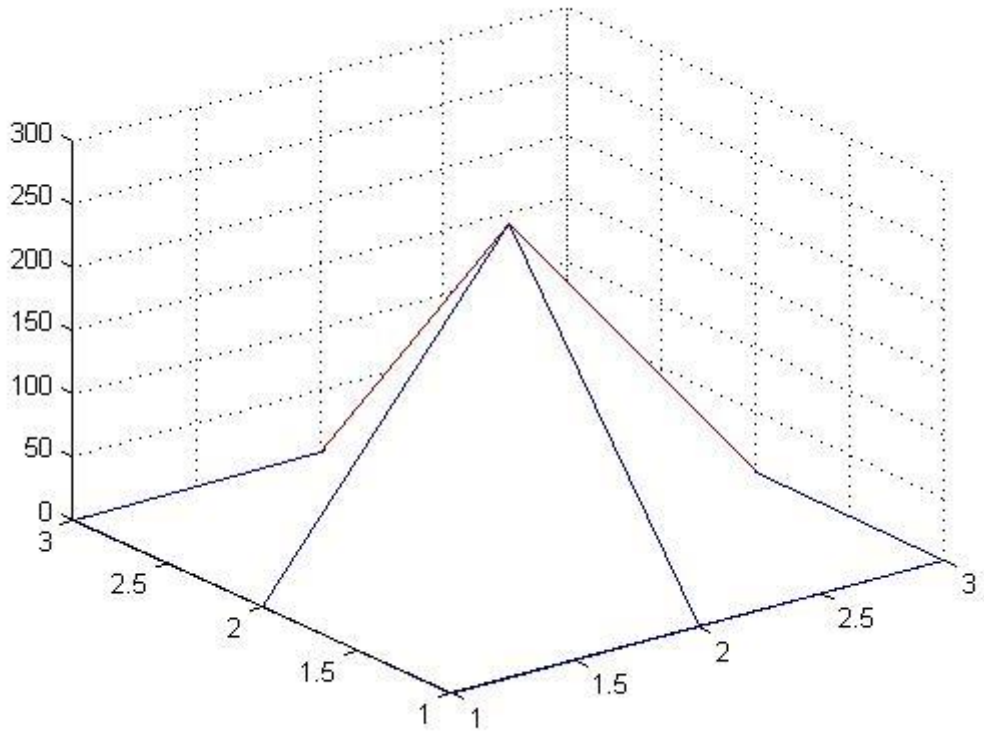
$$f(x,y) = \frac{1}{\sigma} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{1.11}$$

   Čím je menší sigma, tím menší vliv mají na filtraci okolní pixely. Pokud je obraz hodně zašuměný volí se větší sigma, pokud ne preferuje se z důvodu lepší lokalizace hran menší sigma viz obr. 4. Rozmaní probíhá mechanismem 2D konvoluce viz (1.6)

s Gaussovským konvolučním jádrem. Pro ukázku takovýchto 3x3 jader s různou hodnotou sigma viz obr. 5, 6



Obr.12a Gauss, 3x3, sigma=3, 8bitů



Obr.4 Gauss,3x3, sigma=0.3, 8bitů

$$h_{Gauss} = \frac{1}{254}\begin{pmatrix} 27 & 29 & 27 \\ 29 & 30 & 29 \\ 27 & 29 & 27 \end{pmatrix}$$

Obr.5 Gauss, 3x3, sigma=3, 8bitů

$$h_{Gauss} = \frac{1}{255}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 251 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

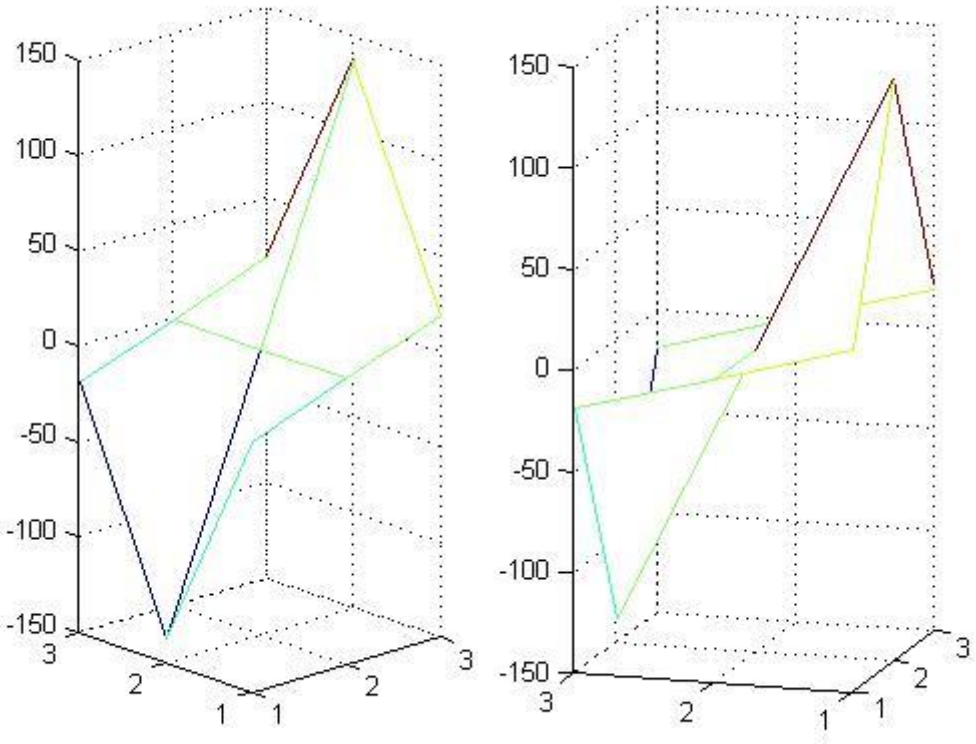
Obr.6 Gauss,3x3, sigma=0.3, 8bitů

Jádra byla navržena v prostředí MATLAB pomocí funkce „fspecial()“ pro dané parametry a dále upravena na 8-bitový kladný rozsah tak, aby váhovaný součet prvků jádra byl roven jedné.

Tato fáze hranové detekce minimalizuje vliv šumu a zohledňuje tak kritéria (2) a (3).

2. Aproximace gradientu jasu

V tomto kroku je vypočítán gradient jasu obrazové funkce ve formě jeho modulu viz (1.4) a úhlu viz (1.5). Parciální derivace se zde aproximují dvojí filtrací dle (1.6), s jádry pro směry x a y. Pro dobrou aproximaci gradientu je v článku [18] na straně 688 doporučeno využít jádro DoG, viz obr. 7 , 8 a 19, které je sice o 20% horší než optimální numericky vypočítaný hranový detektor, ale je možné ho analyticky popsat a tudíž snadno realizovat.



Obr.7 DoGx a DoGy s parametrem sigma=0.5

$$h_{DoGx} = \frac{1}{255}\begin{pmatrix} -19 & 0 & 19 \\ -138 & 0 & 138 \\ -19 & 0 & 19 \end{pmatrix}$$

Obr.8 DoGx, 3x3, sigma=3, rozsah +/- 255

$$h_{DoGy} = \frac{1}{255}\begin{pmatrix} 19 & 138 & 19 \\ 0 & 0 & 0 \\ -19 & -138 & -19 \end{pmatrix}$$

Obr.9 DoG,3x3, sigma=0.3, rozsah +/-255

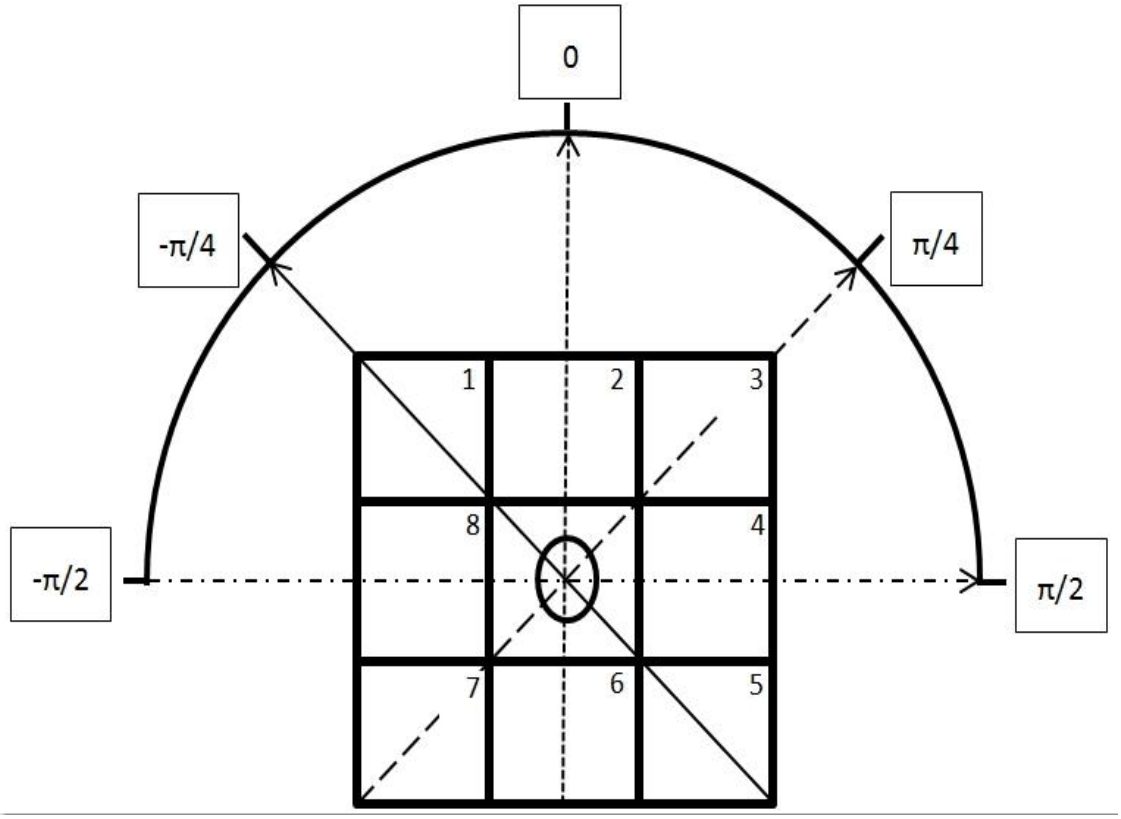
Tato fáze je stěžejní pro každou hranovou detekci. Jejím výstupem jsou informace o síle (modulu) a orientaci (úhlu) hran. Použité jádro zohledňuje (1), (2), (3).

3. Lokalizace hran

V tomto kroku dochází k upřesnění lokalizace hran na základě hledání lokálních maxim modulu gradientu ve směru, který udává jeho úhel. Matematicky lze úlohu hledání lokálních maximálních hodnot gradientu vyjádřit, jako hledání nulových hodnot první derivace modulu gradientu podle změny směru viz [19] str. 144 vztah (5.59).

V digitálním obraze má každý pixel, kromě okrajových, osm sousedních pixelů. Je tedy možné rozlišit čtyři směry: vertikální, horizontální a další dva po obou diagonálách. Lokalizace tedy probíhá tak, že se nejprve musí přiřadit úhel gradientu

těmto směrům. Musí se tzv. kvantizovat. Viz obr. 10 (daný pixel vyznačen kruhem, směry vyznačeny šipkami). Potom se porovnají hodnoty obrazové funkce pixelů sousedících s daným pixelem v daném směru s hodnotami obrazové funkce tohoto pixelu. Hodnota obrazové funkce tohoto pixelu není nulována jen v případě, že je to maximum, což je tehdy, když je větší než hodnoty obou sousedních pixelů.



Obr.10 Určení sousedních dvou pixelů na základě úhlu gradientu

Výstupem je matice maximálních hodnot modulu gradientu

Tato fáze hranové detekce zajišťuje především (2).

4. Dvojí prahování

V tomto kroku dochází k rozdělení maxim modulu gradientu na hrany a možné hrany. Děje se tak na základě porovnávání hodnoty daného pixelu s hodnotami tzv. vyššího (Th) a nižšího (Tl) prahu. Pixely, jejichž hodnota je nižší než hodnota Tl nejsou považovány za možné hrany. Pixely, jejichž hodnota je větší nebo rovna hodnotě Tl ale menší než hodnota Th jsou považovány za možné hrany a pixely, jejichž hodnota je větší nebo rovna hodnotě Th jsou považovány za hrany.

Tato fáze hranové detekce zohledňuje především (1).

5. ***Ověření konektivity***

V tomto kroku dochází k rozhodování, zda považovat možné hrany za hrany nebo ne. Děje se tak na základě konektivity. Za hrany budeme možné hrany považovat jen v případě, že se v jejich okolních pixelech 1-8 viz obr.10 nachází alespoň jeden pixel hrany.

Tato fáze hranové detekce zohledňuje především (1).

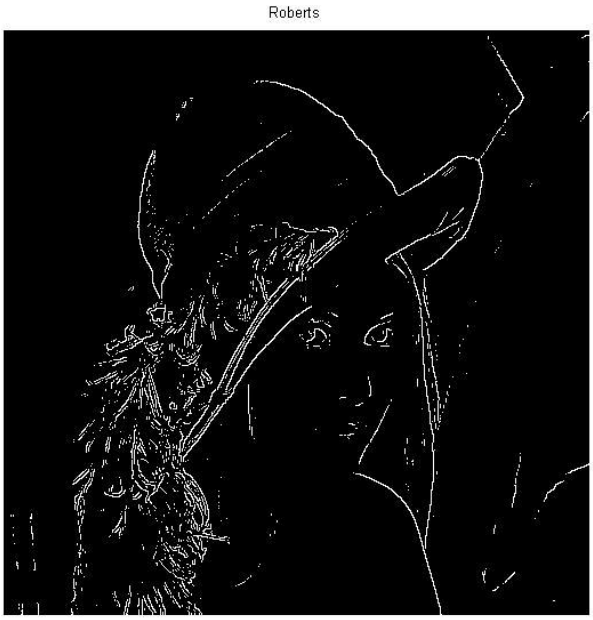
2.2 Implementace

2.2.1 Formulace úlohy

Implementujte Cannyho hranovou detekci video signálu s použitím vývojového kitu TMS320C6748 LCDK.

2.2.2 Simulace algoritmu v prostředí MATLAB

Nejprve jsem si Cannyho hranovou detekci vyzkoušel v MATLABu na funkci edge(). Tuto funkci nalezneme v Image Processing Toolboxu, v sekci Image Analysis v podsekci Object Analysis. Cannyho detekční metodu jsem na testovacím obrázku „lena.gif“ porovnal s ostatními implementovanými metodami viz obr.11.

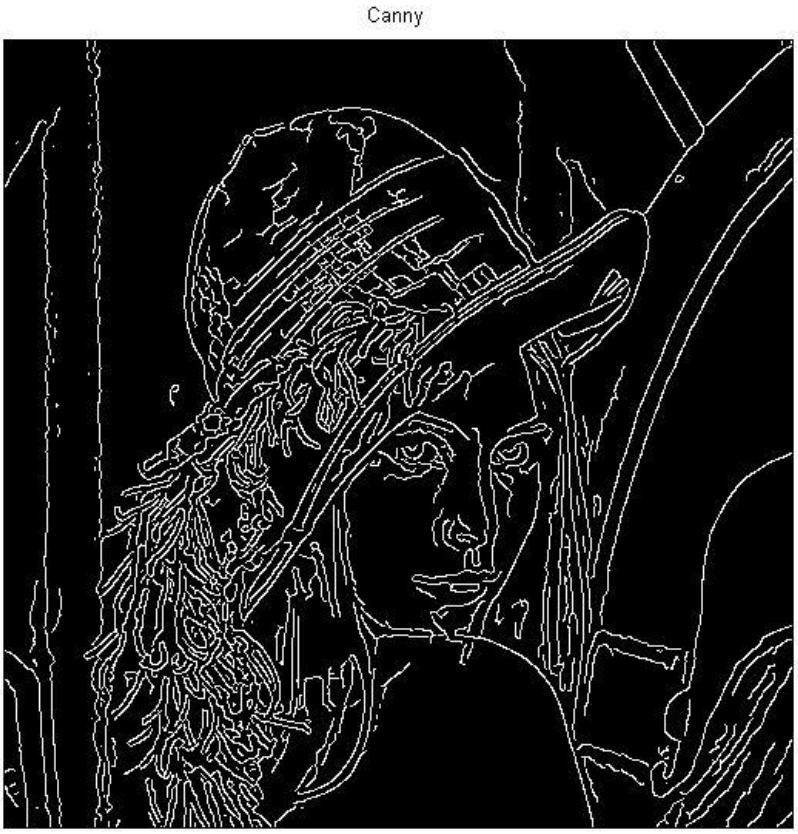


Prewitt

Sobel

Laplacian of Gaussian (LoG)

Zero-crossing



Obr.11 Porovnání hranových detektorů implementovaných v MATLABu ve funkci edge()

V porovnání s ostatními hranovými detektory vychází Cannyho metoda z hlediska rozpoznání hran nejlépe. Dále jsem si snažil metodu zažít. Měnil jsem hodnoty prahů, hodnoty použité sigmy a sledoval změny v detekovaných hranách. Potom jsem použil funkci open(), otevřel jsem si skript funkce edge() a analyzoval jsem tu část, která se týká Cannyho hranové detekce. Zjistil jsem, že je napsána příliš objektově a příliš využívá optimalizovaných Toolboxů. Hledal jsem proto skript s přímočařejším přepisem Cannyho algoritmu, který bych mohl lépe použít v LCDK. Nakonec jsem vzcházel z [22]. MATLAB jsem pak i nadále využíval během implementace Cannyho hranové detekce pro analytické účely. Často jsem využíval možnosti CCS5 uložit obsah paměti na daném adresovém rozsahu do datového souboru a možnosti MATLABu importovat data v této podobě. Takto jsem například kontroloval funkčnost mé implementace, porovnával ji s výchozím kódem a kódem implementovaným ve funkci edge().

2.2.3 Implementace na LCDK

Nejprve jsem se v [3] zběžně seznámil s hardwareovým zapojením videa na LCDK a v [1] s použitými moduly C6748. Poté jsem využil příkladu „vpif“ ze složky „examples“ z kolekce souborů StarterWare, který je ke stažení zde: [11]. Mým cílem v této části práce bylo zprovoznit přehrávání videa na LCDK, tedy zapisovat z vstupního bufferu na výstup, což již bylo obsaženo ve “vpif_lcd_loopback.c”, který je součástí příkladu “vpif”. Hlavní problém tedy bylo nastavení CCS5 pro správný překlad a nahrání “vpif” vzorového příkladu na LCDK. Dále bylo potřeba implementovat normu PAL, ve které pracovala použitá kompozitní kamera. Pro bližší pochopení převzatého kódu bylo nutné pochopit jeho strukturu a získat povědomí o funkci a nastavení použítých modulů, o video kodeku a DAC a také o formátu vstupních a výstupních dat. Poté zbývalo takto fungující kód doplnit o algoritmus Cannyho hranové detekce, zhodnotit jeho efektivnost a kvalitu a kód optimalizovat.

2.2.4 Hardwareové řešení zpracování videa na LCDK

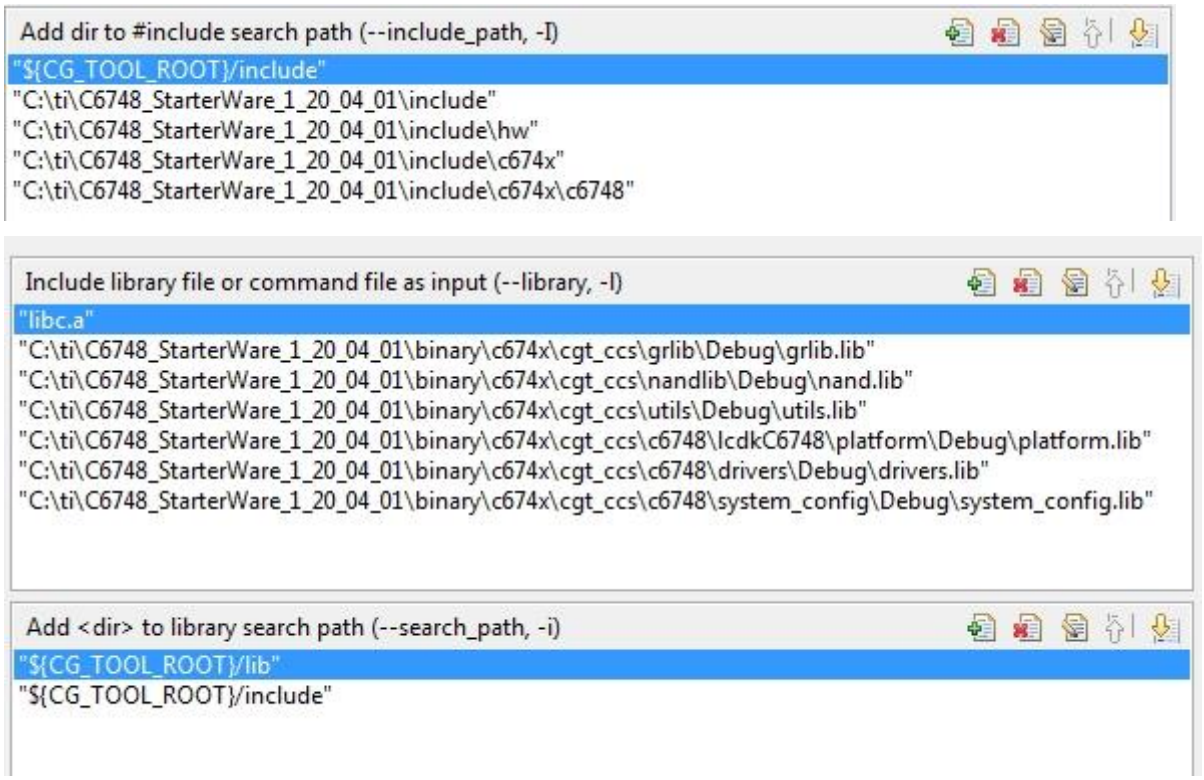
Podle [3], strany 9 je pro video vstup použit videodekodér TVP5147M1PFP, který je propojen s kompozitním video vstupem RCJ-014 přes pin 8 (VI_2_B). S megamodulem VPIF výstupními piny 43-47, 50-51(Y_2-Y_9) a datovým hodinovým signálem na pinu 40 (DATA_CLK) a I2C megamodulem C6748 přes piny 28 (SCL) a 29 (SDA).

Podle [3], strany 7 jsou výstupní piny megamodulu VPIF- VPIF_DOUT0-15 připojeny přes sběrnicový vysílač s napěťovým překladačem k video DAC THS8135 tak, že signály VPIF_DOUT0-4 odpovídají modrému kanálu, signály VPIF_DOUT5-10 odpovídají zelenému kanálu a signály VPIF_DOUT11-15 odpovídají modrému kanálu. Tím je implementován výstupní formát s pěti bitovým rozlišením v kanálech R a B a šestibitovým rozlišením v kanále G, tzv. RGB565. Výstupní analogové barevné signály jsou filtrovány a spolu s horizontálním a vertikálním synchronizačním signálem a DDC_SDA signálem jsou přiváděny na příslušné vodiče VGA konktoru.

2.2.5 Funkční spuštění „vpif_lcd_loopback.c“ na CCS5

Nejprve jsem si vytvořil nový workspace v souboru work, který jsem pojmenoval VIDEO_test1. Poté jsem otevřel CCS5 a vybral workspace VIDEO_test1. Poté jsem založil nový CCS projekt. (File=> New=> CCS Project) pojmenoval ho video_test1, výstupní typ nechal „Executable“ zaškrtl checkbox „Use default location“ Vybral C6000 jako použitou rodinu mého procesoru. Z roletového menu vybral „LCDK6748“. Z roletového menu „Connection“ vybral způsob připojení LCDK k počítači tj. „Texas Instruments XDS100v2 USB Emulator“. V sekci advanced settings jsem zvolil endianitu „little“ (je uvedená v [2]), vybral nejnovější compiler v mém případě TI v7.4.4, změnil výstupní formát na „eabi(ELF)“ Pro více informací o tomto rozhraní viz [13] odstavec 2.16 a 6.4. Dále je potřeba zvolit správný linker command file. Pro bližší informace viz [14] kapitola 7 Linker description. Zvolil jsem C6748.cmd. V menu „Runtime support library“ jsem nechal možnost „<automatic>“. V sekci „project templates and examples“ jsem zvolil „empty project“. Vše jsem potvrdil tlačítkem „finish“. Nyní se v „Project exploreru“

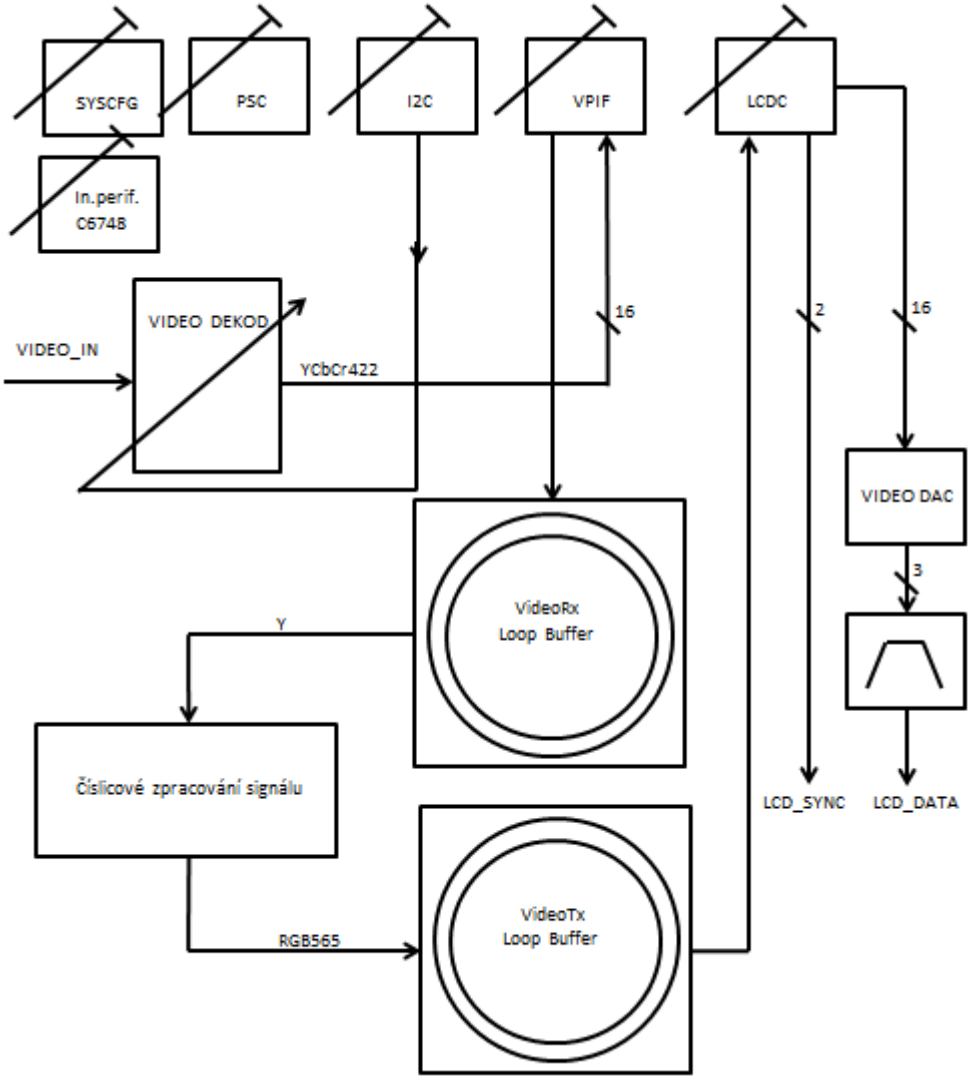
objevil prázdný projekt „video_test1“ s inicializovaným compilerem a linkerem pro C6748LCDK. Nyní jsem přistoupil k importu příkladu „vpif_lcd_loopback“ ze StarterWare. V „Project Exploreru“ jsem kliknul na projekt pravým tlačítkem a vybral „import“. Zvolil jsem „General“ potom „File system“ a zadal jsem cestu k souboru „vpif_lcd_loopback“ v sekci examples u nainstalovaného StarterWare. Potvrdil jsem výběr všech přítomných částí kódu a spustil import tlačítkem „finish“. Nyní se do projektu naimportovaly potřebné soubory, ale je ještě potřeba nastavit cesty k hlavičkovým souborům u compileru a k použitým knihovnám u linkeru. Pro tato nastavení je potřeba pravým tlačítkem myši kliknout na projekt v „Project Exploreru“, rozkliknout „C6000 Compiler“ a „C6000 Linker“ a pro „Include Options“ a „File Search Path“ doplnit adresy k potřebným hlavičkovým souborům a knihovnám. Viz Obr.12. A nyní už stačí jen „postavit“ projekt a nahrát ho přes připojený JTAG do C6748. Aby se tak stalo, stačí kliknout na tlačítko s ikonou brouka. Nakonec je potřeba zapojit kameru na kompozitní video vstup a monitor přes VGA konektor. Pro úspěšné spuštění programu musíme změnit v C6748.cmd v sekci „.far“ používaný paměťový prostor pro tuto sekci z SHRAM na DDR2. Pro použitou kameru bylo potřeba nastavit formát PAL v rozlišení 576i. Protože vpif_lcd_loopback.c předpokládá použití NTSC kamery v rozlišení 480i. Pro formát PAL 576i bylo potřeba nastavit příslušné registry megamodulu VPIF, pro informace o tomto modulu viz [1] str.1689. Pro PAL parametry požadovaného formátu SDTV BT.656 viz str. 1693, 1710. Dále bylo potřeba nastavit příslušné registry video dekodéru TVP5147M1PFP viz [23], kde se lze inspirovat na straně 100 příkladem 3. Nakonec jsem však použil výukový kód pana inženýra Václava Svobody, který z vpif_lcd_loopback.c vychází a PAL už implementuje.



Obr.12 CCS5-Nastavení adres k hlavičkovým souborům a knihovnám, nahoře compiler dole linker

2.2.6 Struktura programu- co se děje s video signálem

Signál z kompozitní PAL kamery je přiveden do dekodéru TVP5147M1PFP, kde je převeden na digitální formát YCbCr422 dle specifikace ITU-R BT.656. Poté je přiveden do megamodulu VPIF, kde je serializován a přiveden do vstupního kruhového bufferu Video Rx Loop Buffer, odkud se pro číslicové zpracování signálu využívá jen jasový kanál. Nahrávání a zpracování video signálu se děje blokově. Vždy se čeká na nahrání celého snímku, respektive půlsnímku, protože jde o prokládané video. Po zpracování signálu je signál zapsán ve formátu RGB565 do Tx Loop Buffer, odkud je načítán do tzv. rámcového bufferu řadiče LCD viz [1] str. 956 a 964. Řadič LCDC provádí deserializaci dat a generuje synchronizační signály pro LCD. Deserializovaná digitální RGB565 data pak putují do DAC THS8135, kde jsou převedena na analogové RGB signály, které jsou následně filtrovány. Viz obr.13.

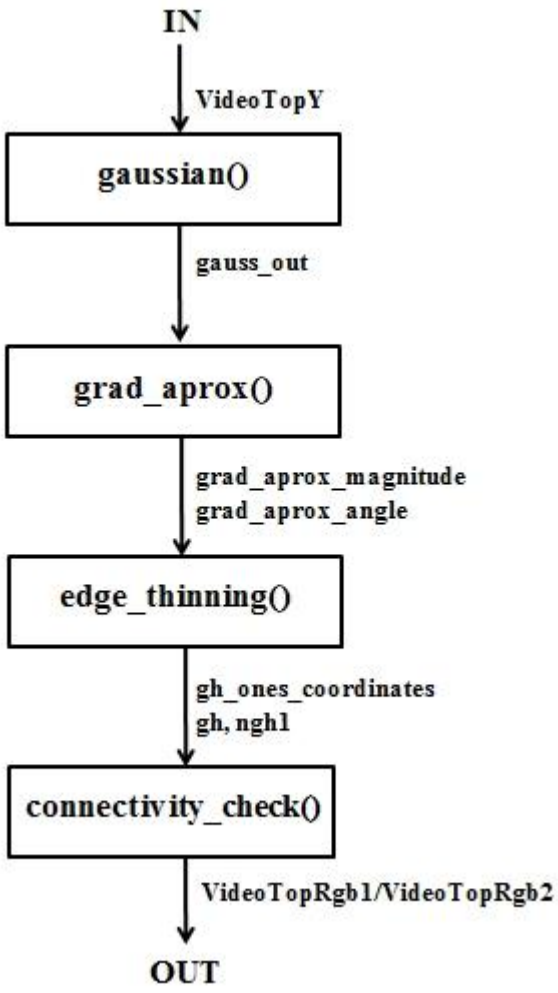


Obr.13 Průchod videosignálu

V kódu se nejprve inicializují použité moduly C6748, prostřednictvím I2C se nastaví dekodér TVP5147M1PFP. Inicializují se spínané buffery, které tvoří Video Rx/Tx Loop Buffer a také buffery, které tvoří rámcový buffer pro LCDC. Po inicializaci dojde k nahrávání, zpracování a ukládání video dat, přičemž se tak děje synchronizovaně.

2.2.7 Implementace algoritmu Cannyho hranové detekce

Cannyho hranovou detekci jsem implementoval v bloku číslicového zpracování signálu dle obr. 13. jako kaskádu funkcí vycházejících z postupu optimální hranové detekce popsané v teoretické části této úlohy. Pro blokové schéma hranové detekce viz obr.14.



Obr.14 Implementace Cannyho hranové detekce

1. ***gaussian()***

V této funkci jsem implementoval filtraci gaussiánem. For cykly jsem implementoval 2D konvoluci dle vztahu (1.6), přičemž jsem se vzhledem k efektivnímu využití paměti vyhnul použití 2D pole. Okrajové podmínky konvoluce jsem vyřešil použitím výřezu ze vstupního obrazu, který zajistil potřebné přesahy pro krajní pixely. Konvoluční jádro jsem navrhoval v matlabu použitím funkce fspecial() a následným přemapováním hodnot do osmi bitového neznaménkového rozsahu. Funkce je implementována celočíselně. Zkoušel jsem pro konvoluční jádro používat různé hodnoty sigma. Výsledný obraz jsem si v CCS5 exportoval za použití nástrojů „Expressions“ a „Memory browser“ do datového souboru, který jsem pomocí nástroje

„Import Data“ importoval do MATLABu, ve kterém jsem vizuálně porovnával vliv zvolené sigma na filtrovaný obraz. Toto porovnávání nešlo provádět přímo na monitoru, z důvodu nízkého rozlišení formátu RGB565. Výsledky filtrace se lišily velikostí přítomného šumu, který byl způsoben světelnými podmínkami při snímání scény.

2. grad_aprox()

V této funkci jsem implementoval aproximaci gradietnu dle vztahu (1.4) a (1.5) s použitím 2D konvoluce dle (1.6) pro výřez vstupního obrazu. Postupně jsem vyzkoušel Prewittovo, Sobelovo a DoG jádro. Výsledky jsem porovnával v matlabu, kde se ukázalo, že DoG jádro je z výše uvedených nejlepší. Pro výpočet modulu a úhlu gradient jsem používal odladěné funkce sqrtsp() a atansp() matematické knihovny “mathlib_c674x_3_1_0_0” kterou lze stáhnout ze stránek ti. Viz [24]. Z důvodu snížení výpočetní náročnosti jsem pro výpočet modulu gradientu nahradil odmocninu kvadrátů postupně jen kvadráty a poté jsem použil místo kvadrátů, absolutní hodnoty. V případě výpočtu úhlu, jsem atan() nahradil stromem ifů, který rovnou kvantizoval úhly dle obrázku 10. Tato funkce, je vzhledem k použitým operacím formát float časově nejnáročnější částí celé hranové detekce. Tato funkce také hledá největší hodnotu, která se v obraze vysktla a ukládá ji do proměnné maxval. Tato hodnota se potom využívá při dvojím prahování ve funkci edge_thinning() a nuluje se jednou za zpracování určitého počtu snímků, který je daný hodnotou REFRESH_VAL.

3. edge_thinning()

V této funkci jsem implementoval potlačení nemaximálních hodnot viz bod 3 „Lokalizace hran“ v teoretické části této úlohy a dvojí prahování viz bod 3 „Dvojí prahování“ tamtéž. Hodnoty pro nízký a vysoký práh byly voleny vzhledem k maxval získané v grad_aprox(). Výsledkem je hranová mapa gh, kde „1“ označuje hrany, „2“ možné hrany a „0“ pixely, které nejsou hrany. Dalšími výstupy jsou souřadnice hran dle gh a počet pixelů označených jako hrana. Hranovou mapu již bylo možné zobrazit na LCD displeji, aniž by vadilo nízké rozlišení výstupního formátu. Pro lepší odlišení jsem hrany zobrazoval bíle (0xFFFF) a možné hrany modře (0x001F). Zkoušel jsem přitom měnit hodnoty nízkého a vysokého prahu.

4. connectivity_check()

V této funkci jsem rozhodoval, zda jsou možné hrany skutečně hrany nebo ne a to na základě sousedství s pixely označených jako hrany. Průchod touto funkcí jsem zrychlil znalostí souřadnic a počtu pixelů označených jako hrany získanou v předešlé funkci.

2.2.8 Zhodnocení kvality výstupu Cannyho hranové detekce a optimalizace algoritmu

Implementace Cannyho hranové detekce dává dobré výsledky, ale algoritmus není optimální. Nelze zde mluvit o videu, ale o pohyblivých obrázcích. Obrázky se totiž zobrazují jen jednou za 3s. Přičemž jsem nahradil kritické sekce kódu (atan, sqrt) časově méně náročnými operacemi a vypustil jsem funkci gaussian(). Z hlediska času je kritická funkce grad_aprox() a edge_thinning(). Myslím, že je možné algoritmus obecně urychlit podvzorkováním vstupního obrazu a použitím intrinzických funkcí procesoru C6748. U funkce edge_thinning() by bylo možné místo hledání maximálních hodnot zkusit použít morfologickou erozi. Pro zkrácení doby výpočtu je možné použít jen výřez vstupního obrazu.

3 Středovlnný AM přijímač

V této úloze jsem řešil implementaci jednoduchého středovlnného radiového přijímače pomocí LCDK a externích obvodů využívající princip superhetu se směšováním do základního frekvenčního pásma.

3.1 Teoretický popis

3.1.1 Modulace a demodulace

Modulace je způsob přenosu modulačního signálu prostřednictvím parametrů nosného signálu, které modulační signál ovlivňuje. Demodulace je rekonstrukce modulačního signálu z modulovaného signálu. Modulace se nejčastěji využívá pro přenos nízkofrekvenčního signálu pomocí vysokofrekvenční harmonické nosné. Matematické vyjádření nosné udává vztah 2.1

$$s_c(t) = A_c \sin(2\pi f_c t + \varphi_c) \tag{2.1}$$

Modulace dělíme podle počtu modulačních signálů na jednoduché a složité, podle charakteru modulačního signálu a nosné na analogové (spojitý modulační signál v hodnotách i v čase + spojitá nosná v hodnotách i v čase), digitální (diskrétní modulační signál v hodnotách i v čase + spojitá nosná v hodnotách i v čase) a diskrétní (spojitý modulační signál v hodnotách i v čase + diskrétní nosná v hodnotách a v čase) a podle parametru nosné, který modulační signál ovlivňuje, na Amplitudovou (A), frekvenční (f) a fázovou (fi). V této úloze jsem pracoval s analogovou amplitudovou modulací.

3.1.2 Amplitudová modulace

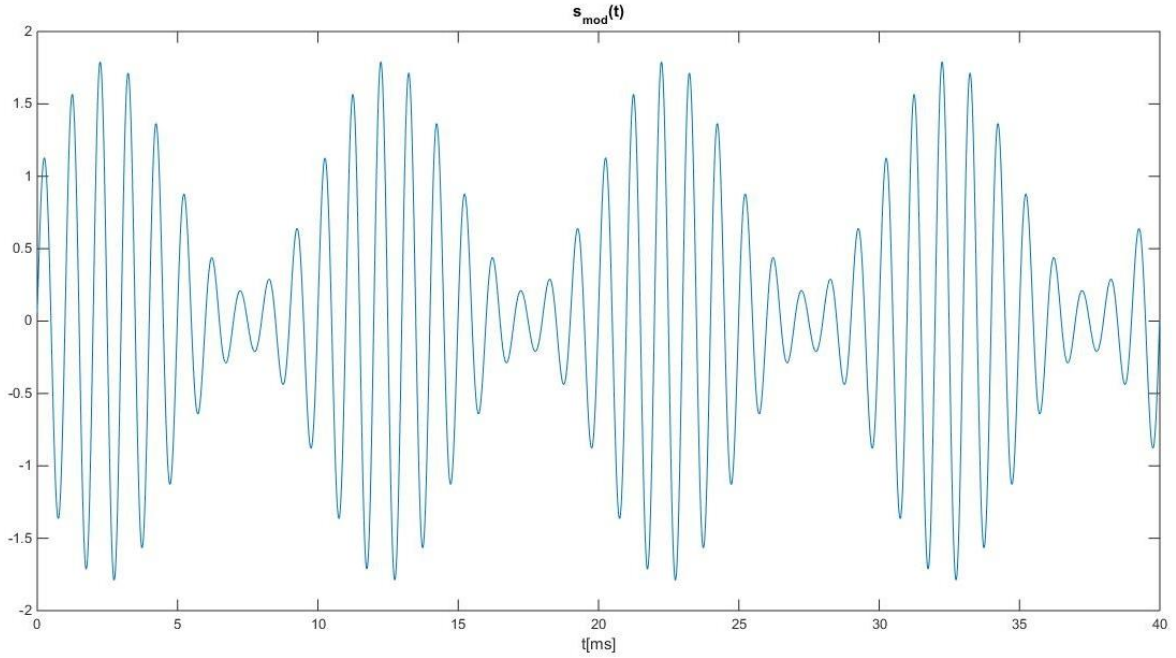
Amplitudová modulace je podle [25] dána přepisem (2.2).

$$s_{mod}(t) = A_c(1 + m \cdot s_{nf}(\mathrm{t}))\sin(2\pi f_c t + \varphi_c) \tag{2.2}$$

Kde $s_{mod}(t)$ je amplitudově modulovaný tzv. pásmový signál viz obr. 15, A_c je amplituda nosné, m je hloubka modulace definovaná dle vztahu (2.3), $s_{nf}(t)$ je nízkofrekvenční modulační signál a f_c je frekvence nosné.

$$m = \frac{A_{nf}}{A_c} \cdot 100\% \tag{2.3}$$

$$s_{nf}(t) = m \cdot \sin\left(2\pi f_{nf} t + \varphi_{nf}\right) \tag{2.4}$$



Obr.15 Pásmový signál, fs=100kHz, fc=1kHz, f_{nf}=100Hz, m=80%

Nyní se podíváme, jak bude vypadat nosná daná vztahem (2.1) modulovaná harmonickým nízkofrekvenčním signálem zadaným vztahem (2.4). (2.4) tedy dosadíme do (2.2) a roznásobíme. Pro výpočet součinu sinů budeme vycházet buď ze součtových vzorců (2.5), ze kterých získáme vztah (2.8), podle kterého součin rozepíšeme, nebo z Eulerova vzorce pro definici komplexní exponenciály (2.9), pomocí něhož odvodíme vztahy pro sinus (2.11) a cosinus (2.10). Siny pak v součinu sinů nahradíme tvarem (2.11), roznásobíme, a komplexní exponenciály přeskupíme tak, aby vyhovovaly (2.10). Oběma popsanými způsoby získáme vztahy (2.12) a (2.13), které spolu s nosnou tvoří výsledný vztah pro pásmový signál (2.14). Je vidět, že pásmový signál se skládá z nosné frekvence a rozdílové a součtové frekvence harmonického nf signálu viz Obr. 22. Pro složitější nf signál bude pásmový signál obsahovat frekvenci nosné a dále dolní (LSB) a horní (USB) postranní pásmo. Pro Evropu je pro AM středovlnné vysílání stanovena šířka postranního pásma na 4.5kHz. Sousední nosné jsou tedy od sebe frekvenčně vzdáleny 9kHz. V současné době ale nejsou všechny AM kanály obsazeny. Pro AM rozhlasové vysílání se v České republice používá 80% hloubka modulace.

$$\sin(x \pm y) = \sin(x)\cos(y) \pm \cos(x)\sin(y) \tag{2.5}$$

$$\cos(x \pm y) = cos(x)\cos(y) \mp \sin(x)\sin(y) \tag{2.6}$$

$$\sin(x) \cdot \cos(y) = \frac{\sin(x+y) + \sin(x-y)}{2} \tag{2.7}$$

$$\sin(x) \cdot \sin(y) = \frac{\cos(x-y) - \cos(x+y)}{2} \tag{2.8}$$

$$e^{j\varphi} = \cos(\varphi) + j \cdot \sin(\varphi) \tag{2.9}$$

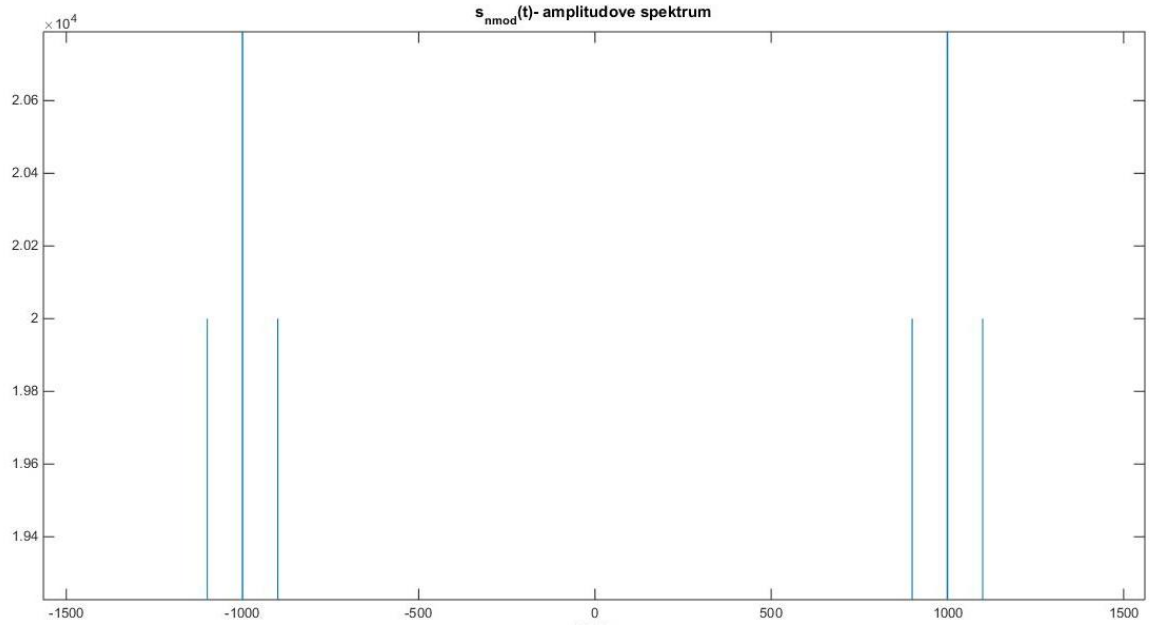
$$\cos(\varphi) = \frac{e^{j\varphi} + e^{-j\varphi}}{2} \tag{2.10}$$

$$\sin(\varphi) = \frac{e^{j\varphi} - e^{-j\varphi}}{2j} \tag{2.11}$$

$$s_{c-}(t) = \frac{A_c \cdot \mathrm{m}}{2} \cos\left(2\pi\left(f_c - f_{nf}\right)t + \varphi_c - \varphi_{nf}\right) \tag{2.12}$$

$$s_{c+}(t) = -\frac{A_c \cdot \mathrm{m}}{2} \cos\left(2\pi\left(f_c + f_{nf}\right)t + \varphi_c + \varphi_{nf}\right) \tag{2.13}$$

$$s_{mod}(t) = s_{c-}(t) + s_c(t) + s_{c+}(t) \tag{2.14}$$



Obr. 16 Modulové spektrum pásmového signálu, fs=100kHz, fc=1kHz, f_{nf}=100Hz, m=80%

3.1.3 Princip amplitudové demodulace se směšováním do základního pásma

Amlitudová demodulace, kterou jsem použil v této úloze, využívá změny frekvence při součinu dvou harmonických funkcí a následné filtrace filtrem typu dolní a horní propust. Pásmový signál (2.2) vynásobíme demodulačním signálem (2.15). Součin rozepíšeme buď pomocí součtových vzorců nebo pomocí Eulerova vztahu obdobně jako předtím u modulace a získáme tvar (2.20), který pro přehlednost sestavíme z členů (2.16),

(2.17), (2.18) a (2.19). Pokud budeme mít radiopřijímač naladěný, potom $f_c=f_d$ a tedy s_{dem1} bude stejnosměrná složka, kterou odstraníme filtrem typu horní propust, složky s_{dem2} a s_{dem4} mají vysokou frekvenci a odstraníme je filtrem typu dolní propust, složka s_{dem3} obsahuje užitečný demodulovaný nf signál.

$$s_d(t) = A_d \sin(2\pi f_d t + \varphi_d) \tag{2.15}$$

$$s_{dem1}(t) = \frac{A_c A_d}{2} \cdot \cos(2\pi(f_c - f_d) \cdot t + \varphi_c - \varphi_d) \tag{2.16}$$

$$s_{dem2}(t) = -\frac{A_c A_d}{2} \cdot \cos(2\pi(f_c + f_d) \cdot t + \varphi_c + \varphi_d) \tag{2.17}$$

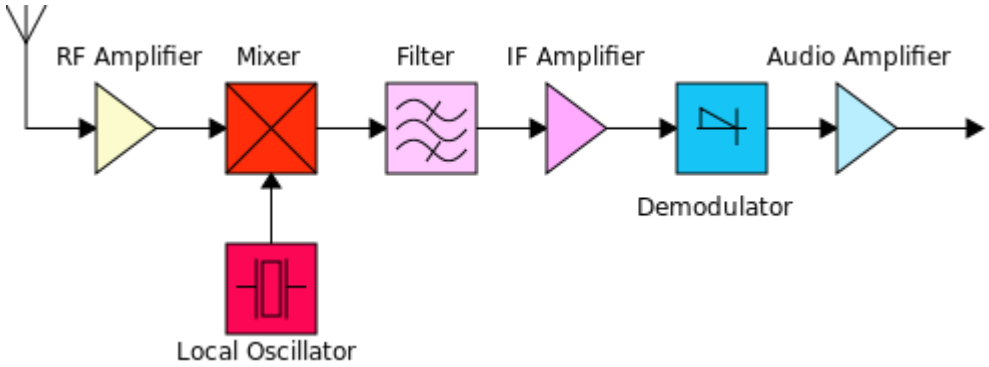
$$s_{dem3}(t) = \frac{A_c A_d}{2} \cdot m \cdot s_{nf}(\mathrm{t}) \cdot \cos(2\pi(f_c - f_d) \cdot t + \varphi_c - \varphi_d) \tag{2.18}$$

$$s_{dem4}(t) = -\frac{A_c A_d}{2} \cdot m \cdot s_{nf}(\mathrm{t}) \cdot \cos(2\pi(f_c + f_d) \cdot t + \varphi_c + \varphi_d) \tag{2.19}$$

$$s_{dem}(t) = s_{dem1}(t) + s_{dem2}(t) + s_{dem3}(t) + s_{dem4}(t) \tag{2.20}$$

3.1.4 Princip superheterodynního rádiového přijímače

Ve své práci jsem vycházel ze základních myšlenek heterodynního AM přijímače (viz obr. 17). Preselektorem (laditelná pásmová propust) je vybrán zvolený AM kanál(není na obrázku). RF signál vybraného AM kanálu je zesílen v RF zesilovači a přiveden do směšovače. Do směšovače je dále zaveden harmonický signál z lokálního oscilátoru, jehož frekvence je závislá na zvoleném AM kanálu. Frekvence lokálního oscilátoru je nastavována v součinnosti s preselektorem tzv. spřaženým laděním tak, aby byla nosná frekvence na výstupu směšovače stejná pro všechny AM kanály. Pro AM středovlnný rozhlasový příjem se používá mezifrekvence 455kHz. Potom následuje filtr typu pásmová propust, který odstraní nežádoucí složky po směšování a ještě více potlačí vliv ostatních AM kanálů na přijatý signál. Dále následuje zesilovač a demodulátor. Jako demodulátor se velmi často používá diodový detektor maximálních hodnot, který je tvořený diodou v sérii s paralelní kombinací RC. Nf signál je následně zesílen a přiveden do reproduktoru.



Obr.17 Princip heterodynního radiopřijímače

3.1.5 Seznam AM středovlnných vysílačů a rádiostanic v ČR

Název vysílače	Program	Kmitočet	ERP	Vých. délka	Sev. šířka	Výška nad m.	Stanoviště	Okres
Č. Budějovice	Rádio Dechovka	1233	1000	14-29-37	48-59-25	401	České Budějovice, Trocnovská 53, Husova kolonie	České Budějovice
Č.Budějovice	ČRo Dvojka/ČRo Plus	954	30000	14-29-37	48-59-25	401	České Budějovice, Trocnovská 53, Husova kolonie	České Budějovice
Dobrochov	ČRo Dvojka/ČRo Plus	954	200000	17-07-29	49-23-07	306	Brodek, Dobrochov	Prostějov
Domamil	ČRo Dvojka/ČRo Plus	1332	50000	15-42-41	49-04-25	547	Domamil, Domamil	Třebíč
Karlovy Vary (St. Role)	ČRo Dvojka/ČRo Plus	954	20000	12-49-26	50-14-22	433	Stará Role u KV, Stará Role	Karlovy Vary
Liblice	ČRo Dvojka/ČRo Plus	639	750000	14-53-13	50-04-02	235	Český Brod, Liblice	Kolín
Ostrava	ČRo Dvojka/ČRo Plus	639	30000	18-11-38	49-48-43	234	Ostrava-Svinov, Ostravská	Ostrava
Topolná	ČRo 1-Radiožurnál	270	50000	17-31-04	49-07-27	181	Topolná u Uh.Hradiště, Topolná	Uherské Hradiště
Zbraslav	Country radio	1062	20000	14-22-21	49-56-59	335	Jíloviště, Jíloviště	Praha-západ
Zbraslav	Country radio	1062	1000	14-22-21	49-56-59	335	Jíloviště, Jíloviště	Praha-západ

Obr.18 Seznam AM středovlnných vysílačů v ČR

Přičemž kmitočet je uváděn v kHz a hodnota ERP ve wattech.

3.2 Implementace

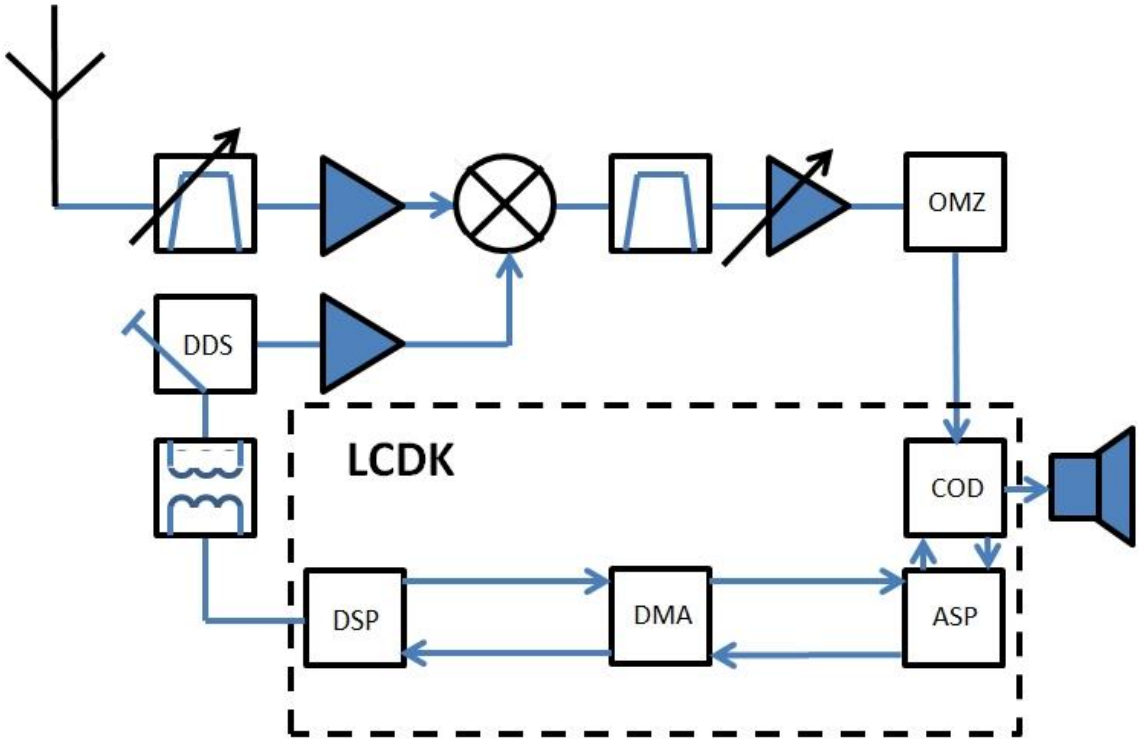
3.2.1 Formulace úlohy

Implementujte středovlnný AM radiopřijímač na principu přímého směšování do základního pásma s využitím LCDK k přelaďování AM kanálů a k nastavování jejich hlasitosti.

3.2.2 Koncept obvodu pro AM příjem středovlnného rádiového signálu

Při návrhu jsem vycházel z blokového diagramu na obr.19, který vychází z heterodynního přijímače viz obr.17. Feritová anténa je paralelně spojena s ručně laditelným kondenzátorem a tvoří tak paralelní rezonanční obvod- preselektor. Ladícím kondenzátorem preselektoru vybereme požadovaný AM kanál. RF signál dále putuje přes koaxiální kabel do RF zesilovače, ze kterého signál směřuje do směšovače, tvořeného analogovou násobičkou. V programu si zvolíme požadovaný AM kanál, požadovanou hlasitost a případně číslicovou filtraci. Pomocí JTAGu program zavedeme do paměti a spustíme ho. Přes emulované rozhraní SPI naprogramujeme obvod přímé číslicové syntézy (DDS). DDS je od LCDK galvanicky oddělený. DDS vygeneruje demodulační signál o dané frekvenci, který je zesílen a přiveden do směšovače. Výstupní signál ze směšovače je filtrován, zesílen a jeho amplituda je omezena v bloku OMZ. Signál je poté přiváděn na audio vstup kodéru COD, kde je filtrován, převeden do číslicové podoby a odeslán do bufferu audio sériového portu McASP (ASP), kde jsou audio data formátována a odebírána blokem přímého přístupu do paměti DMA, který těmito daty plní daný datový prostor viz

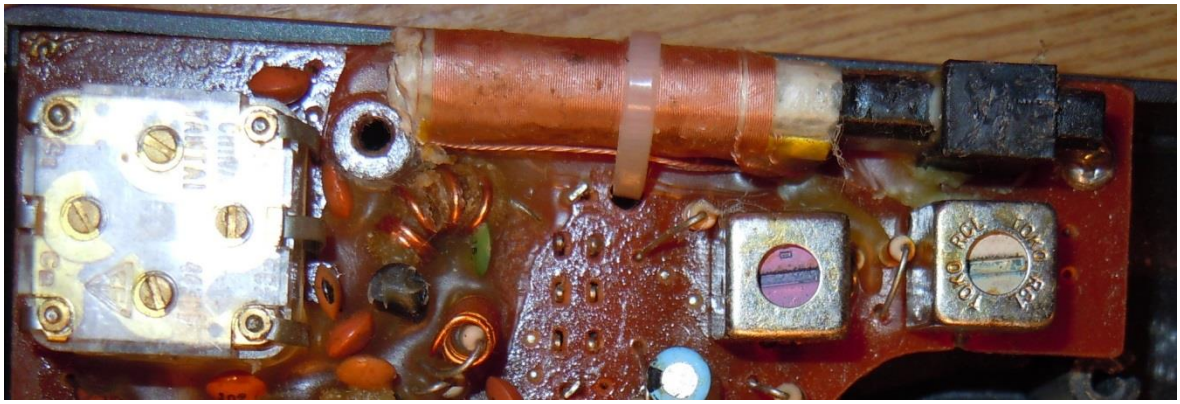
obr.8. Poté probíhá k číslicovému zpracování. Zpracovaný signál je poslán zpět do audiokodeku, který ho přeposílá na výstup audia, ke kterému lze připojit reproduktory nebo sluchátka a poslouchat zvolenou rozhlasovou stanici. Nyní budou detailněji popsány jednotlivé části přijímače.



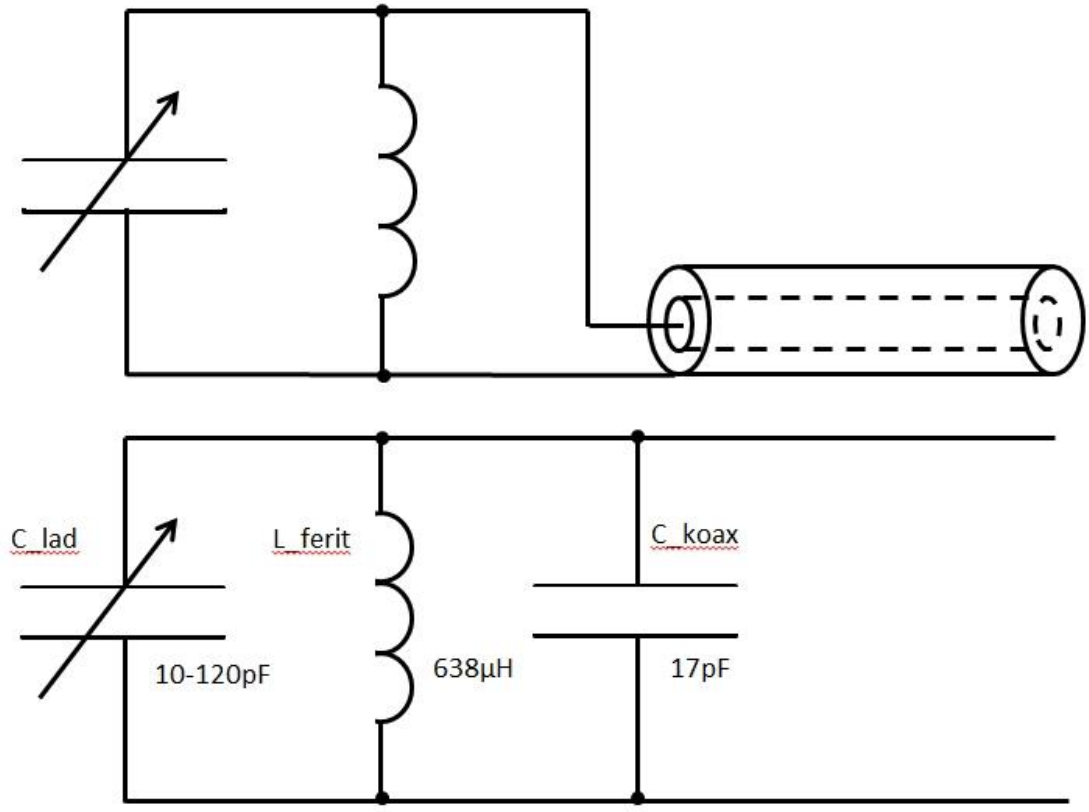
Obr.19 Blokový diagram realizovaného AM přijímače

3.2.3 Anténa, preselektor a propojovací koaxiální kabel

Feritovou anténu a ladící kondenzátor jsem získal ze starého německého rádia. Viz obr. 20. Desku s plošnými spoji jsem zbavil ostatních osazených součástek, vyčistil ji a dále použil pro konstruovaný radiopřijímač. Měřením RLC metrem Smart Tweezers jsem určil hodnotu indukčnosti antény, kapacitní rozsah laditelného kondenzátoru a kapacitu cca 16cm dlouhého koaxiálního kabelu. Viz Obr. 21. Pro výpočet rezonanční frekvence preselektoru jsem použil Thompsonův vztah (3.1). Pro mezní polohy kapacity laditelného kondenzátoru s přihlédnutím ke kapacitě koaxiálního kabelu: 27-137pF, vyšly rezonanční frekvence: 1212-538kHz.



Obr. 20 Feritová anténa a ladící kondenzátor



Obr.21 Zapojení a naměřené hodnoty laditelného kondenzátoru, feritové antény a koaxiálního kabelu

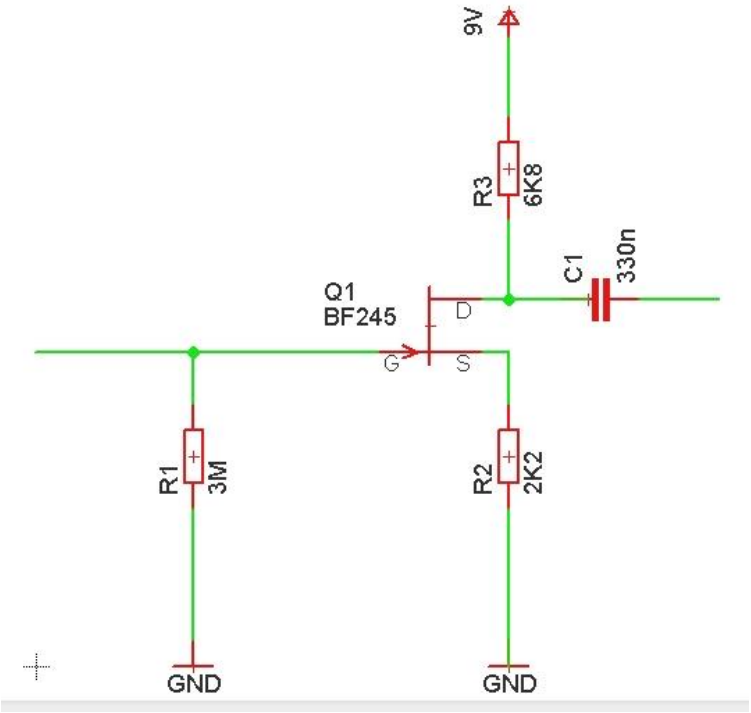
$$f_0 = \frac{1}{2\pi\sqrt{LC}} \tag{3.1}$$

3.2.4 RF zesilovač

Zesilovač je tvořen kaskádním zapojením tří tranzistorů, přičemž na vstupu je JFET , který má jen malé zesílení, ale zato z důvodu velkého vstupního odporu neovlivňuje předchozí rezonanční obvod. Další dva stupně tvoří BJT zesilovače se společným emitorem. Zesilovač je postaven tak, aby se zesílený RF signál mohl v případě potřeby odvádět z kteréhokoliv stupně.

1.stupeň

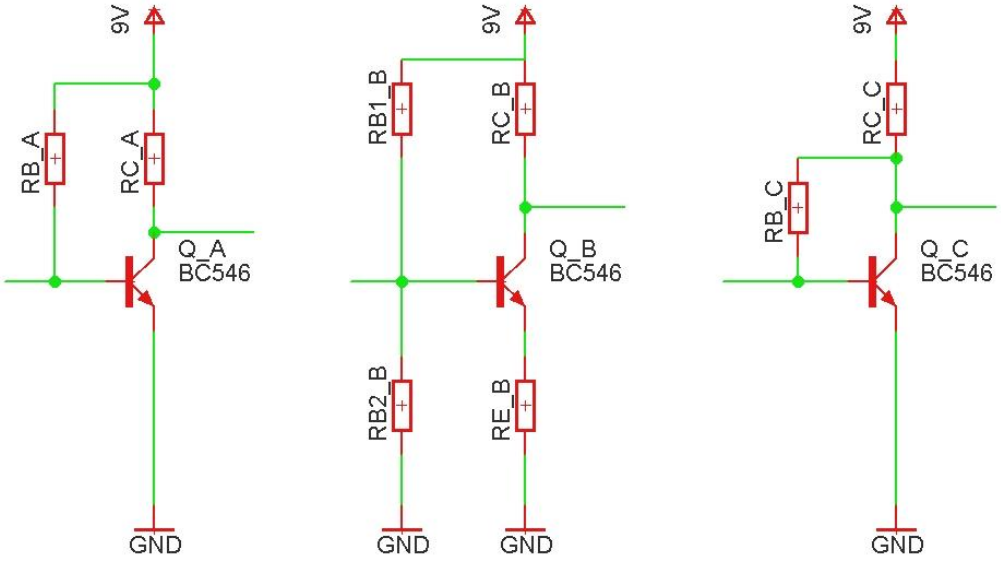
První stupeň zesilovače je tvořen zapojením SS s JFETem BF245 viz Obr.22, jehož zapojení i s použitými součástkami jsem převzal z [30]. Z měření napětí multimetrem MT-1232 jsem zjistil parametry pracovního bodu: Q(U_{GS}=-1,31V, U_{DS}=3,55V, I_D=0,63mA). Napěťové zesílení tohoto stupně je cca 2, měřeno osciloskopem TDS 1001B.



Obr.22- 1.stupeň RF zesilovače

2. a 3.stupeň

Druhý a třetí stupeň zesilovače je tvořen zapojením SE s tranzistorem BC546. Postupně jsem vyzkoušel všechny běžně používané varianty zapojení a teplotní stabilizace viz Obr.29 a rozhodl se z důvodu nízké frekvenční závislosti zesílení, dobré teplotní stability a nízkého počtu potřebných součástek pro variantu C.



Obr.23- Varianty nastavení pracovního bodu a teplotní stabilizace zesilovače CE s BJT

Nastavení pracovního bodu- Varianta C

1) Předpoklady: V_{CC}=9V, β_{DC}=270 (zvoleno jako kompromisní hodnota z katalogu)
2) V_{CE}=4.5V (zvoleno jako polovina napájecího napětí)
3) I_C=4mA (zvolená hodnota)
4) $I_B \approx \frac{I_C}{\beta_{DC}} = \frac{4 \cdot 10^{-3}}{270} \doteq 15\mu A$
5) V_{BE}=0.7V
6) $R_B = \frac{V_{CE}-V_{BE}}{I_B} = \frac{4,5-0,7}{15 \cdot 10^{-6}} = 253\,333\Omega => 270K$ (zvolena hodnota řady E12)
7) $R_C = \frac{V_{CC}-V_{CE}}{I_B+I_C} = \frac{9-4,5}{15 \cdot 10^{-6}+4 \cdot 10^{-3}} = 1120\Omega => 1K2$ (zvolena hodnota řady E12)

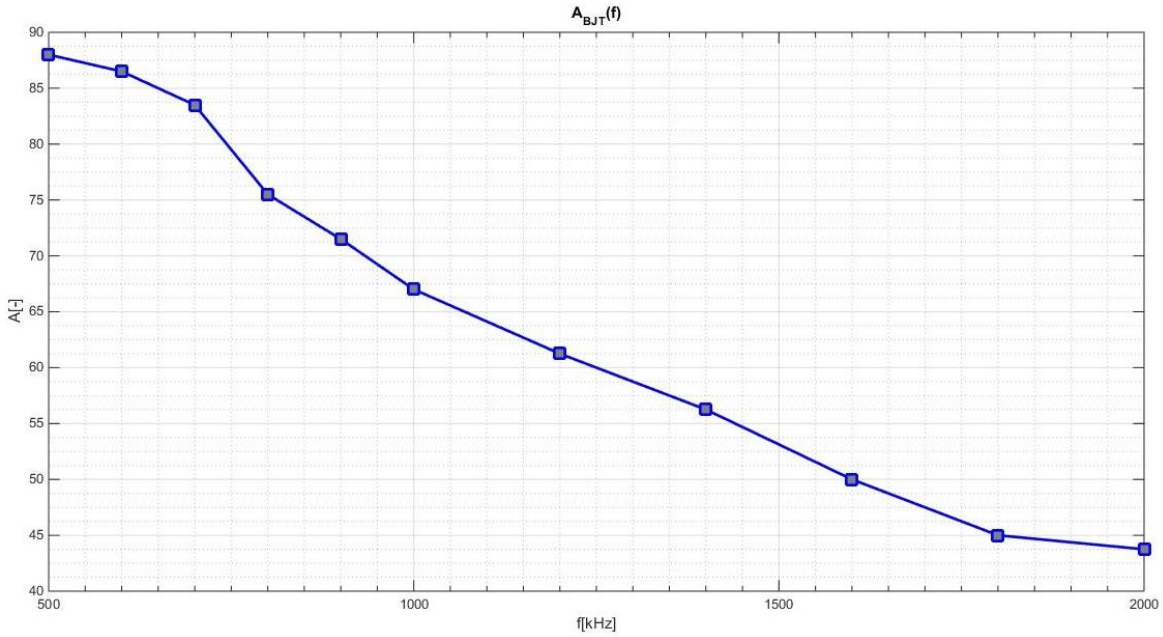
Proměření pracovního bodu

Měření jsem prováděl multimetrem MT-1232, pro napájení jsem použil regulovatelný zdroj napětí MW9115GS. Z Výsledků měření viz tab. 5 je vidět, že pracovní bod přibližně odpovídá nastavovanému pracovnímu bodu.

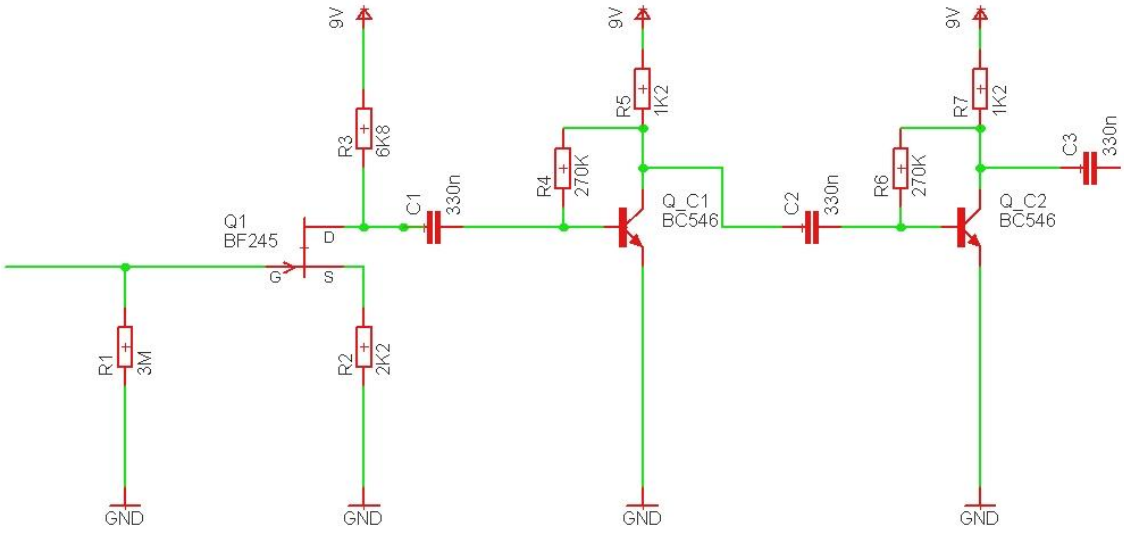
Změřeno:	**Spočítáno:**
V_{CC}=9,3V	I_C=3,7mA
V_{CE}=4,78V	I_B=15,17µA
V_{BE}=0,683V	β_{DC}=248
Tab.5 Ověření nastavení pracovního bodu	

Změření napěťového zesílení- Varianta C

Měření jsem prováděl v režimu „MEASURE“ dvoukanálovým osciloskopem TDS1001B. Sonda kanálu 1 měřila vstupní napětí [V_{pp}] druhá sonda pak výstupní napětí [V_{pp}]. Podílem výstupního ku vstupnímu napětí jsem získal zesílení. Zesílení jsem měřil v nf oblasti. Jako generátor jsem použil mp3 přehrávač s testovacími vzorky vytvořenými v MATLABu. Měřil jsem na frekvencích 440Hz, 1kHz a 10kHz na čtyřech hodnotách vstupního napětí. Zesílení na těchto hodnotách jsem potom zprůměroval a vyšlo 123x na všech měřených frekvencích. Pro toto měření jsem použil vazební bipolární elektrolytické kondenzátory s hodnotou kapacity 10µF. Zesílení jsem také měřil v rozmezí frekvencí 500-1000kHz s krokem 100kHz a 1200-2000kHz s krokem 200kHz. Jako generátor jsem používal DDS, který je popsán v oddíle 3.2.5. Frekvenční průběh zesílení ukazuje graf na obr.24. Pro toto měření jsem použil svitkové kondenzátory s hodnotou kapacity 330nF. Celkové zapojení RF zesilovače je na obrázku 25.



Obr.24- Zesílení jednoho BJT zesilovacího stupně



Obr.25- RF zesilovač

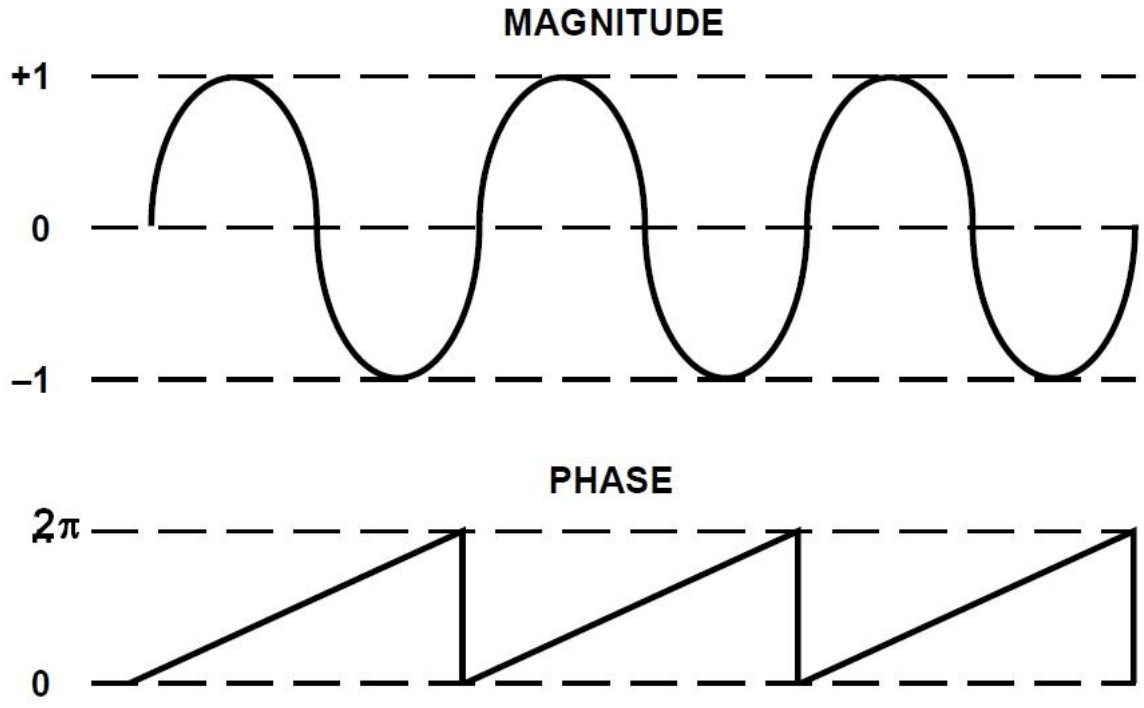
3.2.5 Přímý číslicový syntetizér (DDS)

Princip DDS

Pro vysvětlení principu DDS budu vycházet z [30].

Úlohou DDS je vytvořit amplitudu harmonické funkce v daném čase o dané frekvenci viz (3.2).

Závislost amplitudy sinu na čase je nelineární, avšak závislost fáze na čase je lineární viz Obr.32. Při řešení úlohy DDS budeme postupovat tak, že určíme fázi požadovaného harmonického průběhu, kterou pak převedeme na informaci o amplitudě.



Obr.26 Časová závislost amplitudy a fáze funkce sinus

$$a(t) = \sin(\omega t) \tag{3.2}$$

Úhlová rychlost je dána pomocí (3.3) jako změna fáze (fázový krok) za krátký časový úsek. V případě DDS nahradíme tento časový úsek periodou pracovního hodinového signálu a výraz upravíme pro vyjádření Δφ viz (3.4). Úhlová rychlost je rovna úhlové frekvenci, definované v (3.5), kterou dosadíme do vztahu (3.4) a dostáváme výraz (3.6), ve kterém už figuruje požadovaná frekvence výstupního signálu. V DDS je konstanta π nahrazena rozlišením fázového akumulátoru viz (3.7). Pro okamžitou hodnotu fáze v diskrétním čase N pak platí výraz (3.8). Okamžitá hodnota fáze představuje index do tabulky hodnot amplitudy, která je uložena v paměti. Tato hodnota je potom převedena pomocí DA převodníku a filtru typu dolní propust do analogové podoby, čímž je splněna úloha DDS. Ještě k terminologii. Výraz (3.8) je označován jako fázový akumulátor a převodní tabulka jako SIN ROM nebo LUT (Lookup table). Část DDS, která se zabývá volbou f_{out} a φ_{out} bývá označována jako NCO (Numerical Controlled Oscillator). φ_{out} se nastavuje přičtením konstanty (offsetu) k fázovému akumulátoru.

$$\omega = \frac{\Delta\varphi}{\delta t} \tag{3.3}$$

$$\Delta\varphi = \frac{\omega}{f_{MCLK}} \tag{3.4}$$

$$\omega = 2\pi f_{out} \tag{3.5}$$

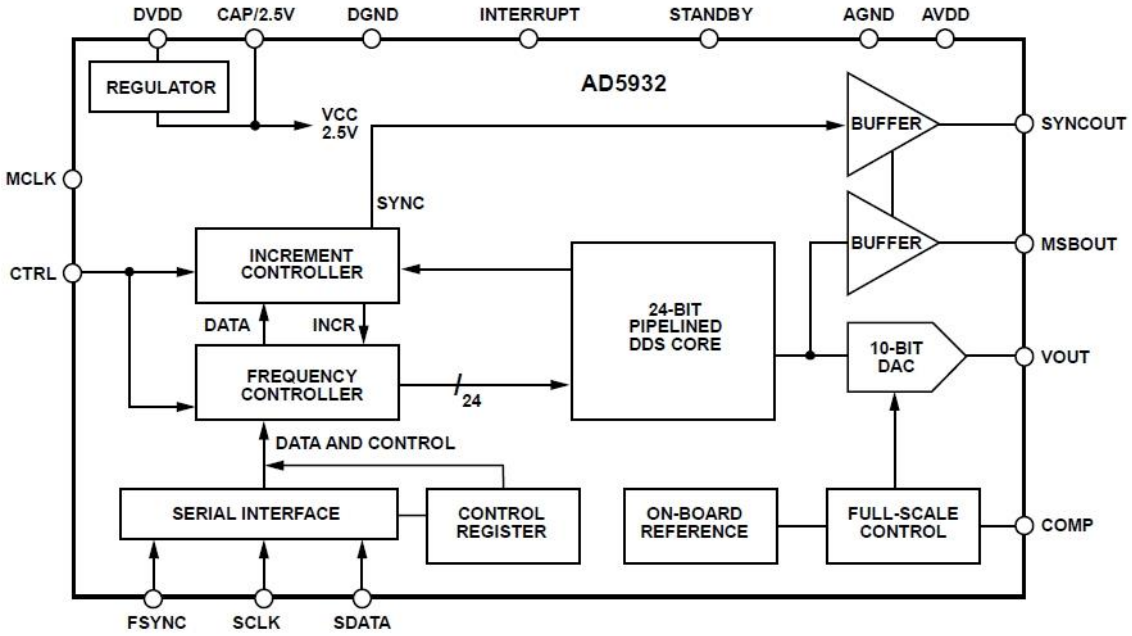
$$\Delta\varphi = 2\pi \cdot \frac{f_{out}}{f_{MCLK}} \tag{3.6}$$

$$\Delta\varphi = 2^n \cdot \frac{f_{out}}{f_{MCLK}} \tag{3.7}$$

$$\varphi[N] = \sum_{k=1}^{N} \Delta\varphi[k] \tag{3.8}$$

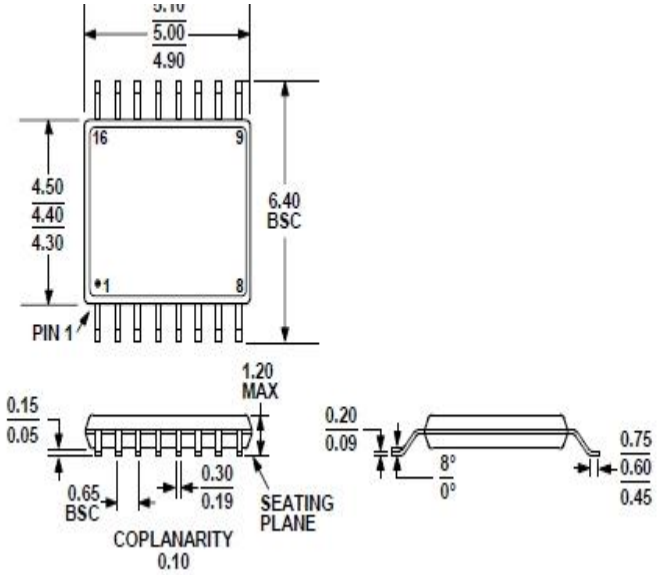
AD5932

Pro návrh jsem použil obvod AD5932, který implementuje DDS a automatické přelaďování frekvencí. Neumožňuje však přímo měnit fázi výstupního signálu. Pro generování harmonického signálu je potřeba určit počáteční frekvenci F_{START}, velikost frekvenčního kroku Δf, počet vykonaných frekvenčních kroků N_{INCR} a případně dobu trvání generování dílčí frekvence v rámci přelaďování. Obvod je řízen DSP prostřednictvím rozhraní SPI viz Obr.27. Celkem je třeba prostřednictvím SPI odeslat 7 16-bitových slov. V každém slovu představují horní 4 bity adresu příslušných registrů a zbývajícími bity se nastavují příslušné registry. Činnost obvodu je po naprogramování spuštěna přivedením úrovně High na vstup CTRL. Pro komunikaci s AD5932 po SPI a pro emulaci SPI pomocí GPIO vznikly programy v jazyku C, které jsou umístěny na nosiči přiloženém k této práci. Základní elektrické údaje shrnuje tabulka 6. Informace o tomto obvodu jsem čerpal z datasheetu [26]. Informace o jeho programování pak z [27] a parametrech komunikačního rozhraní SPI z [28]. Pro tento obvod jsem navrhnul zvláštní tištěný spoj viz obr. 30, který obsahuje podpůrné součástky pro jeho bezproblémovou činnost. Návrhy tištěného spoje v Eaglu jsou k dispozici na přiloženém nosiči pod označením DP-BLOK2-D. Jako zdroj hodinového signálu MCLK byl použit krystalový oscilátor SG8002DB naprogramovaný na 50MHz.

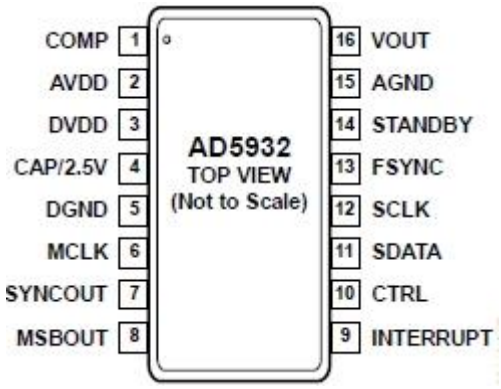


Obr.27 Blokové schéma programovatelného generátoru AD5932

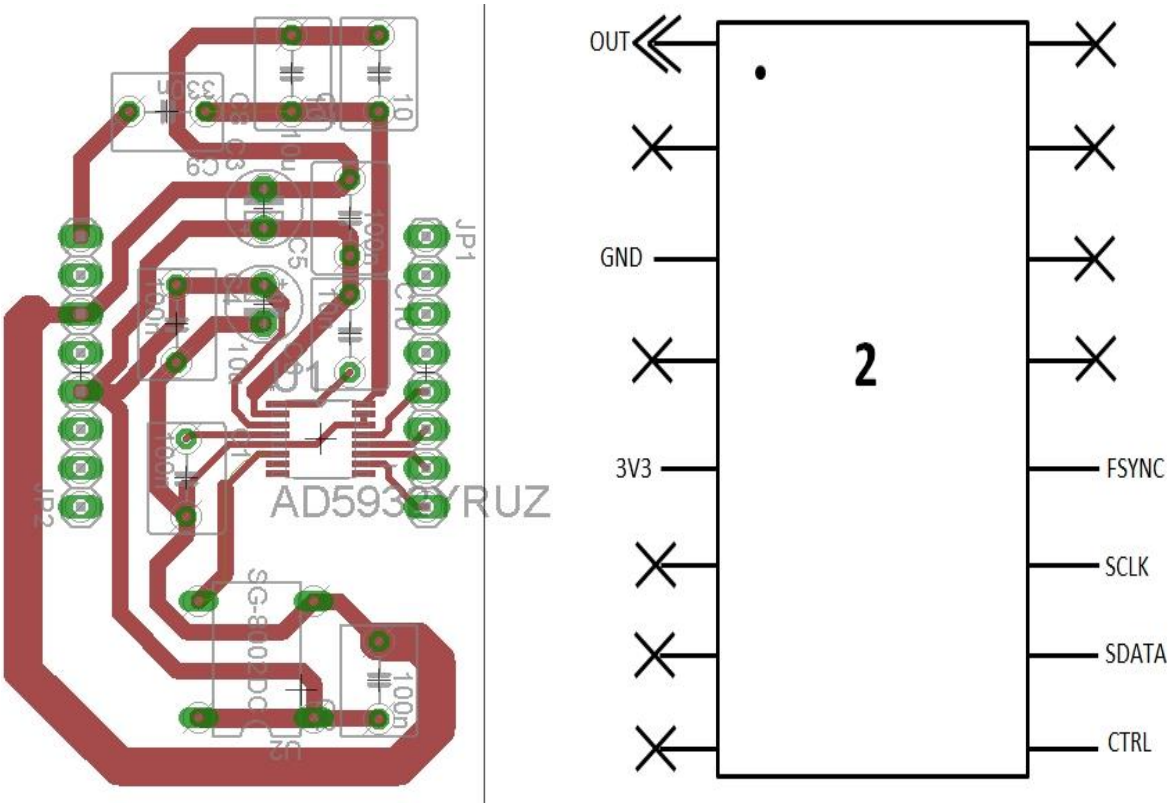
Napájecí napětí AVDD/DVDD	2.3-5.5V
Maximální proudový odběr (I_{DD}+I_{AA})	6.7mA
Hodinový signál MCLK	50 MHz
DDS-Min SNR	53 dB (MCLK=50MHz, fout=MCLK/4096)
DAC-rozlišení	10 bitů
DAC-vzorkovací frekvence	50 MHz
DAC-výstupní napětí fout	0.58Vpp
SPI- maximální frekvence SCLK	20 MHz
Tab.6 Základní parametry AD5932	



Obr.28 AD5932 pouzdro 16TSSOP



Obr.29 AD5932 rozložení pinů

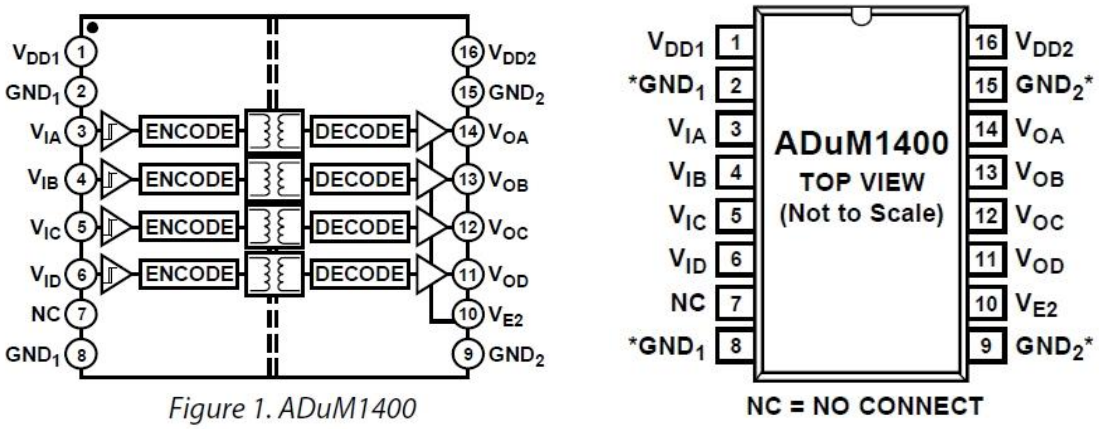


Obr.30 deska tištěných spojů s AD5932 s popisem pinů

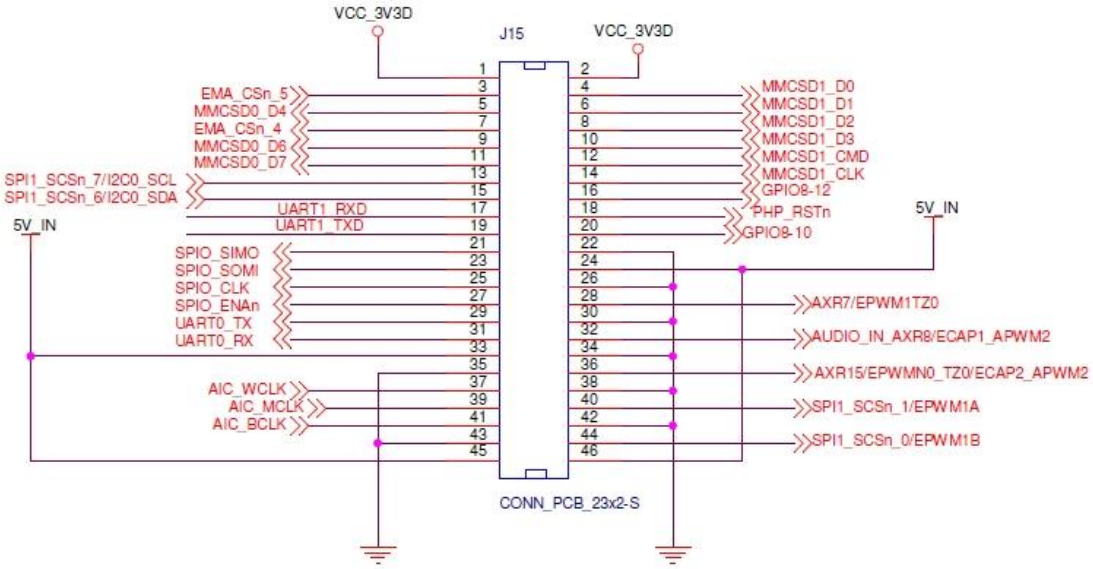
3.2.6 Galvanické oddělení

Pro ochranu přípravku jsem vložil mezi výstup z LCDK a obvod DDS galvanické oddělení, které je realizováno obvodem ADUM1400 viz datasheet [31], viz Obr.31.

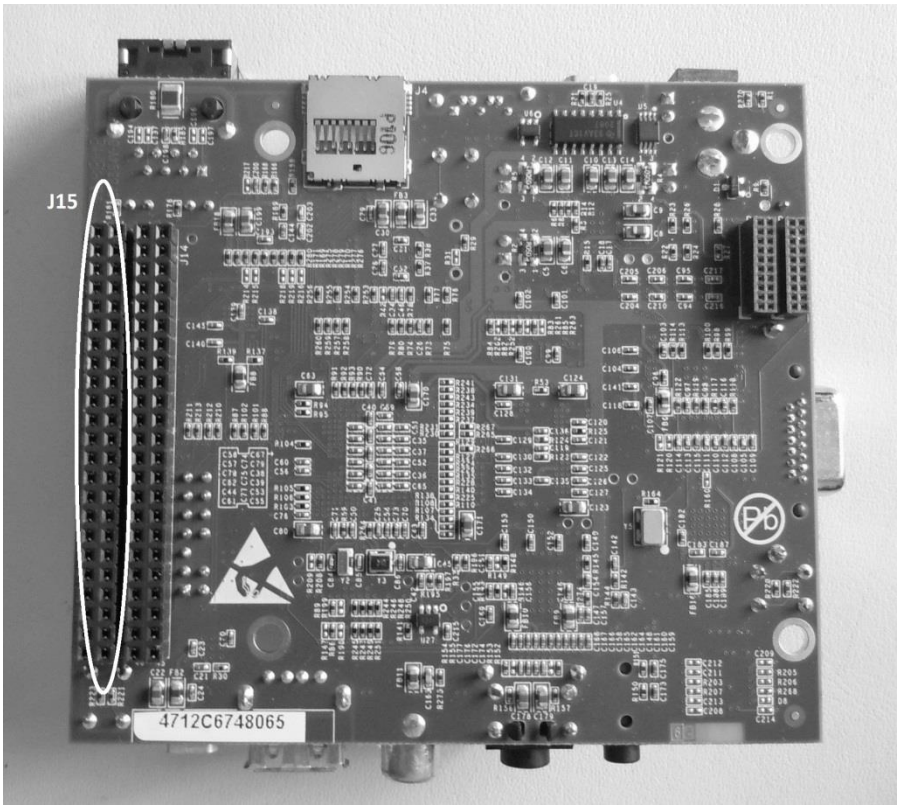
Vstupní bránu oddělovače bylo potřeba spojit se zemí a napájet napětím z přípravku (3,3V). Výstupní bránu bylo potřeba napájet 3,3V a spojit se společnou zemí zbytku rádiového přijímače. Na vstupní bránu jsou přivedeny řídící signály z konektoru J15 LCDK. Viz Obr. 32 a obr. 33. Signálové cesty jsou chráněny proti přepětí pomocí transilů SMBJV3V viz [32]. Pro tento obvod jsem navrhnul zvláštní tištěný spoj viz obr. 34, který obsahuje podpůrné součástky pro jeho bezproblémovou činnost. Návrhy tištěného spoje v Eaglu jsou k dispozici na přiloženém nosiči pod označením DP-BLOK1-E. Pro přehled použitých pinů J15, jejich funkci pro komunikaci s AD5932, označení ve schématu [3] a pinový multiplex přikládám tabulku 7.



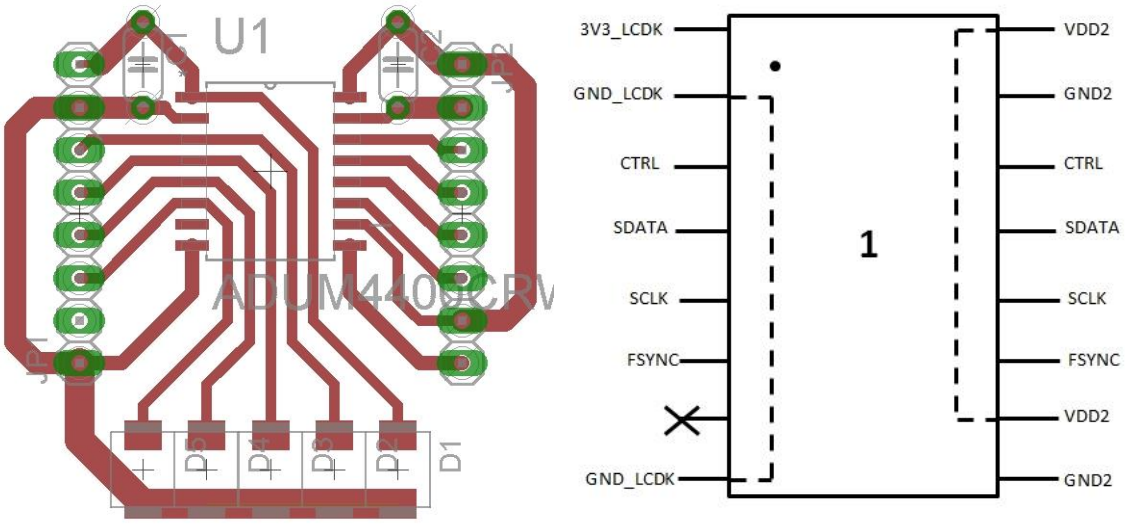
Obr. 31 blokové schéma a rozložení pinů galvanického oddělovače



Obr.32 konektor J15



Obr.33 konektor J15 na spodní straně LCDK

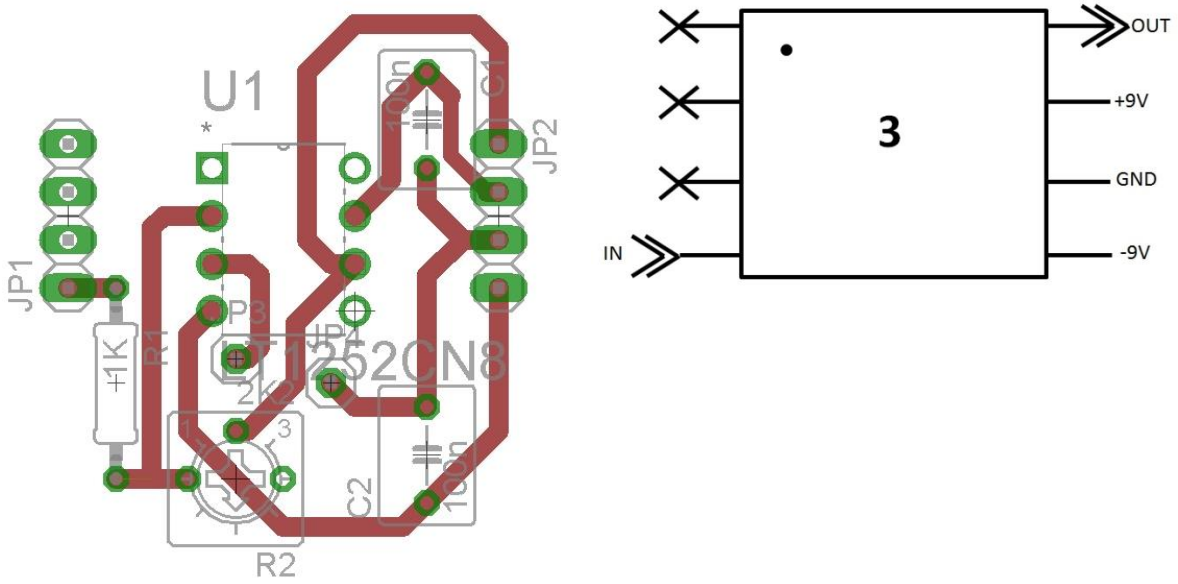


Obr.34 deska tištěných spojů s ADUM1400 s popisem pinů

#PIN	Označení- Schéma	Funkce	Prog
2	VCC_3V3D	3V3_LCDK	-
16	GPIO8_12	FSYNC	GPIO8_12
20	GPIO8_10	SCLK	GPIO8_10
26	GND	GND1	-
40	SPI1_SCSn_1	SDATA	GPIO2_15
44	SPI1_SCSn_0	CTRL	GPIO2_14
Tab.7 Souhrnný popis pinů konektoru J15			

3.2.7 Zesilovač demodulačního signálu

Pro možnost ručního nastavení zesílení demodulačního signálu z DDS obvodu byl navržen invertující zesilovač s vstupním odporem 1kΩ a 10kΩ trimrem ve zpětné vazbě. Byl použit OZ lt1252 od firmy Linear Technology. Pro tento obvod jsem navrhnul zvláštní tištěný spoj viz obr. 35, který obsahuje podpůrné součástky pro jeho bezproblémovou činnost. Návrhy tištěného spoje v Eaglu jsou k dispozici na přiloženém nosiči pod označením DP-BLOK3-D.



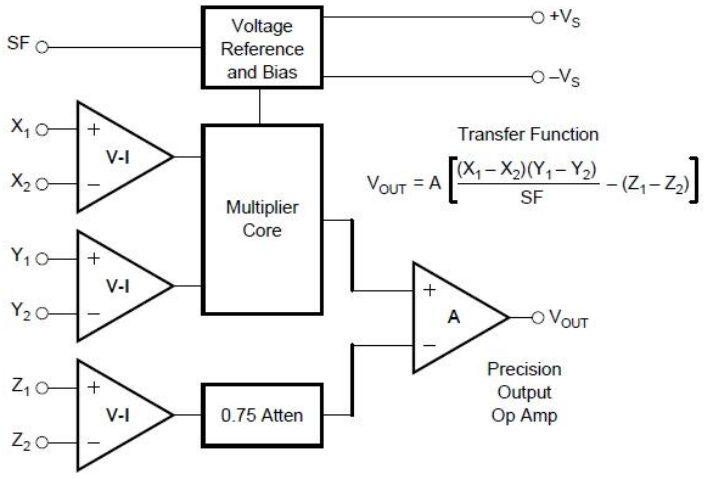
Obr.35 deska tištěných spojů s lt1252 s popisem pinů

3.2.8 Směšovač

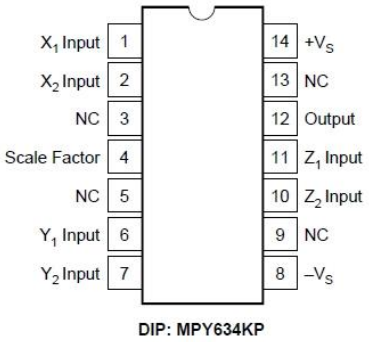
Pro realizaci demodulace popsané v sekci 3.1.3 byla použita čtyřkvadrantová analogová násobička MPY634 od firmy Texas instruments. Potřebné informace o jejím zapojení a parametrech jsem čerpal z [29]. Blokové schéma násobičky, včetně její přenosové funkce ukazuje obr. 36. Její základní parametry tabulka 8 a její rozložení pinů obr.37.

S MPY634 jsem také sestavil podle aplikačního schématu uvedeného v [29] lineární amplitudový modulátor viz Obr.38. Harmonickým signálem z mp3 přehrávače jsem moduloval signál generovaný DDS obvodem a výstupní napětí a v režimu FFT i modulové spektrum zobrazoval na osciloskopu TDS1001B. Ověřil jsem si tím vlastnosti popsané v sekci 3.1.2.

Poté jsem realizoval demodulátor S MPY634 v základním zapojení pro násobičku viz Obr.39, přičemž na vstup Y_1 jsem přivedl RF signál z RF zesilovače a na vstup X_1 zesílený signál z DDS. Aby bylo možné měnit velikost výstupního napětí V_{OUT} byl mezi piny $-V_S$ a SF zařazen trimer viz přenosová funkce na Obr.36. Hodnota SF se mění podle vztahu (3.9). kde R_{SF} je hodnota odporu mezi piny $-V_S$ a SF. Pro tento obvod jsem navrhnul zvláštní tištěný spoj viz obr. 40, který obsahuje podpůrné součástky pro jeho bezproblémovou činnost. Návrhy tištěného spoje v Eaglu jsou k dispozici na přiloženém nosiči pod označením DP-BLOK4-D.

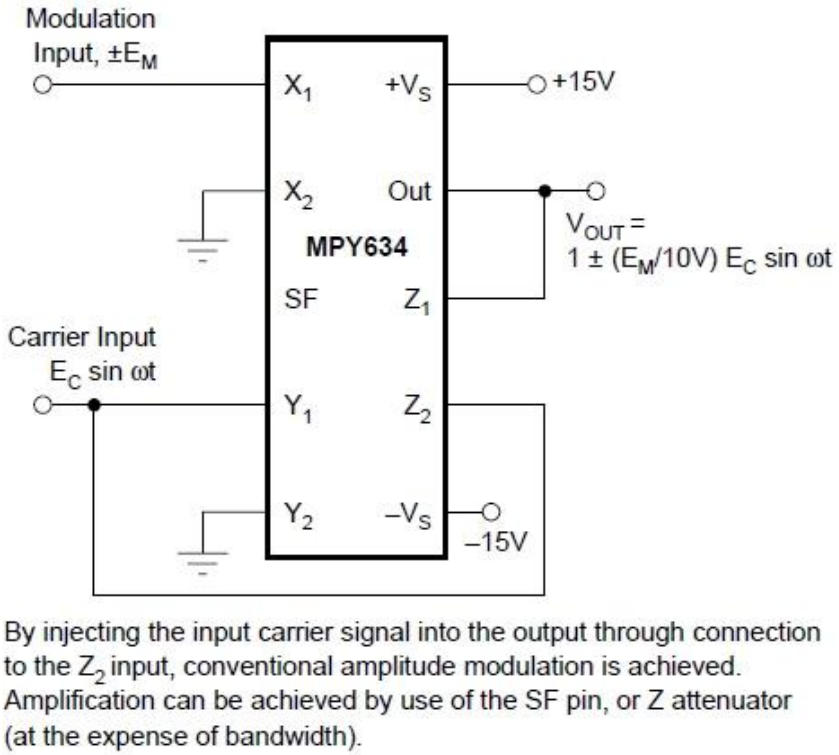


Obr.36 Blokové schéma MPY634 a její přenosová funkce

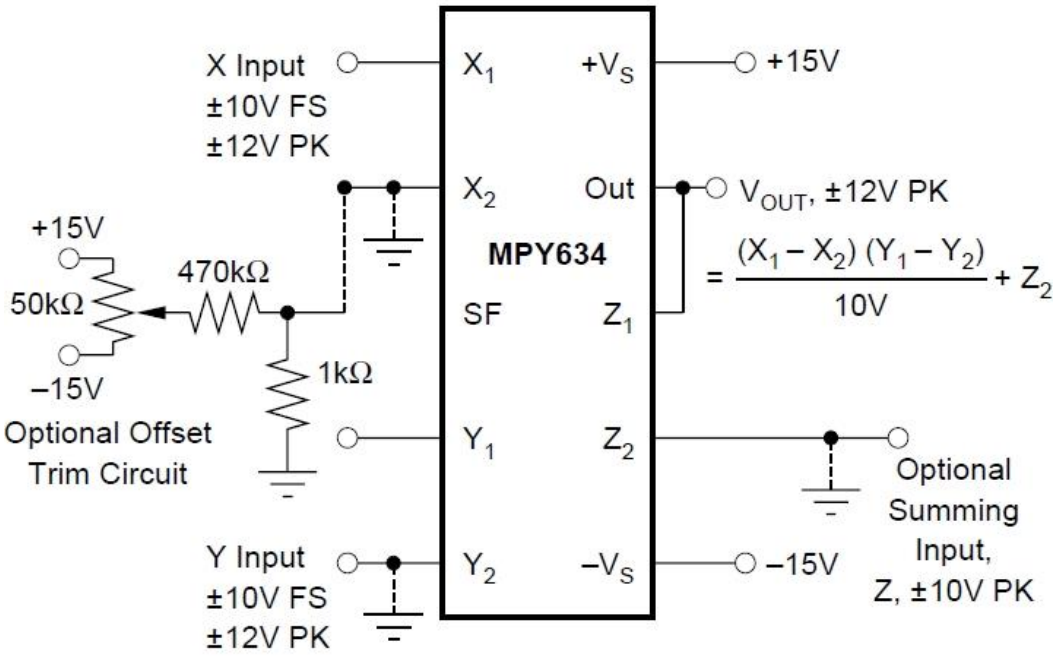


Obr.37 Rozložení pinů MPY

Napájecí napětí	+-9V
Minimální šířka pásma pro malé signály (0.1Vrms)	6MHz
Doba přeběhu	20uV/s
Rozkmit výstupního signálu	+-11V
Výstupní zkratový proud	30mA
Tab.8 Základní parametry MPY634	

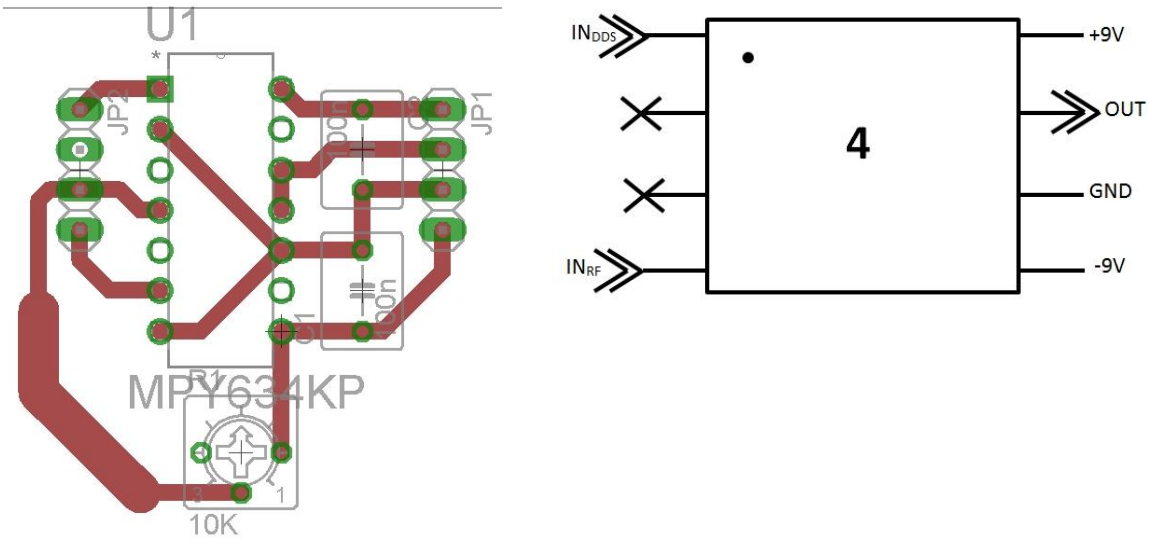


Obr.38 Zapojení MPY634 jako lineární amplitudový modulátor



Obr.39 Zapojení MPY634 jako amplitudový demodulátor

$$SF = 10 \cdot \frac{1}{1 + \frac{5{,}4k\Omega}{R_{SF}}} \tag{3.9}$$

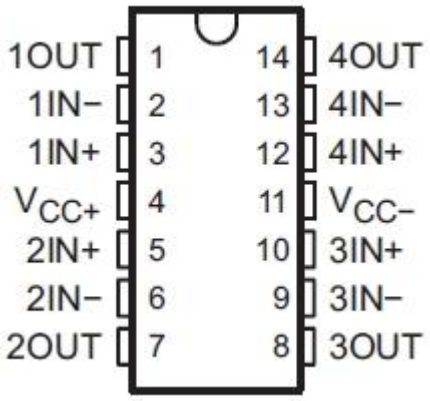


Obr.40 deska tištěných spojů s MPY634 s popisem pinů

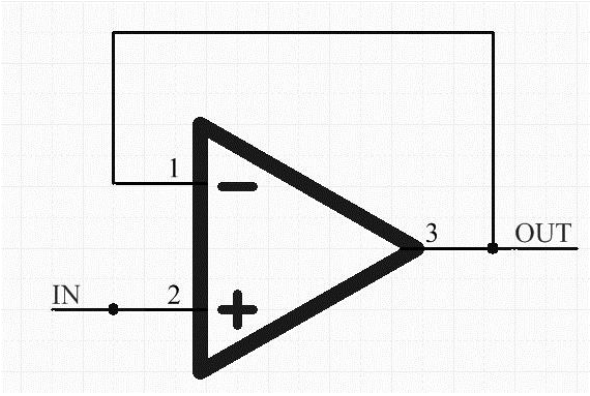
3.2.9 Filtr

Pro odfiltrování vysokofrekvenčních složek vzniklých po směšování jsem pomocí RLC online kalkulátoru vypočítal hodnoty LP filtru: L=680µH, C=0.68µF, R=62Ω pro zlomový kmitočet 7400Hz. Abych docílil větší strmosti filtru (-120dB) kaskádně jsem spojil tři takovéto RLC flitry a vzájemně jsem je oddělil napěťovými sledovači s OZ TL064CN viz. Obr. 41 a 42. Pro základní charakteristiky TL064CN viz tab.9. Pro více informací viz [35].

Pro odfiltrování ideálně téměř stejnosměrné složky (2.16) odvozené v teoretické části 3.1.3 jsem na výstupu TL064CN použil foliový vazební kondenzátor s kapacitou 470nF. Pro tento obvod jsem navrhnul zvláštní tištěný spoj viz obr. 43, který obsahuje podpůrné součástky pro jeho bezproblémovou činnost. Návrhy tištěného spoje v Eaglu jsou k dispozici na přiloženém nosiči pod označením DP-BLOK5-D.



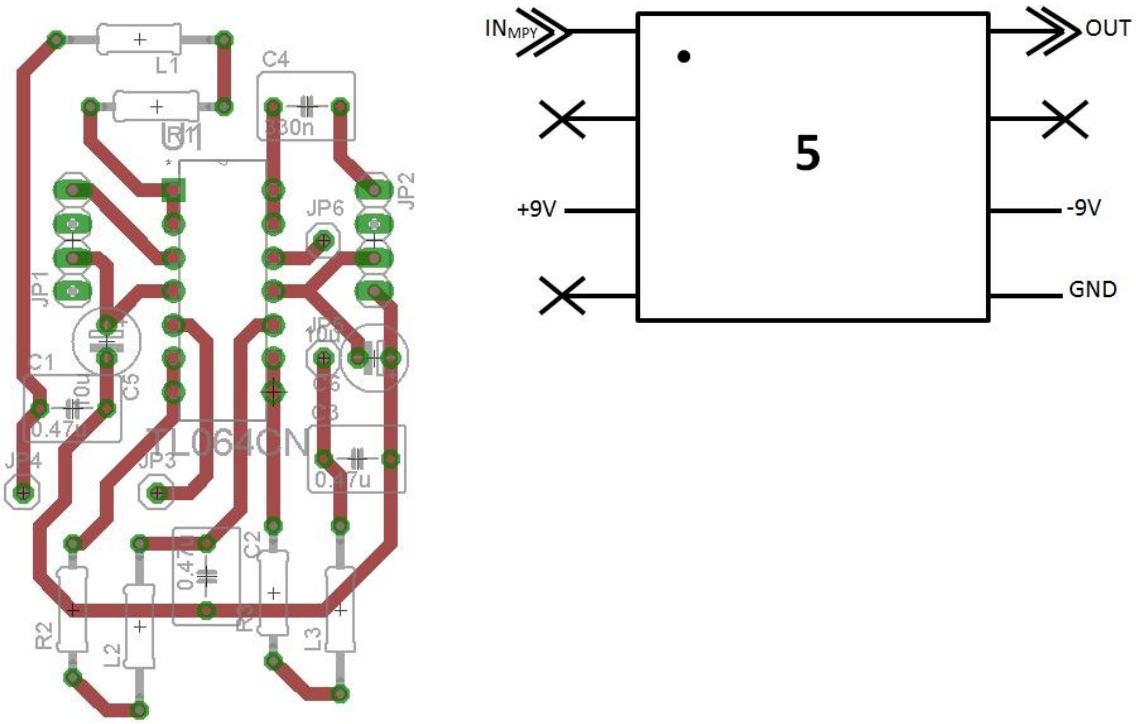
Obr.41 TL064CN rozložení pinů



Obr.42 Zapojení napěťového sledovače

Maximální napájecí napětí	+-18V
Maximální vstupní napětí	+-15V
Šířka pásma pro jednotkový zisk	1MHz
Minimální doba přeběhu	1.5V/us

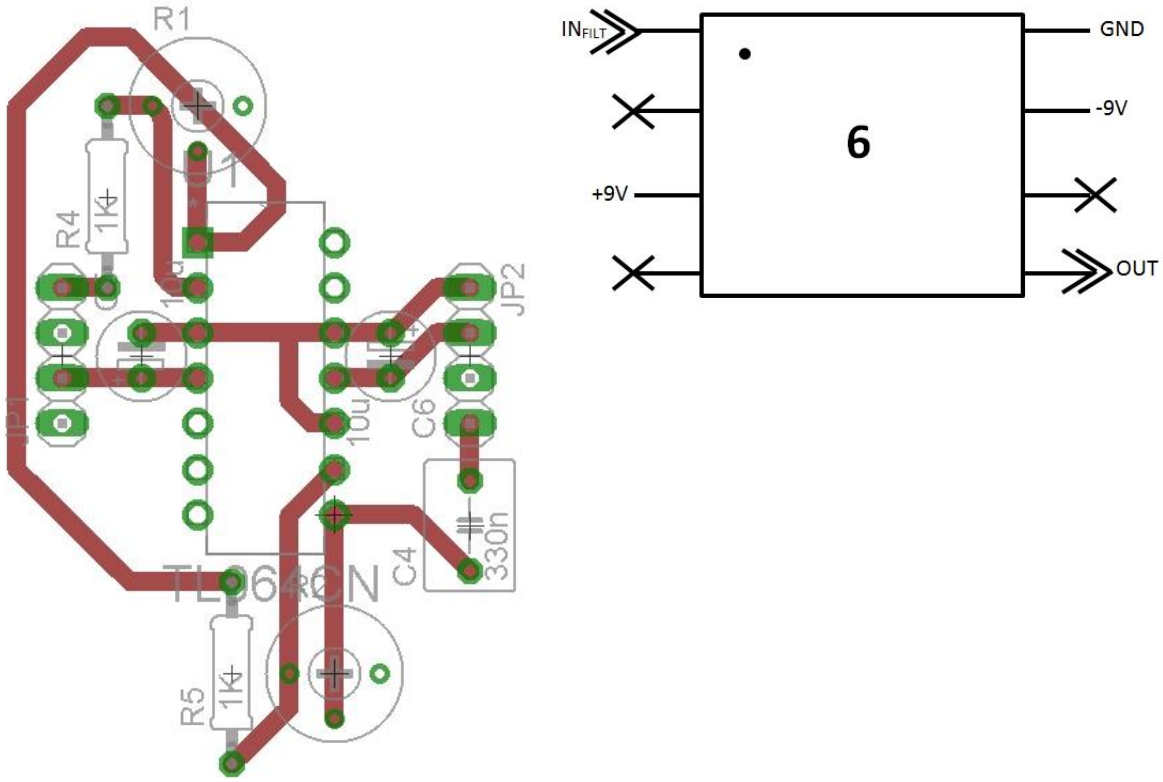
Tab.9 Základní parametry operačních zesilovačů TL064CN



Obr.43 deska tištěných spojů s TL064CN a RLC filtrem s popisem pinů

3.2.10 Audio zesilovač

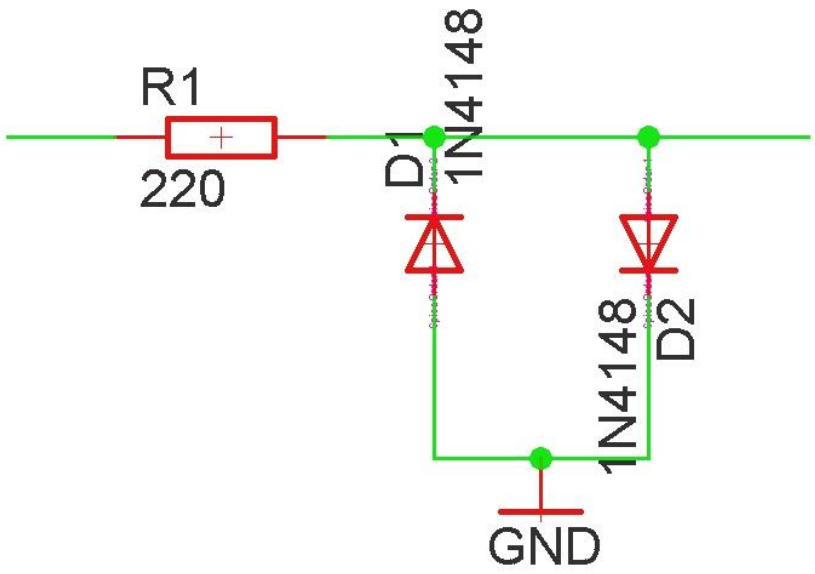
Jako audio zesilovač bylo použito kaskádní zapojení dvou invertujících zesilovačů s 1kΩ vstupními odpory a 25kΩ trimery ve zpětné vazbě. Jako OZ byl tak jako v případě filtru zvolen TL064CN od firmy Texas Instruments. Pro tento obvod jsem navrhnul zvláštní tištěný spoj viz obr. 44, který obsahuje podpůrné součástky pro jeho bezproblémovou činnost. Návrhy tištěného spoje v Eaglu jsou k dispozici na přiloženém nosiči pod označením DP-BLOK9-A.



Obr.44 deska tištěných spojů s TL064CN a zesilovači s popisem pinů

3.2.11 Omezovač

Pro ochranu audiokodeku na LCDK je za výstupem zesilovače zařazen diodový omezovač viz Obr. 45, který zaručuje maximální rozkmit výstupního signálu 1,4V_{pp}. Hodnota odporu 220Ω byla zvolena proto, aby při maximálním předpokládaném napětí 9V netekl diodami příliš velký proud. Tato hodnota zaručuje průtok proudu menšího než 40mA. Zároveň byla zvolena tak, aby byla zanedbatelná vzhledem k předpokládanému vstupnímu odporu audiokodeku 10kΩ. V zapojení byly použity diody 1N4148.



Obr.45 Diodový omezovač

3.2.12 Číslicové zpracování signálu

Pro detailnější popis cesty audio signálu v rámci LCDK viz Obr. 50 v úloze adaptivní filtrace. Pro ovládání hlasitosti jsou v rámci bloku číslicového zpracování signálu vzorky násobeny zvolenou konstantou. Je také možné podobně jako v úloze adaptivní filtrace provádět na vzorcích filtraci, např. FIR filtrem typu dolní propust se zlomovým kmitočtem 4kHz.

3.2.13 Napájení a uzemění

5V: -Vyžaduje LCDK, zajišťuje ho síťový adaptér dodávaný s přípravkem.

3,3V z LCDK:- Vyžaduje vstupní strana galvanického oddělení s ADUM1400 (BLOK1), je vyvedeno na zdířkách 1, 2 konektoru J15 viz [3] str. 12

3,3V ne z LCDK:- Vyžaduje výstupní strana galvanického oddělení s ADUM1400 (BLOK1), vyžaduje DDS obvod AD5932 (BLOK2), zajišťuje ho regulátor napětí LP2950 viz [36], který na toto napětí reguluje hlavní napájecí napětí 9V.

Jen +9V: - Vyžaduje ho RF zesilovač a pro regulaci regulátor na 3,3V , zajišťuje ho hlavní napájecí zdroj.

+9V a -9V:- Vyžadují ho zesilovač DDS signálu, směšovač, filtr a audio zesilovač, zajišťují je dva hlavní napájecí zdroje MW9115GS zapojené do série se společnými svorkami jako zemí.

Společná zem:- Společná zem je paprskovitě rozváděna do všech částí přijímače od společných svorek dvou hlavních napájecích zdrojů.

Ochranné zemění:- Zem napájecího adaptéru LCDK, zem USB kabelu, který propojuje počítač s JTAGem, společná zem přijímače, antistatická podložka, na které přijímač leží a ochranný antistatický pásek návrháře to vše je propojeno v jednom bodě, který je vodičem spojen s kolíkem zásuvkové řady, ve které jsou zapojeny všechny s přijímačem související elektrospotřebiče.

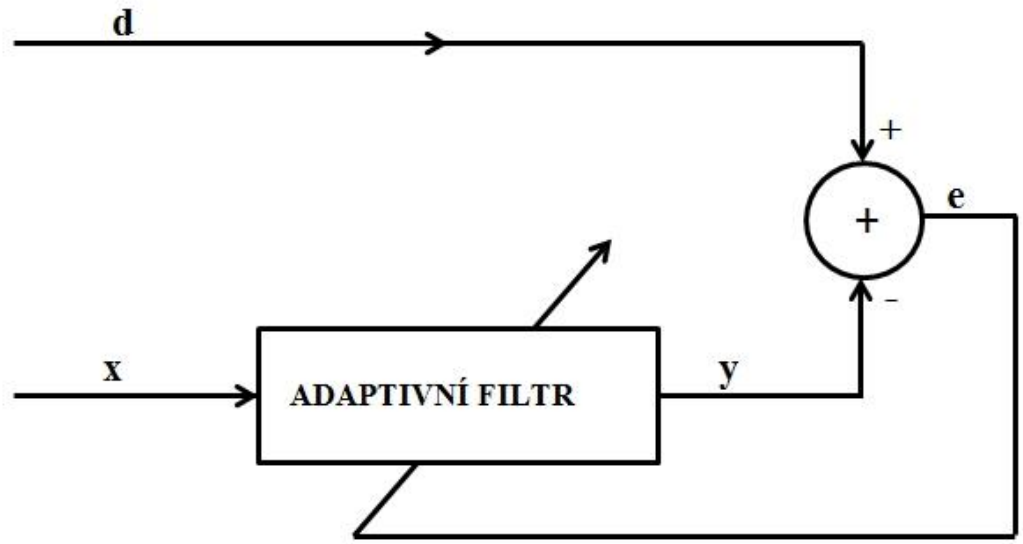
3.2.14 Zhodnocení kvality audio-výstupu přijatého signálu

Radiopřijímačem jsem přijímal stanici ČRo plus na frekvenci 639kHz AM vysílané z Českého Brodu viz část 3.1.5 Obr. 18. v průměrné kvalitě odpovídající AM příjmu.

4 Adaptivní filtrace

4.1 Teoretický popis

Na rozdíl od klasických FIR nebo IIR filtrů, které jsou navrhované s pevně danými parametry a mají tudíž pevně dané koeficienty jsou adaptivní filtry schopné reagovat na vstupní signál a pomocí adaptivního algoritmu měnit své koeficienty a tím se adaptovat na vstupní signál. Základní úlohu adaptivní filtrace, na které si popíšeme její principy lze blokově vyjádřit pomocí obrázku 46. Kde požadovaný - referenční signál d porovnáváme s výstupním signálem filtru y, který získáme filtrací vstupního signálu x. Pomocí chybového signálu e, potom nastavíme koeficienty filtru tak, abychom minimalizovali chybový signál e. V ideálním případě je chybový signál e nulový, koeficienty filtru se adaptovaly a už se nemění a filtrací vstupního signálu x obdržíme signál y, který je roven referenčnímu signálu d. Výhoda je, že potom můžeme použít úplně jiný vstupní signál x a filtr si upraví své koeficienty podle nového x. Aplikační úlohy adaptivní filtrace vycházejí z této základní struktury, liší se pouze interpretací jednotlivých signálů a dále určením vstupů a výstupů adaptivní filtrace.



Obr.46 Blokový diagram adaptivní filtrace

Při řešení této úlohy budeme používat filtr s konečnou impulzní odezvou (FIR) a to z důvodu jeho zaručené stability. Ve smyslu rychlejší konvergence, ale mohlo by také vést k oscilacím nebo k divergenci. Matematicky filtraci FIR filtrem popisuje konvoluce vstupního signálu s impulzní odezvou použitého FIR filtru. Impulzní odezva tak představuje jeho koeficienty. Předpis pro vztah mezi x a y dle obr.46 vyjadřuje vztah (3.1).

$$y(n) = \sum_{k=0}^{N-1} w_k\,(n)x(n-k) \qquad (3.1)$$

Výstupní signál y je součtem aktuálního vzorku s N-1 zpožděnými vzorky vstupního signálu x, které jsou váhovány (násobeny) příslušnými koeficienty w, platnými pro čas n. Rozdíl aktuálních vzorků referenčního signálu d a filtrovaného signálu y pro úplnost popisuje vztah (3.2)

$$e(n) = d(n) - y(n) \qquad (3.2)$$

K aktualizaci koeficientů byl použit algoritmus LMS(Least Mean Square), který pro následující koeficient minimalizuje velikost střední hodnoty druhé mocniny chybového signálu. Výsledný vztah ukazuje rovnice (3.3) pro detailní odvození viz [6] str. 257-261.

$$w_k(n+1) = w_k(n) + 2\beta e(n)x(n-k) \quad k = 0, 1, \dots N-1 \qquad (3.3)$$

Pro všechny koeficienty filtru tedy od 0 do N-1 jsou vypočteny nové koeficienty pro čas n+1 tak, že se k jejich stávající hodnotě přičte hodnota příslušného vzorku vstupního signálu váhovaná aktuální hodnotou chybového signálu. Velikost změny koeficientů a tudíž i rychlost a přesnost adaptace pak udává adaptační koeficient Beta.

Adaptivní filtrace nachází uplatnění v celé řadě aplikací, jako je potlačování ozvěn, ekvalizace, potlačení interference, potlačení rušení harmonickým signálem nebo v úloze lineární predikce. Pro více informací viz [7] kapitola 7, [9], [10]. Na LCDK jsem implementoval adaptivní filtr pro odstranění harmonického rušení z audio signálu. Struktura adpativního filtru pro tuto úlohu bude popsána v sekci implementace.

4.2 Implementace

4.2.1 Formulace úlohy

Implementujte adaptivní filtraci pro odstraňování harmonického rušení z audio signálu s použitím vývojového kitu TMS320C6748 LCDK.

4.2.2 Simulace algoritmu v prostředí MATLAB

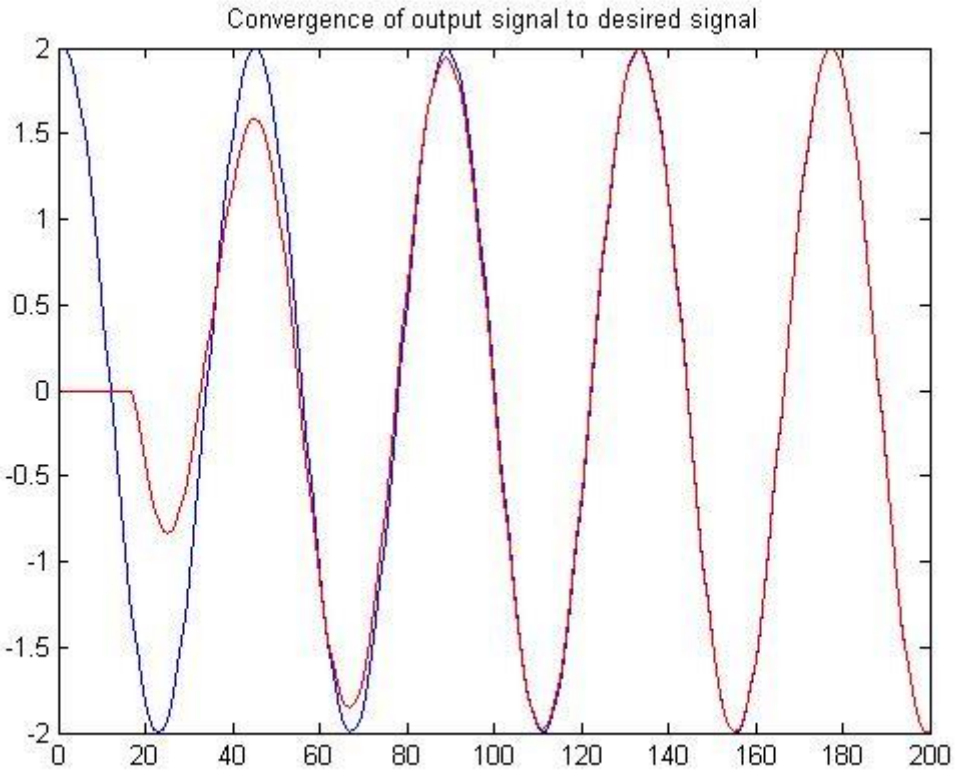
Po seznámení se s teorií potřebnou pro implementaci adaptivní filtrace jsem přikročil k její simulaci. Jako simulační program jsem využil MATLAB. Výhoda simulace spočívá v tom, že se zde nemusíme zabývat nastavováním registrů pro použité moduly procesoru, nemusíme řešit použité typy proměnných a můžeme se soustředit čistě na algoritmus. Přičemž si díky indexování a dobře vybaveným toolboxům můžeme zjednodušit a zpřehlednit kód. Další výhodou je fakt, že výsledky je možné jednoduše zobrazovat a také je možné jednoduše připravovat testovací vzorky. Ideálním výstupem simulace algoritmu by mohl být kód, napsaný v takové podobě, která by umožňovala přímý přepis do jazyka c bez použití specifické syntaxe MATLABu. V úloze adaptivní filtrace jsem ale k tomuto výstupu nedošel. Spíš jsem si jen ověřil funkčnost algoritmu, který jsem dolaďoval už přímo v CCS5.

Nejprve jsem simuloval vztah (3.1) tedy konvoluci s impulzní odezvou pro FIR s pevnými koeficienty.

Různé typy FIR filtrů jsem navrhoval jednak ve skriptu fir.m metodou okna použitím funkce FIR1(), jednak pomocí různých návrhových metod integrovaných do nástroje fdatool. Impulzové odezvy takto navržených FIR filtrů jsem pak používal v simulaci zjednodušeného vztahu (3.1) ve skriptu konvoluce.m.

Poté jsem přistoupil k simulaci základní struktury adaptivního filtru dle obrázku 46., kterou jsem odsimuloval ve skriptu Adapt_filt.m. Koeficienty adaptivního filtru se skutečně adaptovaly tak, že se vstupní harmonický signal po filtraci změnil v požadovaný referenční signal. Viz obr. 47

Po řadě pokusů jsem přistoupil k adaptivní filtraci aplikované na audio signal zarušený harmonickým rušením, viz. Adapt_filt5.m. Z této simulace jsem potom vycházel při implementaci algoritmu na LCDK.



Obr.47 Konvergence výstupního signálu(červeně) k referenčnímu signálu(modře)

4.2.3 Implementace na LCDK

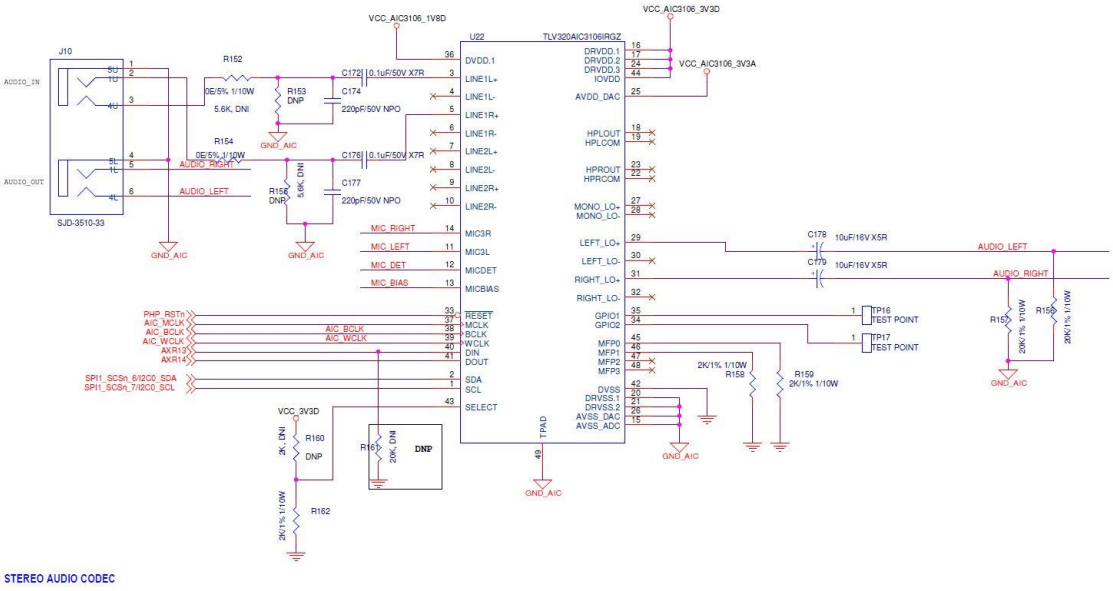
Nejprve jsem se v [3] zběžně seznámil s hardwareovým zapojením audia na LCDK a v [1] s použitými moduly C6748. Poté jsem využil příkladu „mcasp“ ze složky „examples“ z kolekce souborů StarterWare, který má usnadnit programátorům práci s C6748. StarterWare je ke stažení zde: [11]. Mým cílem v této části práce bylo zprovoznit přehrávání audia na LCDK, tedy zapisovat z vstupního bufferu na výstup, což již bylo

obsaženo v “mcaspPlayBk.c”, který je součástí příkladu “mcasp”. Hlavní problém tedy bylo nastavení CCS5 pro správný překlad a nahrání “mcasp” vzorového příkladu na LCDK. Pro bližší pochopení převzatého kódu bylo nutné pochopit jeho strukturu a získat povědomí o funkci a nastavení použitých modulů a audiokodeku a také o formátu vstupních a výstupních dat. Poté zbývalo takto fungující kód doplnit o algoritmus adaptivní filtrace a poslechově zhodnotit jeho kvalitu a možnosti použití na různě zarušených audionahrávkách připravených v MATLABu.

4.2.4 Hardwareové řešení audia na LCDK

Podle [3], strany 10, viz Obr.48 je pro audio použit stereoaudiokodek TLV320AIC3106, který je propojen s audio vstupem AUDIO_IN přes piny 3 (LINE1L+) a 5 (LINE1R+), s audio výstupem AUDIO_OUT přes piny 29 (LEFT_LO+) a 31 (RIGHT_LO+) s mikrofonem přes piny 14 (MIC3R), 11 (MIC3L), 12 (MICDET) a 13 (MICBIAS) s I2C megamodulem C6748 přes piny 2 (SDA) a 1 (SCL) a s McASP megamodulem přes piny 37 (MCLK), 38 (BCLK), 39 (WCLK), 40 (DIN), 41 (DOUT).

4.2.4.1 Stereoaudiokodek TLV320AIC3106

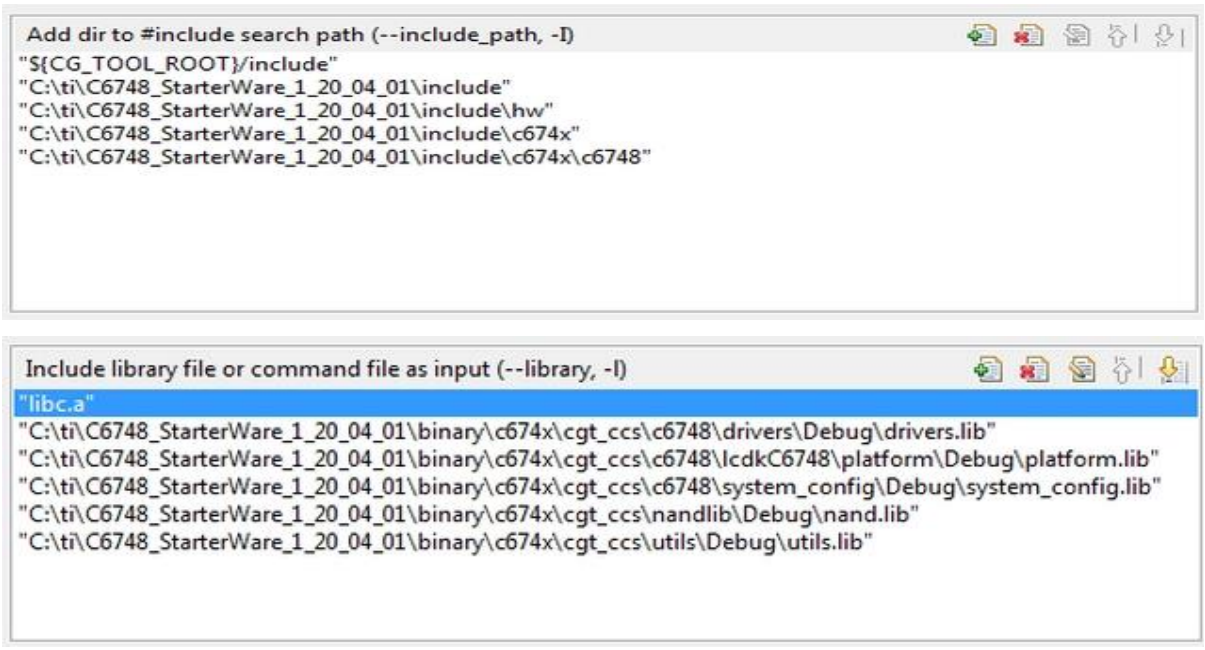


Obr.48 Zapojení stereoaudiokodeku na LCDK

4.2.5 Funkční spuštění „mcaspPlayBk.c“ na CCS5

Nejprve jsem si vytvořil nový workspace v souboru work, který jsem pojmenoval AUDIO_test4. Poté jsem otevřel CCS5 a vybral workspace AUDIO_test4. Poté jsem založil nový CCS projekt. (File=> New=> CCS Project) pojmenoval ho audio_test4, výstupní typ nechal „Executable“ zaškrtl checkbox „Use default location“ Vybral C6000 jako použitou rodinu mého procesoru. Z roletového menu vybral „LCDK6748“. Z roletového menu „Connection“ vybral způsob připojení LCDK k počítači tj. „Texas

Instruments XDS100v2 USB Emulator“. V sekci advanced settings jsem zvolil endianitu „little“ (je uvedená v [2]), vybral nejnovější compiler v mém případě TI v7.4.4, změnil výstupní formát na „eabi(ELF)“ Pro více informací o tomto rozhraní viz [13] odstavec 2.16 a 6.4. Dále je potřeba zvolit správný linker command file. Pro bližší informace viz [14] kapitola 7 Linker description. Zvolil jsem C6748.cmd. V menu „Runtime support library“ jsem nechal možnost „<automatic>“. V sekci „project templates and examples“ jsem zvolil „empty project“. Vše jsem potvrdil tlačítkem „finish“. Nyní se v „Project exploreru“ objevil prázdný projekt „audio_test4“ s inicializovaným compilerem a linkerem pro C6748LCDK. Nyní jsem přistoupil k importu příkladu „mcasp“ ze StarterWare. V „Project Exploreru“ jsem kliknul na projekt pravým tlačítkem a vybral „import“. Zvolil jsem „General“ potom „File system“ a zadal jsem cestu k souboru „mcasp“ v sekci examples u nainstalovaného StarterWare. Potvrdil jsem výběr všech přítomných částí kódu a spustil import tlačítkem „finish“. Nyní se do projektu naimportovaly potřebné soubory, ale je ještě potřeba nastavit cesty k hlavičkovým souborům u compileru a k použitým knihovnám u linkeru. Pro tato nastavení je potřeba pravým tlačítkem myši kliknout na projekt v „Project Exploreru“, rozkliknout „C6000 Compiler“ a „C6000 Linker“ a pro „Include Options“ a „File Search Path“ doplnit adresy k potřebným hlavičkovým souborům a knihovnám. Viz Obr.49. A nyní už stačí jen „postavit“ projekt a nahrát ho přes připojený JTAG do C6748. Aby se tak stalo, stačí kliknout na tlačítko s ikonou brouka. Nakonec je potřeba zapojit sluchátka/reproduktory do AUDIO_IN a mp3 přehrávač/AUDIO_OUT_LINE z PC do AUDIO_OUT viz Obr.48.

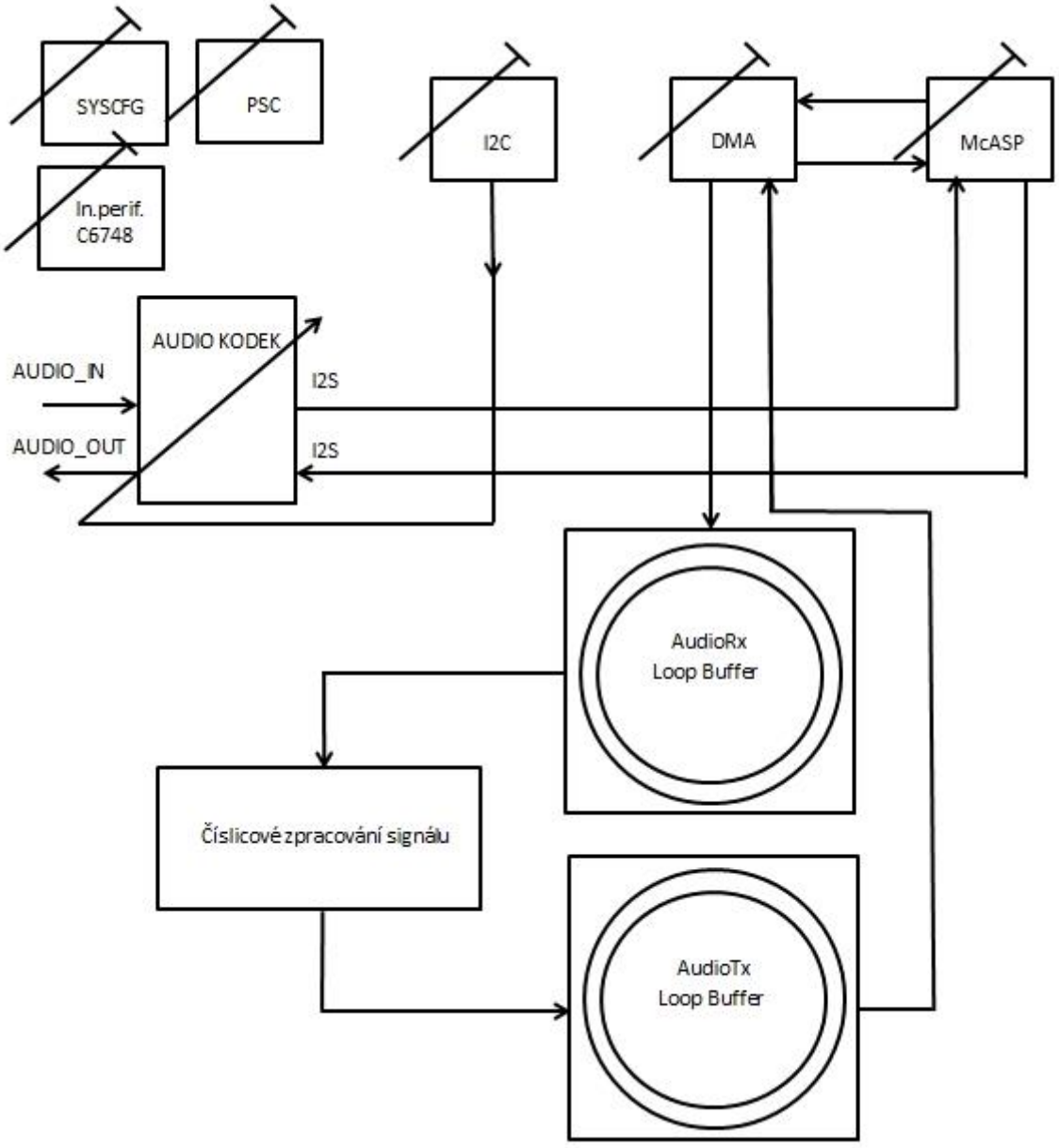


Obr.49 CCS5-Nastavení adres k hlavičkovým souborům a knihovnám, nahoře compiler dole linker

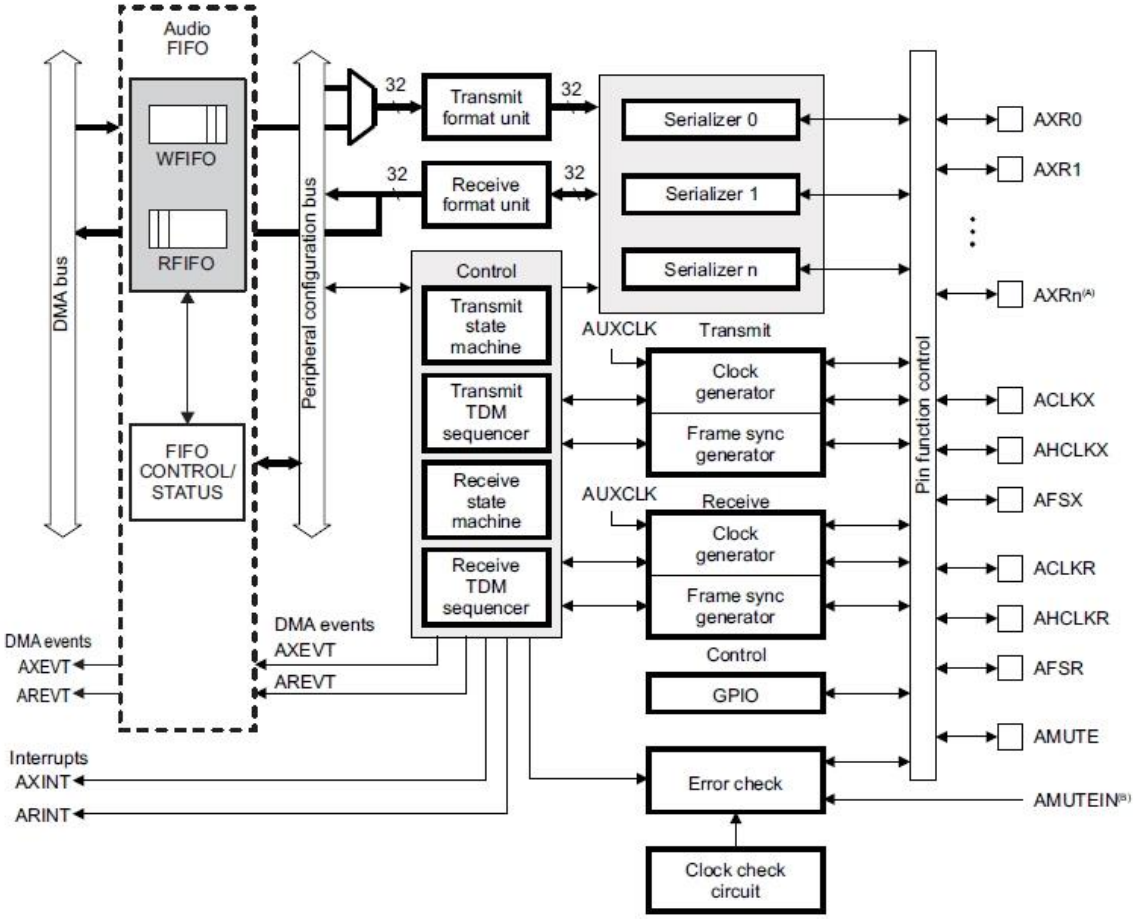
4.2.6 Struktura programu- průchod audio signálu

Obrázek 50. Vyjadřuje symbolicky strukturu programu, činnosti, které se v něm vykonávají a průchod audiosignálu od audio vstupu až po výstup. V prvním kroku je potřeba nastavit na obr. 50 uvedené moduly. Nastavení modulů i kodeku se provádí zápisem nastavovacích hodnot do příslušných registrů. V případě modulů je zápis prováděn prostřednictvím vnitřní infrastruktury procesoru v případě kodeku je zápis prováděn přes rozhraní I2C. Nejprve je potřeba vybrat správnou funkcionalitu pro piny modulů I2C a McASP. C6748 má totiž méně pinů než by bylo pro současnou činnost všech modulů potřeba, proto implementuje strategii sdílených pinů tzv. pinový multiplex. Z filosofie pinového multiplexu vyplývá, že ne všechny moduly lze použít současně, je potřeba se nejprve podívat do pinové mapy v dokumentu [3] na straně 23-27 nebo použít pomocný program pro pinový multiplex na C6748, který je přiložený k této práci, a který lze stáhnout zde [16]. Pinový multiplex a další systémové konfigurace se nastavují v modulu SYSCFG. Detailnější popis tohoto modulu viz [1] str. 205. Dále je potřeba „probudit“ potřebné moduly. C6748 má totiž implicitně z důvodu redukce spotřeby své moduly v klidovém režimu. Pro jejich aktivaci je třeba nastavit příslušné registry modulu PSC. Viz [1] str. 167. V dalším kroku inicializujeme řadič přerušení C6748, který patří k vnitřním periferiím C6748. Bližší informace o popisu a nastavení řadiče přerušení (INTC) viz [5] str.155. Poté inicializujeme modul I2C pro přenos dat do audiokodeku. I2C je dvouvodičové rozhraní typu otevřený konektor, umožňující propojení více zařízení typu MASTER se zařízeními typu SLAVE. Pro bližší informace o datovém přenosu a parametrech I2C viz [16]. Pro popis komunikačního protokolu I2C audiokodeku viz [12] strana 21. Na LCDK se rozhraní I2C využívá k nastavení audio a video kodeku. Pro detailní informace o I2C modulu a jeho nastavení viz [1] str.913. Inicializace představuje: reset modulu I2C, nastavení I2C pro standardní režim, tj. pro frekvenci SCL 100kHz, určení adresy SLAVE zařízení, kterým je v tomto případě kodek a registraci přerušovací rutiny pro I2C do vektoru přerušení modulu INTC. Dále se nastavuje modul DMA, McASP a audiokodek TLV320AIC3106. McASP je sériové rozhraní mezi kodekem a jádrem C6748 viz [1] str.1005. Jeho blokové schéma je na obrázku 51. V našem případě je potřeba nastavit pin AXR 13 jako vstupní, AXR14 jako výstupní, nastavit formát dat shodně s kodekem na I2S, nastavit rámcovou synchronizaci a hodinové signály v souladu s I2S, povolit audio FIFO pro DMA přenos dat a povolit serializéry . V kodeku jehož blokové schéma je na obr.10 nastavíme příslušné registry pro formát I2S, požadovanou vzorkovací frekvenci a inicializujeme ADC a DAC. Dále můžeme také využít možnosti bloku „Effects“ viz [12] str.35 a „Volume control“ viz[12] str.37. Modul DMA viz [1] str.499 se používá k přenosu dat mezi oběma kruhovými buffery a modulem McASP. Můžeme říci, že tyto kruhové buffery spoluutváří, o tom však níže. Pro použití DMA je potřeba nastavit parametry přenosu, které jsou organizovány do sad parametrů tzv. „PaRAM setů“ a ukládány do paměti RAM tzv. „PaRAM“. Pro strukturu PaRAM setu a popis jednotlivých parametrů viz [1] str. 511, pro příklad nastavení PaRAM setu pro kontinuální datové toky, což je případ McASP, viz [1] str. 553. Nejdůležitějšími parametry, které je třeba nastavit jsou: použité DMA kanály, ty jsou pro jednotlivé moduly

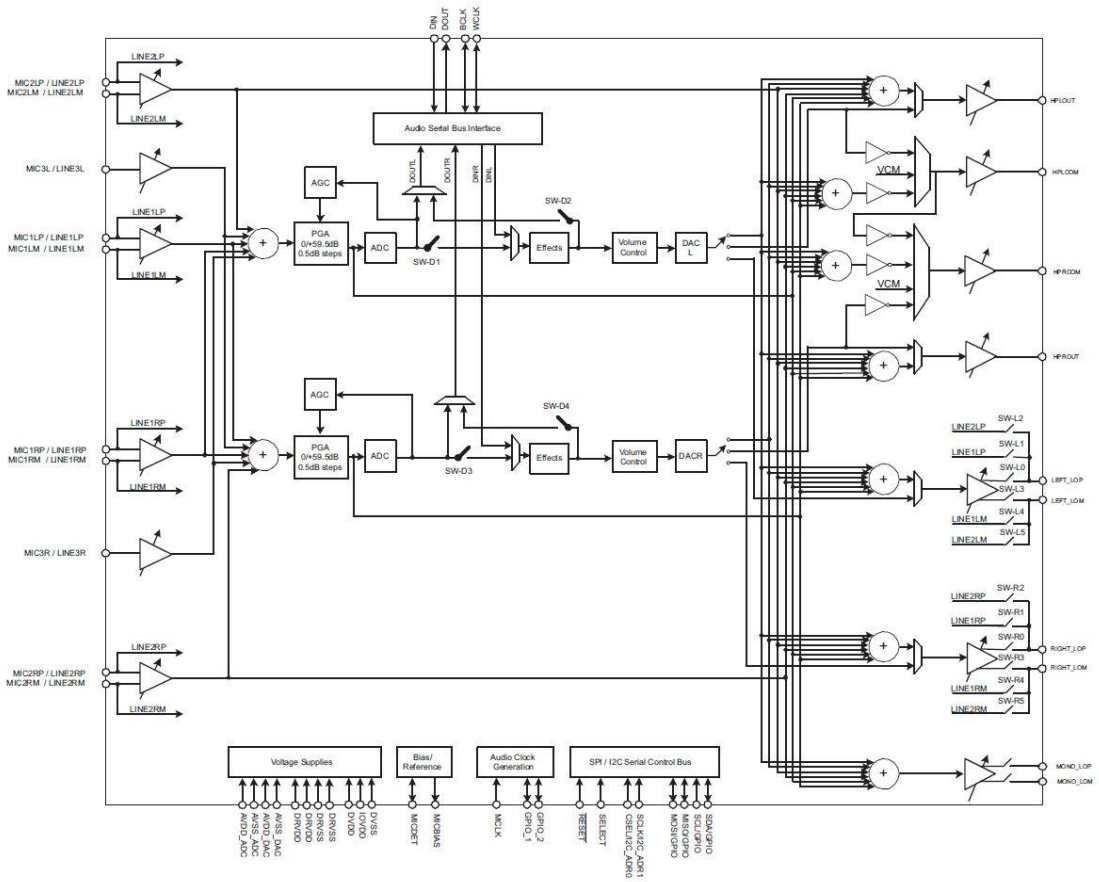
pevně dané, je potřeba se podívat do tabulky v [2] na straně 100, v tomto případě jde o kanály „0“ pro příjem a „1“ pro vysílání dat a parametry PaRAM setu. Kruhové buffery pro Rx a Tx jsou v tomto programu vytvořeny programovými přepínači rxBufPtr a txBufPtr, které postupně přepínají mezi jednotlivými buffery a změnou PaRAM setů kanálů DMA pro aktuálně zvolené buffery. V případě RX kruhového bufferu dochází ke změně PaRAM setu DMA kanálu „0“ podle aktuálně vybraného rx bufferu ve funkci BufferRxDMAActivate(), která je volána ve funkci McASPRxDMAComplHandler(), ve které se určí parametry nového PaRAM setu. Funkce McASPRxDMAComplHandler() je volána v obslužné rutině pro přerušení EDMA3CCComplIsr(). K tomuto přerušení dochází při ukončení DMA RX přenosu. V případě TX kruhového bufferu dochází ke změně PaRAM setu DMA kanálu „1“ podle aktuálně vybraného tx bufferu ve funkci BufferTxDMAActivate(), která je volána na konci if(), v těle while(1) během main(). Parametry nového PaRAM setu jsou spolu s aktuálním rx bufferem jsou určeny na začátku téhož if. Pro detailnější pohled na vztahy mezi buffery a PaRAM sety viz [17]. Mezi oběma kruhovými buffery je prostor pro číslicové zpracování audio signal. V příkladu ze Starterware je tento blok “zkratován” tím, že se data z RX buffer kopírují přímo do TX buffer.



Obr.50 Průchod audiosignálu



Obr.51 McASP-blokové schéma

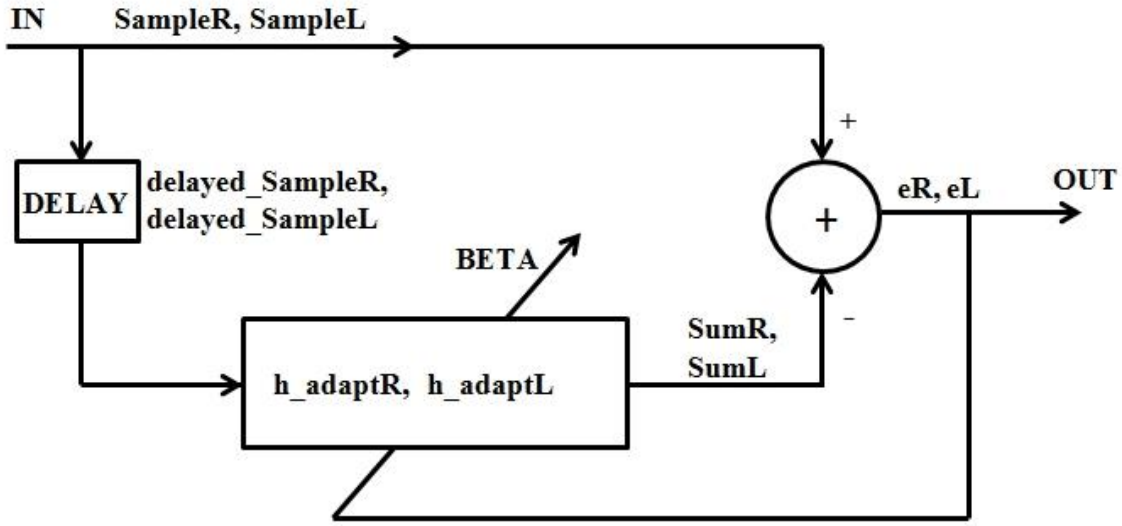


Obr.52 Blokové schéma audiokodeku TLV320AIC3106

4.2.7 Implementace algoritmu adaptivní filtrace

Vlastní implementace algoritmu adaptivní filtrace se odehrává v bloku „Číslicové zpracování signálu“ dle Obr. 50 a je znázorněna na Obr.53. Referenční signál tvoří vstupní audio data. Filtrovaný signál potom zpožděný vstupní signál. Výstupním signálem je pak rozdílový (chybový) signál, který by už neměl obsahovat rušivý signál. Vstupní data SampleX a delayed_SampleX jsou nahrány z RX kruhového bufferu pro oba audio kanály R a L, přičemž vstupní i výstupní audio data jsou znaménková 16 bitová. Na sudých pozicích I/O bufferů jsou uloženy data pravého, na lichých pak levého kanálu. Koeficienty FIR filtru h_adaptR a h_adaptL jsou na začátku nulové, ale to pro algoritmus není kritické. Můžeme jim zadat i nenulovou počáteční hodnotu. V algoritmu je nejprve implementována konvoluce a potom LMS adaptace koeficientů. Vzhledem k implementaci vstupního kruhového bufferu jako spínaných rx bufferů je nutné kvůli získání zpožděných vzorků a výpočtům konvoluce program rozdělit do dvou částí. Ta první řeší situaci, kde jsou k výpočtům potřeba data z předchozího RX bufferu. Tato situace nastává pro prvních HLEN-1+DELAY vzorků, kde HLEN-1 je počet koeficientů filtru pro předešlé vzorky a DELAY je rozdíl v indexech mezi zpožděným a aktuálním vzorkem. Pro zbývající vzorky

stačí pracovat s aktuálním bufferem. Algoritmus jsem nejprve napsal pro jeden kanál a pro vzorky ze stejného bufferu a potom jsem ho rozšířil pro oba kanály a všechny vzorky.



Obr.53 Implementace adaptivní filtrace pro odstranění harmonického signálu

4.2.8 Zhodnocení kvality výstupu adaptivní filtrace

Kvalitu výstupního signálu jsem testoval poslechem pohádky „O kohoutkovi a slepičce“, kterou jsem zarušil střídavě znějícími harmonickými signály. Tuto nahrávku jsem zvolil proto, že se jedná o mluvené slovo, s občasnými pomlkami, které je uvedeno a zakončeno hudbou. Kvalitu filtrace zásadně ovlivňuje hodnota parametru BETA. Pro velkou hodnotu BETA zněl filtrovaný výstup dost zkresleně. Pro malou hodnotu BETA byl výstupní signál od rušení dobře vyčištěný. Rušivé tóny ale zněly déle. Pro pozvolné změny rušivých tónů typu fade-in, fade-out to nevadilo. Problém byl s přechody mezi částmi nahrávky, ve které se náhle, skokově objevil rušivý tón o jiné frekvenci. Nakonec jsem jako kompromis zvolil pro BETU hodnotu 10^-11.

5 Závěr

V rámci diplomové práce jsem implementoval tři úlohy pro výuku aplikace algoritmů číslicového zpracování signálu na signálovém procesoru C6748 umístěném na vývojové desce LCDK. V úloze Cannyho hranové detekce jsem vytvořil funkční program, který ale neprobíhá v reálném čase. V úloze implementace AM radiopřijímače jsem přijímal ČRo na 639kHz v průměrné kvalitě. Úlohu adaptivní filtrace jsem zdárně dokončil, funguje a pan profesor Zahradník k ní neměl výhrady. Tato diplomová práce a hlavně spolupráce s panem profesorem Zahradníkem mě nesmírně obohatila. Získal jsem při ní zkušenosti jak s programováním, tak s návrhem a praktickým sestavováním obvodů. Chtěl bych mu touto cestou ještě jednou poděkovat: Děkuji!

6 Literatura

[1] *TMS320C6748 Technical Reference Manual*, Texas Instruments, Copyright © 2011, spruh79a

[2] *TMS320C6748 Fixed/Floating Point DSP*, Texas Instruments, Copyright © 2012, sprs590d

[3] *OMAP-L138_C6748 LC Dev Kit Ver A5*, Scheme, Texas Instruments, 2011,

[4] *TMS320C674x DSP CPU and Instruction Set Reference Guide*, Texas Instruments, Copyright © 2010, sprufe8b

[5] *TMS320C674x DSP Megamodule Reference Guide*, Texas Instruments, Copyright © 2010, sprufk5a

[6] *DSP Applications Using C and the TMS320C6x DSK*. Rulph Chassaing
Copyright © 2002 John Wiley & Sons, Inc.

ISBNs: 0-471-20754-3 (Hardback); 0-471-22112-0 (Electronic)

[7] *Digital Signal Processing and Applications with the C6713 and C6416 DSK*
Rulph Chassaing
Worcester Polytechnic Institute

[9] http://cdn.intechopen.com/pdfs-wm/16112.pdf

[10] http://www.dspalgorithms.com/aspt/asptnode26.html

[11] http://processors.wiki.ti.com/index.php/StarterWare

[12] *tlv320aic3106, LOW-POWER STEREO AUDIO CODEC FOR PORTABLE AUDIO/TELEPHONY*, Texas Instruments, Copyright © 2013

[13] *TMS320C6000 Optimizing Compiler v7.4 User's Guide*, Texas Instruments, Copyright © 2012, spru187u

[14] *TMS320C6000 Assembly Language Tools v7.4 User's Guide*, Texas Instruments, Copyright © 2012, spru186w

[15] http://processors.wiki.ti.com/index.php/StarterWare_01.20.01.01_User_Guide

[16] http://i2c.info/i2c-bus-specification

[17] http://processors.wiki.ti.com/index.php/StarterWare_Audio_Application

[18] *A Computational Approach to Edge Detection*, John Canny, 1986

[19] *Image Processing, Analysis and Machine Vision*, Milan Sonka, Václav Hlaváč, Roger Boyle, Copyright © 2008

[20] *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, International Telecommunication Union, 2013, ITU-R BT.601-7

[21] http://codesmesh.com/canny-edge-detection-with-matlab/

[22]http://www.mathworks.com/matlabcentral/fileexchange/30621-canny-edge-detection/content/canny.m

[23] SLES140G.pdf

[24] http://www.ti.com/tool/mathlib

[25] http://fyzika.jreichl.com/main.article/print/1389-amplitudova-modulace

[26] *Programmable Frequency Scan Waveform Generator*, AD5932, ANALOG DEVICES, Copyright © 2012

[27] *Programming the AD5932 for Frequency Sweep and Single Frequency Outputs*, Liam Riordan, ANALOG DEVICES, Copyright © 2012, Application note AN-1044

[28] *SPI Interface*, Miguel Usach, ANALOG DEVICES, Copyright © 2012, Application note AN-1248

[29] *Wide Bandwidth PRECISION ANALOG MULTIPLIER MPY634*, Texas Instruments, Copyright © 2012

[30] *50 MHz Direct Digital Synthesizer*, Waveform Generator AD9835, ANALOG DEVICES, Copyright © 2011

[31] *Quad-Channel Digital Isolators Data Sheet ADuM1400/ADuM1401/ADuM1402*, ANALOG DEVICES, Copyright © 2012

[32] *SMBJ3V3 Surface Mount TRANSZORB® Transient Voltage Suppressors*, Vishay General Semiconductor, 2011

[33] *LT1252 Low CostVideo Amplifier*, LINEAR TECHNOLOGY, Copyright © 1994

[34] http://sim.okawa-denshi.jp/en/RLCtool.php

[35] *TL06xx Low-Power JFET-Input Operational Amplifiers*, Texas Instruments, Copyright © 2014

[36] *ADJUSTABLE MICROPOWER VOLTAGE REGULATORS WITH SHUTDOWN*, lp2950, Copyright © 2014

7 Seznam použitých obrázků a jejich zdroje

Obr.1 Vývojový kit LCDK s procesorem C6748

[zdroj:Texas Instruments, http://processors.wiki.ti.com/index.php/L138/C6748_Development_Kit_%28LCDK%29]

Obr.2 Blokový diagram zapojení periferií s DSP na LCDK

[zdroj:Texas Instruments, OMAP-L138_C6748 LC Dev Kit Ver A5, strana 3]

Obr.3 Blokový diagram DSP

[zdroj: Texas Instruments, sprs590d, strana 6]

Obr.4 Gauss, 3x3, sigma=3, 8bitů

[zdroj: Petr Duga->MATLAB]

Obr.5 Gauss, 3x3, sigma=3, 8bitů

[zdroj: Petr Duga->MATLAB]

Obr.6 Gauss,3x3, sigma=0.3, 8bitů

[zdroj: Petr Duga->MATLAB]

Obr.7 DoGx a DoGy s parametrem sigma=0.5

[zdroj: Petr Duga->MATLAB]

Obr.8 DoGx, 3x3, sigma=3, rozsah +/- 255

[zdroj: Petr Duga-> MATLAB]

Obr.9 DoG,3x3, sigma=0.3, rozsah +/-255

[zdroj: Petr Duga-> MATLAB]

Obr.10 Určení sousedních dvou pixelů na základě úhlu gradientu

[zdroj: Petr Duga]

Obr.11 Porovnání hranových detektorů implementovaných v MATLABu ve funkci edge()

[zdroj: MATLAB, výstupy hranové detekce zobrazené funkcí imshow()]

Obr.12 CCS5-Nastavení adres k hlavičkovým souborům a knihovnám, nahoře compiler dole linker

[zdroj: CCS5->compiler, linker]

8 Seznam tabulek