Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

8 Laditelný generátor signálu

V této kapitole je uveden příklad vytvoření laditelného generátoru harmonického signálu. Pro tento účel se používá vývojový kit COM644KIT a přípravky EDAC a PANEL.

8.1 O přelaďování

V kapitole 6 byl uveden příklad vytvoření generátoru harmonického signálu. Tento příklad jednak vysvětloval základní problematiku programování mikrokontroléru ATmega644 a dále předvedl fyzická omezení.

Ve většině případů potřebujeme generátor signálu, který lze ladit. Tedy s možností nastavování kmitočtu.

Přelaďování hrubé – změna vzorkovacího kmitočtu

Základní myšlenka přelaďování vychází z toho, že prodloužíme průchod hlavní smyčkou. V kapitole 6 trval průchod hlavní smyčkou 11 strojních taktů procesoru. Pro uvažovaný příklad jsme pak dostali výsledný kmitočet zhruba 57 kHz.

Celkem jednoduše můžeme napočítat výsledné kmitočty pro dobu trvání jednoho průchodu smyčkou od 11 strojních taktů výše. Situace je shrnuta formou tabulky tab. 8.1. Sloupec **C** odpovídá trvání jednoho průchodu hlavní smyčkou 11 až 22 strojních taktů, $\mathbf{f_{SIG}}$ je výsledný kmitočet a $\mathbf{\delta f_{SIG}}$ je relativní změna kmitočtu vůči předchozímu naladění (rozdíl mezi naladěným kmitočtem a kmitočtem z předchozího řádku vztažený k naladěnému kmitočtu). Poslední sloupec tedy vlastně určuje „jemnost“ ladění.

Tab. 8.1 Hrubé ladění kmitočtu

C	f_{SIG}	δf_{SIG}
11	56 818,2	xxx
12	52 083,3	-9%
13	48 076,9	-8%
14	44 642,9	-8%
15	41 666,7	-7%
16	39 062,5	-7%
17	36 764,7	-6%
18	34 722,2	-6%
19	32 894,7	-6%
20	31 250,0	-5%
21	29 761,9	-5%
22	28 409, 1	-5%

Přelaďování jemné – decimace vzorků

Výše uvedený způsob neposkytuje příliš jemné ladění. Navíc se podle naladěného kmitočtu mění rozestupy mezi vzorky, tedy vzorkovací kmitočet. Takže rekonstrukční filtr se musí při změně generovaného kmitočtu přelaďovat také.

Jednou z možností promyšlenějšího ladění je decimace vzorků.

Základem této techniky je předpoklad, že máme k dispozici větší množství vzorků, než kolik chceme generovat. Z této posloupnosti pak uvažujeme pouze část.

Když například vezmeme pouze každý druhý vzorek, dosáhneme 2× vyššího kmitočtu generovaného signálu než v případě, že uvažujeme všechny vzorky.

Dále budeme uvažovat 32 768 vzorků (opět zvoleno jako mocnina čísla 2). Pokud trváme na podmínce, že v periodě generovaného signálu musí být alespoň 32 vzorků, lze přelaďovat v poměru 32 : 32 768. Tedy 1 : 1024.

Pro nejvyšší kmitočet uvažujeme 32 vzorků na periodu, pro kmitočet o krok nižší pak 33 vzorků na periodu atd. Můžeme tedy napočítat podobnou tabulku jako v případě hrubého ladění. Kmitočet signálu bude nyní normovaný (vztažený k maximální hodnotě), v této chvíli totiž nevíme, kolik strojních cyklů budeme potřebovat pro jeden průchod hlavní smyčkou.

Z tab. 8.2 je zřejmé, že ladění je nyní výrazně jemnější než v předchozím případě. Čili v naší realizaci se budeme věnovat tomuto způsobu přelaďování.

Tab. 8.2 Jemné ladění kmitočtu

N	$f_{SIG(norm.)}$	δf_{SIG}
32	1,000	xxx
33	0,970	-3%
34	0,941	-3%
35	0,914	-3%
36	0,889	-3%
37	0,865	-3%
38	0,842	-3%
39	0,821	-3%
40	0,800	-3%
41	0,780	-3%
42	0,762	-2%
43	0,744	-2%

8.2 Generování posloupnosti vzorků signálu

Pro případ použití 32 768 vzorků není možné vypočítávat jejich hodnoty ručně, rovněž použití programu Microsoft Excel není příliš výhodné.

Pro tento účel byl sestaven velmi jednoduchý program zapsaný v programovacím jazyce C++. Najdete jej na doprovodném CD-ROM v adresáři **SOFTWARE\GENSIN**.

Na tomto místě nemusíme program nikterak komentovat, jen použijeme výsledný soubor **SINUS.DAT**. Jeho forma umožní přímé vložení do zdrojového textu pomocí direktivy **.INCLUDE**. Každý řádek začíná direktivou .DB a následuje 8 vzorků signálu v rozlišení 8 bitů.

8.3 Hlavní smyčka

Hlavní smyčka generátoru signálu je dosti podobná prvnímu příkladu. Vzorky signálu uložené v paměti se budou číst pomocí ukazatele Z a odesílat na D/A převodník.

Nyní ovšem musíme upravit způsob výpočtu nové adresy vzorku. Jestliže provádíme decimaci vzorků, nebudeme obsah ukazatele Z zvyšovat vždy o 1, ale o určitý krok. Změnou kroku provádíme ladění.

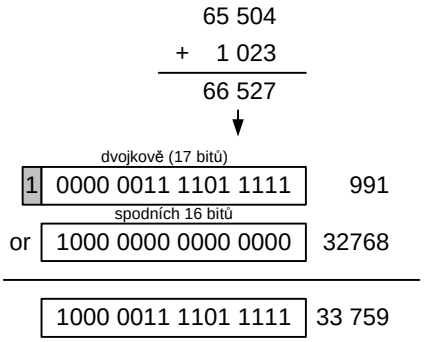
V následující realizaci uložíme vzorky ve Flash od adresy 32 768 (je to mocnina čísla 2). Jelikož Flash má kapacitu 64 KB, nevzniká žádný problém. Adresa prvního vzorku bude 32 768 a adresa posledního vzorku bude 65 535. Ukažme si, jak se má správně měnit obsah ukazatele Z pro případ, že krok bude 1023 (viz obr. 8.1).

Začneme pochopitelně adresou 32 768. Další vzorek vyzvedneme z adresy o 1023 vyšší, tedy 33 791. A tak pokračujeme dále.

Poslední platný vzorek (v pořadí třicátý třetí) má adresu 65 504. Po zvýšení adresy o 1023 dostaneme výsledek 66 527. Tato adresa je mimo rozsah. Opravu zajistíme odečtení čísla 65 536, což odpovídá oříznutí výsledku na spodních 16 bitů ve dvojkovém vyjádření.

Výsledná adresa je 991. Vzorky jsou ale uloženy od adresy 32 768. Čili správná adresa je dána součtem těchto dvou čísel (33 759).

č.	adresa v tabulce vzorků	
	desítkově	dvojkově
1.	32 768	1000 0000 0000 0000
2.	33 791	1000 0011 1111 1111
3.	34 814	1000 0111 1111 1110
33.	65 504	1111 1111 1110 0000
34.	33 759	1000 0011 1101 1111
35.	34 782	1000 0111 1101 1110



Obr. 8.1 Výpočet adresy v tabulce vzorků včetně korekce

Naznačený postup nebudeme realizovat doslovně. Jelikož mikrokontrolér pracuje ve dvojkové soustavě a počáteční adresa je mocninou čísla 2, lze vše zjednodušit:

- pokud po zvýšení obsahu ukazatele Z o KROK **nedojde** k přetečení (překročení čísla 65 535, které je i fyzickým maximem tohoto 16bitového ukazatele), je adresa v pořádku,
- pokud k přetečení dojde, stačí provést logický součet obsahu ukazatele Z s počáteční adresou. Jelikož číslo 32 768 vyjádřené ve dvojkové soustavě (1000 0000 0000 0000) má nižší bity nulové, stačí provést logický součet pouze mezi vyšší částí ukazatele Z (registrem ZH) a vyšší částí konstanty 32 768. Tedy použít instrukci ORI ZH,HIGH(32768).

Pro realizaci hlavní smyčky je třeba seznámit se s dalšími instrukcemi.

Stavový bit C

Přetečení rozsahu žádané operace indikuje bit C umístěný v registru SREG.

K přetečení může například dojít při součtu dvou 8bitových čísel. Je-li první číslo 10 a druhé 200, je výsledek 210 a C = 0 (nepřeteklo). Pokud je první číslo 10 a druhé 250, je výsledek 4 (tedy 260 – 256) a C = 1 (přeteklo).

Instrukce ADD

Instrukce ADD (jako ADDition) realizuje součet dvou registrů. Výsledek se uloží do prvního registru. V případě přetečení součtu je C = 1, jinak je C = 0.

Příklad: součet obsahů registrů ZL a XL, výsledek se uloží do ZL:

```
ADD ZL,XL
```

Instrukce ADC

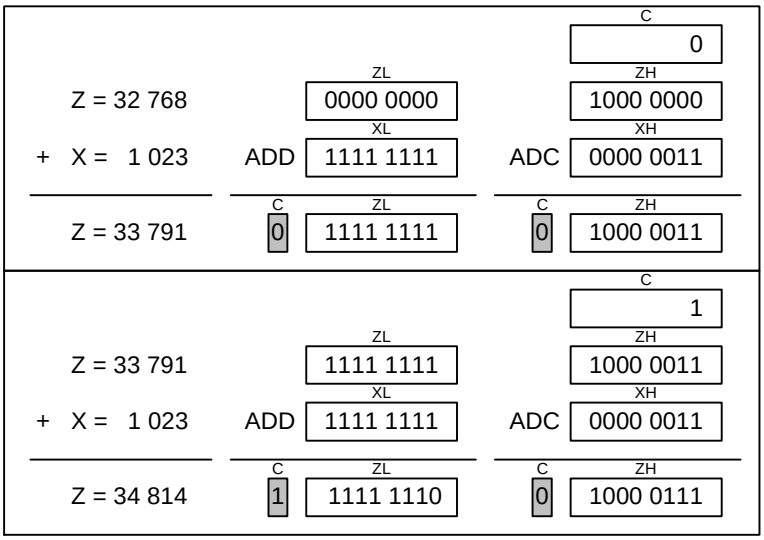
Instrukce ADC (jako ADdition with Carry) realizuje součet dvou registrů včetně přenosu. Tedy k součtu obsahů registrů se ještě přičte hodnota příznaku C před provedením této instrukce. Jinak se instrukce chová stejně jako ADD.

Instrukce ADC se používá pro realizaci součtů v rozsahu větším než 8 bitů. Například při sčítání registrů Z a X postupujeme takto: Nejdříve sečteme ZL a XL (nižších 8 bitů) instrukcí ADD, následně sečteme ZH a XH (vyšších 8 bitů včetně přenosu ze spodního součtu) instrukcí ADC.

Příklad součtu Z a X:

```
ADD  ZL,XL
ADC  ZH,XH
```

Pro lepší vysvětlení si ukažme efekt součtu pro různé hodnoty Z a X formou obr. 8.2. První součet ve spodních bitech nepřeteče, druhý ano.



Obr. 8.2 Realizace součtu Z a X

Instrukce ORI

Instrukce ORI (jako OR Immediate) provádí logický součet zvoleného registru s konstantou.

Pro dvě proměnné A a B platí, že výsledkem logického součtu je log. 1 v případě, že alespoň jedna proměnná má hodnotu log. 1. Platí pravdivostní tabulka:

vstupy		výstup
A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	1

Při programování mikrokontrolérů se častěji využívá tzv. *nastavovacího efektu logického součtu*. Představme si, že pro další běh programu potřebujeme zajistit nastavení nejvyššího bitu registru ZH. Jelikož log. 1 je v logickém součtu agresivní (dostane se vždy až na výstup), zvolíme dvojkovou masku 1000 0000 (ostatní bity se nezmění). To odpovídá instrukci `ORI ZH,0B10000000`.

Celá hlavní smyčka

Hlavní smyčka je realizována dříve popsaným způsobem. Ukazatel Z obsahuje adresu v tabulce vzorků, registr X obsahuje hodnotu kroku.

Instrukcí LPM se nahraje vzorek do registru R0 a odešle na port DAC.

Instrukce ADD a ADC zajistí zvýšení ukazatele Z o hodnotu KROK. Instrukce ORI provede korekci zpět na začátek tabulky.

```
        LDI XL,LOW(KROK)      ;krok
        LDI XH,HIGH(KROK)     ;v reg. X
        LDI ZL,LOW(ADRTAB)    ;adresa
        LDI ZH,HIGH(ADRTAB)   ;tabulky v Z
SMYCKA: SBI CTRL,WR           ;WR=1
        LPM R0,Z              ;cti data
        OUT DAC,R0            ;posli
        CBI CTRL,WR           ;WR=0
        ADD ZL,XL             ;zvys Z
        ADC ZH,XH             ;o KROK
        ORI ZH,HIGH(ADRTAB)   ;korekce
        RJMP SMYCKA           ;a znovu
```

Nyní tedy můžeme udělat rozbor délky trvání jednoho průchodu hlavní smyčkou. Nahlédnutím do tabulky instrukcí zjistíme, že výsledek je 13 taktů. Což odpovídá času 650 ns.

8.4 Přerušení

Uvedená realizace hlavní smyčky se pro urychlení nevěnuje přestavení kroku na hodnotu laděného kmitočtu. Jakákoliv další operace zapsaná do hlavní smyčky ji zpozdí a tedy sníží hodnotu generovaného kmitočtu.

Přestavení kmitočtu tedy bude vhodnější provádět pomocí přerušení.

Přerušení umožňuje zastavit vykonávání programu a zajistit obsluhu podnětu, který vyžaduje přednostní pozornost.

V níže uvedené realizaci budeme hodnotu kroku nastavovat dle hodnoty přijaté sériovou linkou (vyslanou přípravkem PANEL).

Hlavní smyčka bude generovat průběh nastaveného kmitočtu. Pokud uživatel bude žádat změnu kmitočtu, odešle ji pomocí přípravku PANEL sériovou linku. Při příjmu údaje se hlavní smyčka přeruší, obsluha sériové linky údaj uloží do registru X a vykonávání programu se vrátí zpět do hlavní smyčky. Tím dojde k přestavení generovaného kmitočtu.

Obsluhu přerušení zapisujeme na předem stanovenou adresu. Jako příklad si uveďme kostru obsluhy příjmu znaku od sériové linky:

```
        .ORG 0
        RJMP INIC
        .ORG URXC0addr
        RJMP PANEL

INIC:   ;inicializace

PANEL:  ;obsluha seriove linky
        RETI
```

Na adresu 0 umístíme skok na inicializaci procesoru. Na adresu danou symbolem **URXC0addr** umístíme skok na obsluhu sériové linky. Obsluha každého přerušení končí instrukcí **RETI** (návrat z přerušení).

8.5 USART0 – sériový kanál

Na tomto místě nemůžeme popsat celý sériový kanál, jelikož se jedná o poměrně komplexní periferii. Zaměříme se pouze na konfiguraci sériové linky pro příjem a vysvětlení zápisu obslužné rutiny.

Inicializace sériové linky probíhá takto. Do páru registrů **UBRR0H:UBRRL** zapíšeme požadovanou přenosovou rychlost (níže se používá symbol BAUD). Pomocí registru **UCSR0B** povolíme příjem sériovou linkou a přerušení po příjmu. Pomocí registru **UCSR0C** zvolíme rámec šířky 9 bitů. Rovněž musíme povolit přerušovací systém, k tomu slouží instrukce **SEI**.

```
LDI REG,HIGH(BAUD)                                ;nastav
STS UBRR0H,REG                                    ;prenos.
LDI REG,LOW(BAUD)                                 ;rychlost
STS UBRR0L,REG                                    ;9600 Bd
LDI REG,(1<<RXCIE0)|(1<<RXEN0)|(1<<UCSZ02)        ;povol
STS UCSR0B,REG                                    ;prijem+preruseni
LDI REG,(1<<UCSZ00)                               ;ramec
STS UCSR0C,REG                                    ;9 bitu
SEI
```

Všimněte si, že pro zápis do uvedených registrů nepoužíváme běžnou instrukci **OUT**. Problém je v tom, že tato instrukce je použitelná pouze pro některé registry periferií. Těchto registrů je u mikrokontroléru ATmega644 velké množství a registry pro řízení sériového portu již nejsou dostupné jinak, než jako buňky paměti. Tedy instrukcí **STS**.

Údaj přijatý sériovou linkou je uložen do registrů **UDR0** (spodních 8 bitů) a **UCSR0B** (nejvyšší bit, je v tomto registru na pozici bitu 0). Tyto registry nejsou pro čtení dostupné instrukcí **IN**, musí se použít instrukce **LDS**. Důležité je, že registr UCSR0B se musí číst dříve než UDR0.

8.6 Celý program

Celý program vznikne spojením jednotlivých dílčích částí, které byly diskutovány dříve.

Po resetu procesoru přejde vykonávání programu na návěští **INIC**. Zde probíhá inicializace. Nejdříve se porty PB a PD konfigurují jako výstupní (vývod RxD portu PD není řízen registrem DDRD, takže při inicializaci ho nemusíme uvažovat). Dále se nastaví pár SPH:SPL na konec datové paměti. Registry X a Z se nastaví na výchozí hodnotu kroku a na začátek tabulky vzorků. Potom se nastaví parametry sériového kanálu (přenosová rychlost 9600 Bd, 9bitový rámec, povolí se příjem a aktivace přerušení po příjmu bajtu). Nakonec se instrukcí **SEI** povolí přerušení.

Hlavní program je realizován od návěští **SMYCKA**. Tato smyčka pracuje dle výše popsaného principu. Z tabulky vzorků se načte údaj a vystaví na port, potvrdí se signálem WR. Následně se provede součet registrů Z a X a následná korekce instrukcí ORI.

Obsluha sériového portu je uvedena od návěští PANEL. Nejdříve je třeba přečíst registr UCSR0B a otestovat bit RXB80 (odpovídá 9. přijatému bitu). Je-li tento bit vynulován, byl přípravkem PANEL vyslán spodní bajt, ten se tedy uloží do registru XL. V opačném případě se jedná o horní bajt a ten je uložen do registru XH. Obsluha přerušení končí instrukcí RETI.

Na začátku a konci obsluhy přerušení najdeme instrukce IN a OUT, které manipulují s příznakovým registrem SREG. Na začátku obsluhy se stav SREG uloží do registru R1 a na konci obsluhy se stav SREG z tohoto registru obnoví.

9 Filtrace signálů

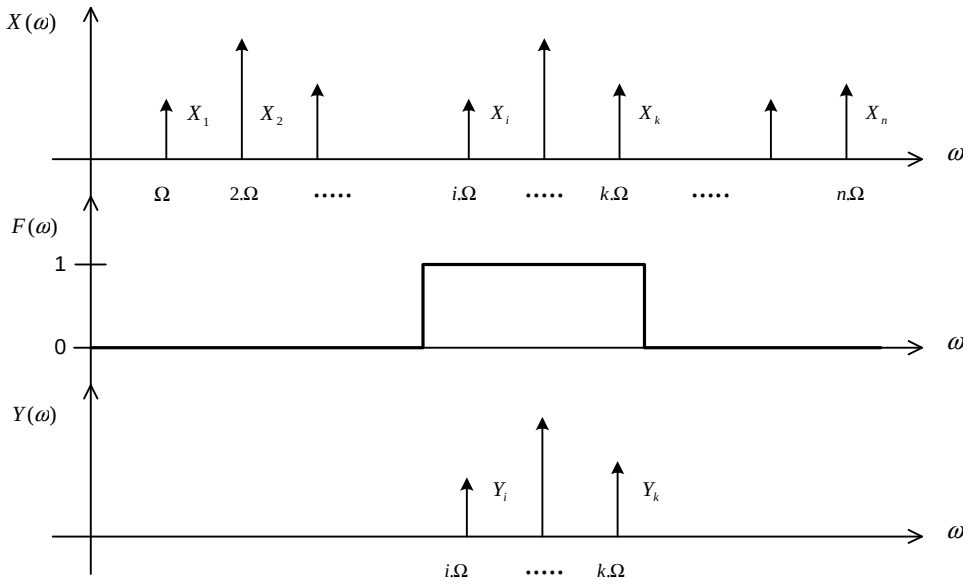
V následujících řádcích je velmi zjednodušeně popsán princip činnosti diskrétních filtrů. Nejprve je na základě ekvivalence popisu signálu časovým průběhem u(t) a spektrem u(f) uvedeno, jaký vliv má na signál tvaru obdélníkových impulsů obecný filtr typu dolní propust. V dalším je vysvětleno, jak téhož vlivu lze dosáhnout pouhým zpracováním impulsů jakožto vzorků tohoto signálu. Dále se rozebírá filtr typu dolní a horní propust a je zmíněn jev aliasingu.

Ke vzájemnému oddělení signálů, jejichž spektra se nepřekrývají slouží tzv. elektrické filtry, což jsou obvody v určitém pásmu kmitočtů elektrický signál propouštějící (tzv. propustné pásmo), a mimo toto pásmo ne (nebo s velkým útlumem). Má-li vstupní signál spektrum popsané spektrální funkcí $X(\omega)$ a spektrum výstupního signálu je $Y(\omega)$, pak filtr má přenosovou funkci $F(\omega)$, danou vztahem:

$$F(\omega) = \frac{Y(\omega)}{X(\omega)}$$

kde: $Y(\omega) = Y_i \cdot e^{ji\Omega t} + ... + Y_k \cdot e^{jk\Omega t}$,

$X(\omega) = X_1 \cdot e^{j\Omega t} + X_2 \cdot e^{j2\Omega t} + ... + X_i \cdot e^{ji\Omega t} + ... + X_k \cdot e^{jk\Omega t} + ... + X_n \cdot e^{jn\Omega t}$, viz obr. 9.1.



Obr. 9.1 K pojmu přenosová funkce filtru

V obr. 9.1 jsou X spektrální čáry, reprezentující amplitudy jednotlivých harmonických složek vstupního signálu a Y pak signálu výstupního.

Podle průběhu přenosové funkce $F(\omega)$ se filtry rozdělují na dolní, horní a pásmovou propust a pásmovou zádrž (průběh $F(\omega)$ na obr. 9.1 odpovídá ideální pásmové propusti), přičemž charakteristika skutečného (reálného) filtru tuto ideální charakteristiku jenom víceméně aproximuje.

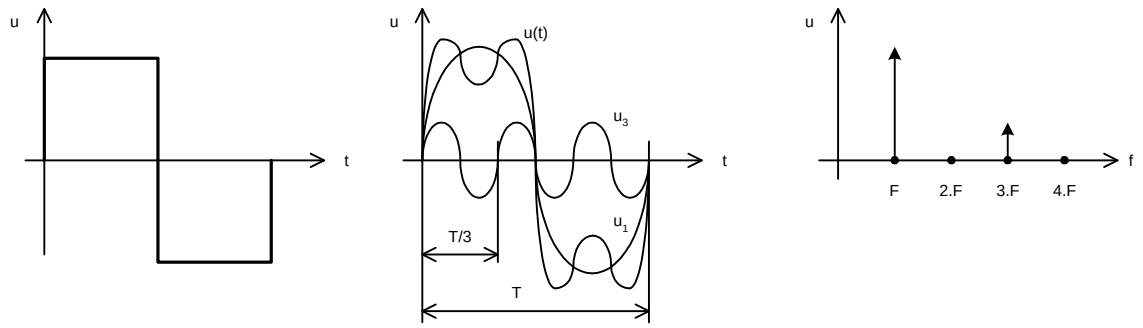
Kromě výše popsané kmitočtové závislosti přenosu napětí $F(\omega)$ je také velmi důležitou vlastností chování filtru pokud se vstupního napětí změní skokem. Ideální odezva filtru na takovýto tzv. jednotkový skok vstupního napětí, označovaný $1(t)$, který je definován vztahem:

$$1(t)\begin{cases} 0 & pro: t < 0 \\ 1 & pro: t > 0 \end{cases}$$

by pak měla co nejrychleji a také co nejvěrněji sledovat vstupní signál $1(t)$.

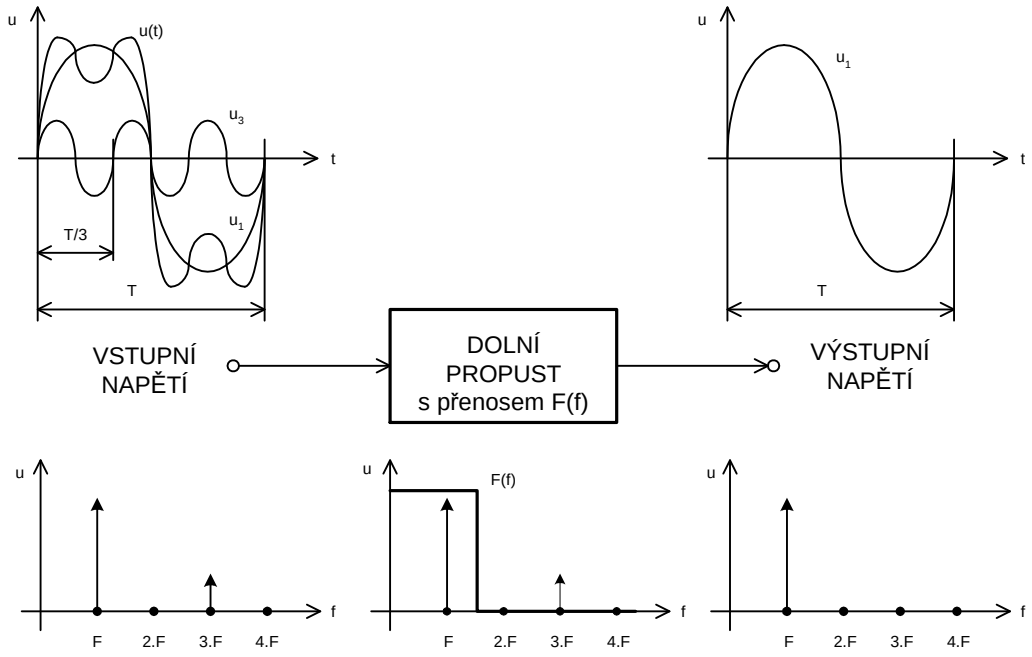
9.1 Znázornění činnosti analogového filtru typu dolní propust

Signál s průběhem blížícím se obdélníkovým impulsům si lze velmi zjednodušeně představit jakožto součet první u_1 a třetí u_3 harmonické podle vztahu $u(t) = u_1 + u_3$, tzn. $u(t) = U_1 \cdot \sin \Omega \cdot t + U_3 \cdot \sin 3 \cdot \Omega \cdot t$, takže jeho spektrum bude obsahovat dvě spektrální čáry: jednu s amplitudou U_1 na kmitočtu $\Omega = 2 \cdot \pi \cdot F$ a další s amplitudou U_3 na kmitočtu $3 \cdot \Omega = 3 \cdot 2 \cdot \pi \cdot F$, jak to znázorňuje obr. 9.2.



Obr. 9.2 Téměř obdélníkové impulsy lze rozložit na dva sinusové signály

Průchodem filtrem typu dolní propust (s mezním kmitočtem např. $\Omega_{MEZ} = 1{,}5.\Omega$) se toto spektrum „upraví“ tak, že filtr v něm potlačí složky s frekvencí větší, nežli Ω_{MEZ} na nový tvar, kterému však odpovídá již zcela jiný časový průběh, což ukazuje obr. 9.3.

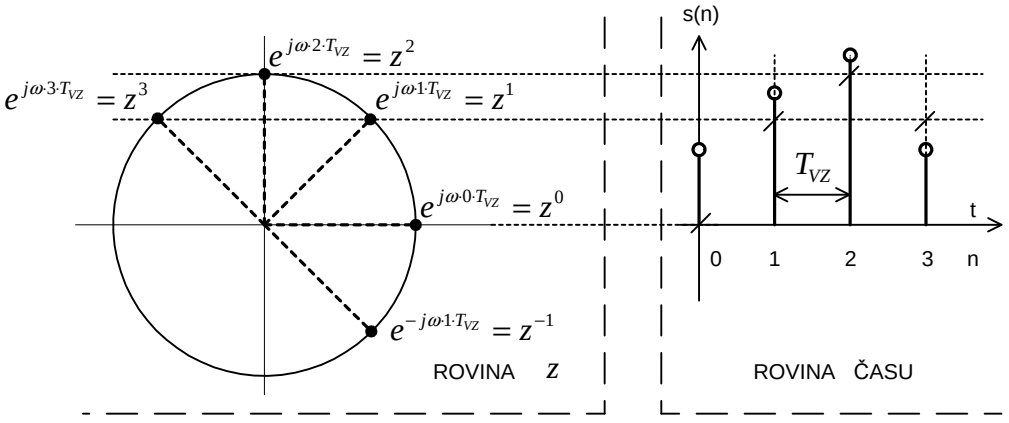


Obr. 9.3 Časové průběhy a spektrum na vstupu a výstupu dolní propusti

9.2 Znázornění činnosti číslicového filtru typu dolní propust

Například výše uvedenému (téměř) obdélníkovému signálu u odpovídá po průchodu dolní propustí sinusový signál s periodou odpovídající periodě signálu u, jehož lze však taktéž dosáhnout prostým přičítáním vzorků tohoto signálu, zpožděných o vzorkovací periodu T_{VZ}. Popsaný princip přibližuje graficky obr. 9.4.

odkud $s = \frac{1}{T_{VZ}} \cdot \ln z$. Tento logaritmus však vede v dalším na tzv. transcendentní funkce, pročež se nahrazuje nekonečnou řadou a podle počtu jejích členů se rozlišují různé aproximační transformace, jedna z nich je tzv. bilineární transformace.



Obr. 10.1 Souvislost roviny z a roviny času

Definiční vztah pro tuto bilineární transformaci lze snadno odvodit z řešení diferenciální rovnice $\frac{dy(t)}{dt} = x(t)$, která získá Laplaceovou transformací tvar $sY(s) = X(s)$, odkud přenosová funkce $F(s)$ je poměr $F(s) = \frac{Y(s)}{X(s)} = \frac{1}{s}$.

Vyjádří-li se hodnota $x(t)$ jako průměr dvou sousedních hodnot $x(n)$ a $x(n-1)$, tj. jako $\frac{x(n) + x(n-1)}{2}$, pak rovnici $\frac{dy(t)}{dt} = x(t)$ lze přepsat takto:

$$\frac{y(n) - y(n-1)}{T} = \frac{x(n) + x(n-1)}{2}.$$

Aplikací Z-transformace bude

$$\frac{Y(z) - Y(z) \cdot z^{-1}}{T} = \frac{X(z) + X(z) \cdot z^{-1}}{2}$$

$$\frac{Y(z) \cdot (1 - z^{-1})}{T} = \frac{X(z) \cdot (1 + z^{-1})}{2}$$

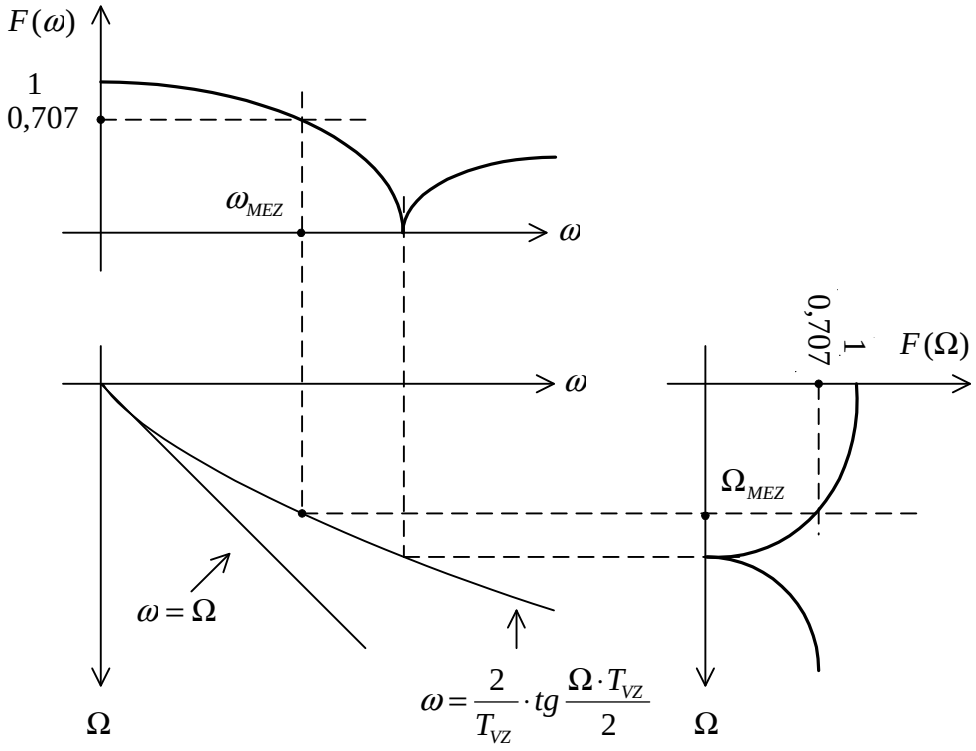
odkud přenosová funkce $F(z)$ je poměr $F(z) = \frac{Y(z)}{X(z)} = \frac{1}{\frac{2}{T_{VZ}} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}}$, kde $T = T_{VZ}$.

Konečně porovnáním vztahů pro $F(s)$ a $F(z)$ plyne, že definiční vztah bilineární transformace má tvar:

$$s = \frac{2}{T_{VZ}} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}.$$

Bilineární transformace zobrazuje celou (nekonečně dlouhou) osu kmitočtu ω na kružnici konečné délky, reprezentující kmitočet Ω v oblasti z (jak je ostatně naznačeno i v obr. 10.1), vztah mezi těmito kmitočty lze získat dosazením do definičního vztahu, po dosazení bude mít tvar $j\omega = \frac{2}{T_{VZ}} \cdot \frac{1 - e^{-j\Omega \cdot T_{VZ}}}{1 + e^{-j\Omega \cdot T_{VZ}}}$, odkud

$\omega = \frac{2}{T_{VZ}} \cdot tg \frac{\Omega \cdot T_{VZ}}{2}$, resp. $\Omega = \frac{2}{T_{VZ}} \cdot arctg \frac{\omega \cdot T_{VZ}}{2}$, což vyjadřuje vlastně kompresi přímky na kružnici. Toto zkreslení os kmitočtů dané výše uvedenými vztahy ukazuje názorně na příkladu dolní propusti obr. 10.2.



Obr. 10.2 Zkreslení os kmitočtů (pro dolní propust)

10.2 Metoda rozmístění nulových bodů a pólů

Nejjednodušší metodou návrhu číslicového filtru je metoda (prostého) rozmísťování nulových bodů a pólů přenosové funkce $F(z)$, kterou lze rozepsat následovně ve formě podílu polynomů čitatele a jmenovatele:

$$F(z) = \frac{a_o + a_1 \cdot z^{-1} + ... + a_M \cdot z^{-M}}{1 + b_1 \cdot z^{-1} + ... + b_N \cdot z^{-N}} = \frac{\sum_{k=0}^{k=M} a_k \cdot z^{-k}}{1 + \sum_{l=0}^{l=N} b_l \cdot z^{-l}}$$

Protože záporné mocniny z lze snadno převést na mocniny z kladné (např. takto):

$$\frac{a_o + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}}{1 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}} = \frac{a_o + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}}{1 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}} \cdot \frac{z^2}{z^2} = \frac{a_o \cdot z^2 + a_1 \cdot z^{-1} \cdot z^2 + a_2 \cdot z^{-2} \cdot z^2}{1 \cdot z^2 + b_1 \cdot z^{-1} \cdot z^2 + b_2 \cdot z^{-2} \cdot z^2} =$$

$$= \frac{a_o \cdot z^2 + a_1 \cdot z^1 + a_2 \cdot z^0}{z^2 + b_1 \cdot z^1 + b_2 \cdot z^0} = \frac{a_o \cdot z^2 + a_1 \cdot z^1 + a_2}{z^2 + b_1 \cdot z^1 + b_2}$$

a součin lze rozepsat takto:

$$(p - a) \cdot (p - b) = p^2 + p \cdot (-a - b) + a \cdot b$$