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

5 Realizace číslicových filtrů pomocí ATmega644

V této kapitole jsou popsány realizace číslicových filtrů dle kapitol 4.2 a 4.3 pomocí mikrokontroléru ATmega644 viz [1], [2].

*Nejdříve je uvedeno zapojení jednouchého přizpůsobovacího modulu **EADC**, který zajistí úpravu amplitudy signálu a vložení nezbytného stejnosměrného předpětí před přivedením na vstup A/D převodníku.*

*Následuje popis modulu **MSPIDAC** což je 12bitový D/A převodník řízený pomocí sběrnice SPI. Pomocí zabudovaného A/D převodníku a vnějšího D/A převodníku jsou vytvořena rozhraní pro připojení vstupního signálu a vytvoření výstupního signálu.*

*Funkce programu je nejdříve ověřena pomocí tzv. **transparentního režimu**, kdy výstupní signál odpovídá vstupnímu signálu. Poté jsou provedeny realizace číslicových filtrů a porovnány výsledky z jednotlivých variant.*

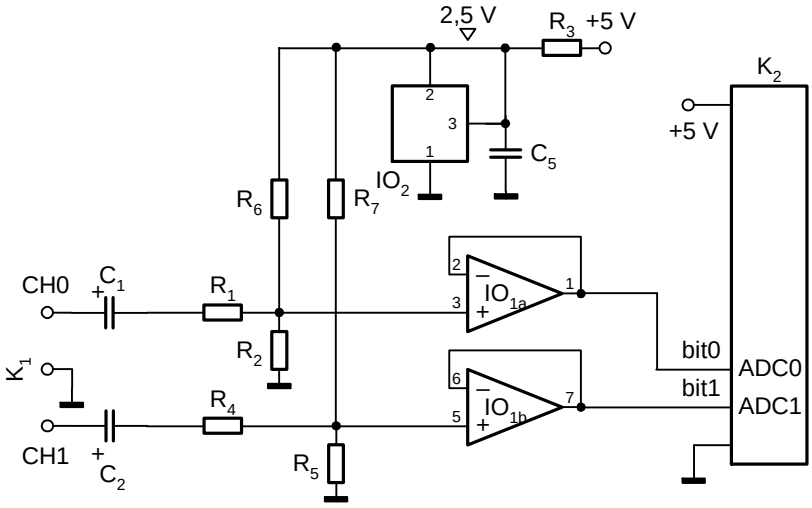
5.1 Přípravek EADC – vstupní modul pro A/D převodník

Schéma zapojení přípravku **EADC** je uvedeno na obr. 11.1.

Provedení je vzhledem k možnostem použitého dvojitého operačního zesilovače dvoukanálové. Dále popíšeme pouze funkci prvního kanálu (CH0).

Vstupní signál přivádíme mezi vývody CH0 a GND. Derivační článek C_1, R_1 slouží pro odstranění stejnosměrné složky. Mezní frekvence je nižší než 1 Hz. Výsledný signál je pak obojí polarity.

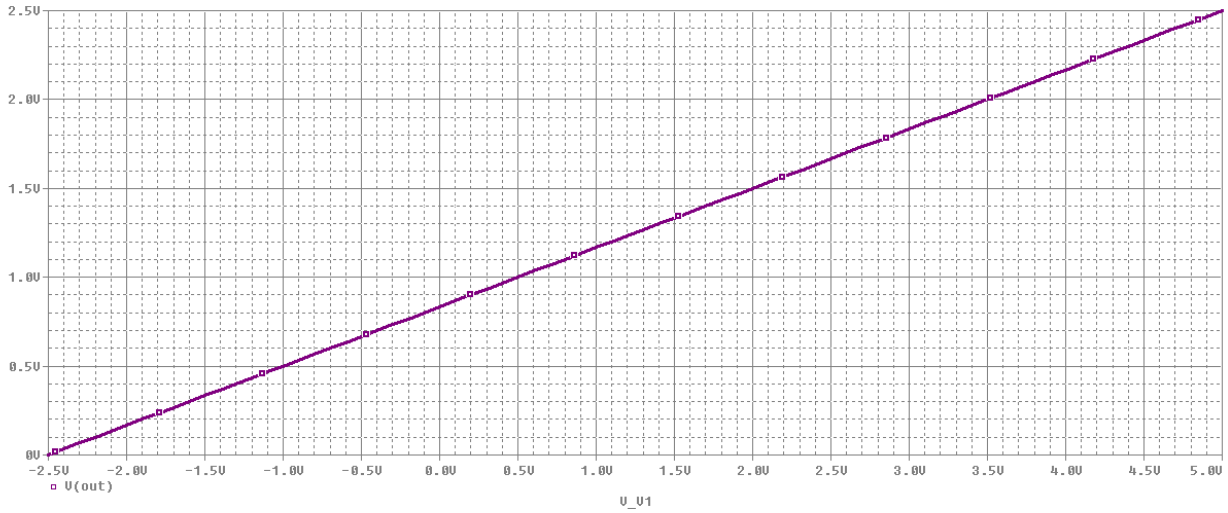
V uzlu R_1, R_6, R_7 je tento signál sloučen se stejnosměrnou složkou poskytovanou napěťovou referencí IO_2 (viz text níže). Následně je zařazen sledovač tvořený operačním zesilovačem IO_1. Sledovač poskytne nízký výstupní odpor, který je nutný pro úspěšný A/D převod. Tento signál je přiveden na bit0 datového konektoru. Připojíme-li přípravek k portu PA procesoru ATmega644, bude připojen na signál **ADC0** (kanál 0 zabudovaného A/D převodníku).



*Obr. 5.1 Schéma zapojení přípravku **EDAC** (blokovací kondenzátory nejsou kresleny)*

Rezistory R_1, R_2 a R_6 mají hodnotu 10 kΩ. Tímto způsobem je vytvořen dělič, který dělí vstupní napětí ze svorky CH0 na třetinu. Podobně i referenční napětí hodnoty 2,5 V je děleno na třetinu. Oba tyto signály se sčítají na vstupu sledovače, který jej pak přivádí na vstup A/D převodníku.

Účelem děliče je především vložit kladné předpětí do signálu tak, aby signál obojí polarity bylo možné zpracovat A/D převodníkem, který má pracovní rozsah 0 až 2,56 V. Konstrukce děliče je velmi jednoduchá, převodní charakteristika není symetrická. Viz obr. 5.2.



Obr. 5.2 Převodní charakteristika vstupního děliče získaná simulací programem PSpice

Z grafu dle obr. 5.2 je zřejmé, že pro přivedení nulové hodnoty napětí na vstup A/D převodníku musí být vstupní napětí –2,5 V. Kdežto pro výstupní napětí 2,5 V musí být vstupní napětí 5 V. Tato nesymetrie je zřejmá i z toho, že pro dosažení poloviny rozsahu výstupu (1,25 V) není vstupní napětí rovno nule.

Zmíněná nesymetrie není příliš na závadu, omezuje však částečně zpracovatelný rozsah signálu.

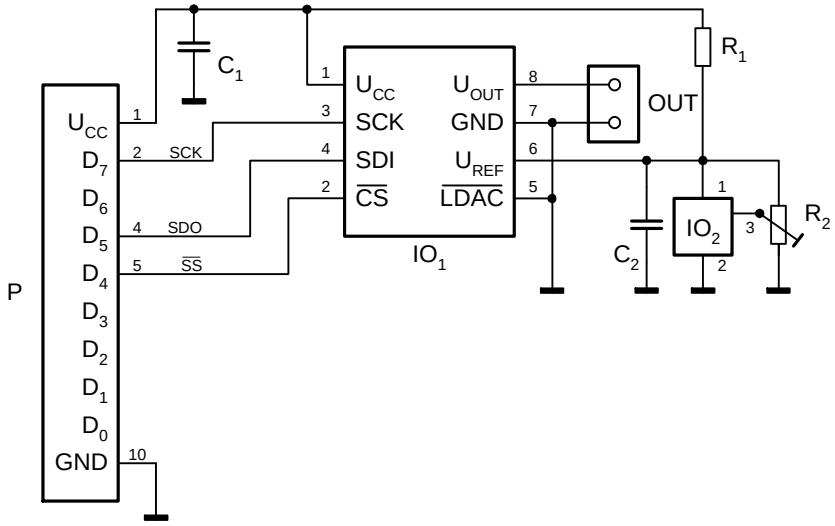
Vstupní odpor kanálu je 15 kΩ. Zdroj signálu nesmí mít příliš velký výstupní odpor, protože jeho hodnota ovlivňuje poměry při úpravě signálu v děliči.

Vstupní obvod tvořený rezistory je důsledkem použití jednoduchého napájecího napětí.

Podklady pro výrobu přípravku **EADC** nalezneme v příloze v kapitole A.2.

5.2 Přípravek MSPIDAC – sériový 12bitový D/A převodník

Schéma zapojení přípravku **MSPIDAC** je uvedeno na obr. 5.3.



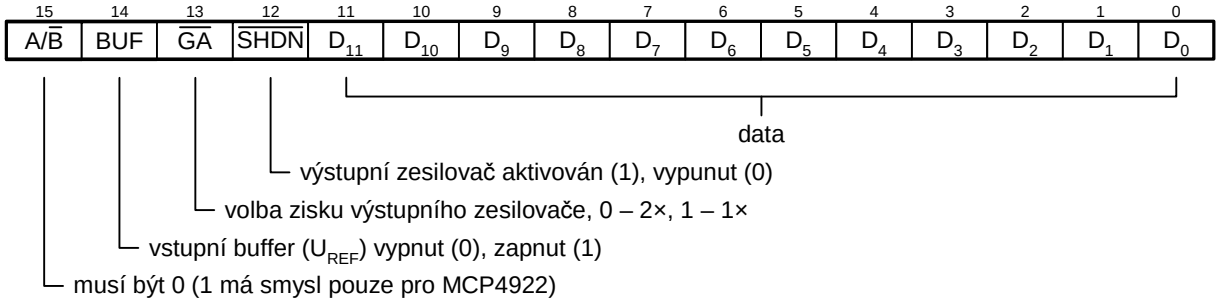
*Obr. 5.3 Schéma zapojení přípravku **MSPIDAC***

Jádrem převodníku je integrovaný obvod **MCP4921** označený jako IO_1. Jako napěťová reference byl použit obvod **TL431** (IO_2). Rezistor R_1 pak omezuje jeho pracovní proud a pomocí trimru R_2 nastavíme referenční napětí (mezi vývody 5 a 6 obvodu IO_1). Dále budeme uvažovat hodnotu 2,5 V, která odpovídá horní poloze jezdce

trimru R_2 dle obr. 5.3.

Kondenzátory C_1 a C_2 pracují jako blokovací. Maximální kapacita kondenzátoru C_2 může být pouze 1 nF! Výstupní signál je k dispozici na svorkovnici **OUT**.

Data odesílaná nadřízeným obvodem do převodníku MCP4921 mají délku 16 bitů. Kromě 12 datových bitů se ještě odesílají úvodní bity, které konfigurují činnost převodníku. Viz obr. 5.4.



Obr. 5.4 Formát příkazu

Klíčové vlastnosti převodníku MCP4921 (podrobnější popis lze nalézt v [2]):

- rozlišení 12 bitů,
- typická diferenciální nelinearita ±0,2 LSB, integrální nelinearita ±2 LSB,
- maximální kmitočet SPI sběrnice až 20 MHz,
- doba ustálení maximálně 4,5 μs,
- volitelné zesílení výstupu 1× nebo 2×,
- napájecí napětí v rozsahu 2,7 až 5,5 V.

Podklady pro výrobu přípravku **MSPIDAC** nalezneme v příloze v kapitole A.3.

5.3 Transparentní režim

Před vlastním ověřením funkce dříve navržených číslicových filtrů je vhodné otestovat funkčnost programu v tzv. transparentním režimu. Jeho činnost je velmi prostá, A/D převodník převádí vstupní napětí a takto získaný vzorek je bez úprav použit D/A převodníkem. Tedy v ideálním případě dostáváme na vstupu a výstupu shodné signály. Signály se však prakticky budou lišit jednak tím, že dochází ke kvantování (výstup bude schodovitou aproximací vstupu) a dále velikostí amplitudy vstupního a výstupního signálu, případně i ofsetem.

Nastavení spouštěcího zdroje

Vzorkovací kmitočet byl uvažován **f_{vz} = 10 kHz**, tedy po uplynutí **100 μs** se musí vykonat sejmutí vstupního vzorku A/D převodníku a spustit následný D/A převod. Pro časování vzorkovací operace použijeme **čítač/časovač 0 v režimu CTC**, který dává široké možnosti konfigurace a zajistí také velmi přesné nastavení požadovaného intervalu. Ve shodě s [2] lze vyjít ze vztahu:

$$T = \frac{P \cdot (OCR0A + 1)}{f_0} \qquad (5\text{-}1)$$

Kde T je časový interval, který lze odměřit. P je nastavení předděličky (1, 8, 64, 256, 1024), $OCR0A$ je obsah komparačního registru a f_0 je hodinový kmitočet mikrokontroléru.

Úpravou lze získat vztah pro určení hodnoty komparačního registru při známých hodnotách časového intervalu, systémového kmitočtu a nastavení předděličky:

$$\mathrm{OCR0A} = \frac{f_0 \cdot T}{P} - 1 \qquad (5\text{-}2)$$

Provedeme rozvahu pro všechny varianty nastavení předděličky s tím, že stanovíme hodnotu OCR0A:

1. bez předděličky, $P = 1, \mathrm{OCR0A} = 1999$, nereálné (OCR0A je 8bitový registr),
2. **předdělička 1/8,** $P = 8, \mathrm{OCR0A} = 249$**, v pořádku,**
3. předdělička 1/64, $P = 64, \mathrm{OCR0A} = 30{,}25$, nepřesné (OCR0A je celé číslo),
4. předdělička 1/256, $P = 256, \mathrm{OCR0A} = 6{,}8125$, nepřesné (OCR0A je celé číslo),
5. předdělička 1/1024, $P = 1024, \mathrm{OCR0A} = 0{,}953123$, nepřesné (OCR0A je celé číslo).

Nastavení A/D převodníku

Dále budeme diskutovat **nastavení A/D převodníku**. Dle [2] platí, že rychlost převodu je až 15 kSPS při rozlišení 10 bitů. S ohledem na požadovanou hodnotu vzorkovacího kmitočtu 10 kHz, která je s předchozí hodnotou srovnatelná, nemusí vzniknout dostatečná časová rezerva pro vlastní výpočet. Přirozeně můžeme využít možnosti automatického spouštění A/D převodníku pomocí již konfigurovaného čítače/časovače 0. Pro případné vyšší vzorkovací kmitočty ale rychlost převodníku nebude dostatečná.

Pro přesný výpočet doby převodu musíme dále uvážit, že hodinový kmitočet převodu se musí pohybovat v rozmezí 50 až 200 kHz (při přesnosti 10 bitů). Hodinový kmitočet převodu se odvodí ze systémového kmitočtu dělením (dělicí poměry: 2, 4, 8, 16, 32, 64, 256). Pro získání co nejvyšší hodnoty hodinového kmitočtu docházíme k volbě 256, tedy kmitočtu f_{ADC} = 156,25 kHz. Převod včetně vzorkování trvá 13 hodinových taktu f_{ADC}. Tedy lze napsat vztah:

$$T_P = \frac{13}{f_{ADC}} \qquad (5\text{-}3)$$

Prostým dosazením zjistíme dobu převodu T_P = 83,2 μs. Pro uvažovanou hodnotu vzorkovacího kmitočtu je tato hodnota v pořádku, neboť platí: $T_P < T_{VZ}$ (83,2 μs < 100 μs). Pro vyšší vzorkovací kmitočty je zabudovaný A/D převodník již nepoužitelný.

Z tohoto důvodu budeme dále používat pouze **8bitovou přesnost**. Dle tab. 16.1 z [2] je totiž možné zvýšit hodinový kmitočet převodu až na 1 MHz s tím, že se sníží přesnost převodu. Dále budeme uvažovat **nastavení předděličky 64** a odpovídající hodnotu kmitočtu **f_{ADC} = 312,5 kHz**. Toto nastavení dovolí hypotetickou hodnotu vzorkovacího kmitočtu až 24 kHz.

Nastavení SPI kanálu pro D/A převodník

Obvod **MCP4921** používá hodinový kmitočet přenosu až 20 MHz. SPI kanál integrovaný v mikrokontroléru ATmega644 poskytuje při systémovém kmitočtu 20 MHz maximální hodnotu přenosového kmitočtu 10 MHz, viz tab. 12.2 z [2]. Tedy **f_{SPI} = 10 MHz**. Celkový interval odesílání 16bitových dat dle obr. 5.4 lze pak určit ze vztahu:

$$T_C = \frac{16}{f_{SPI}} \qquad (5\text{-}4)$$

Prostým dosazením zjistíme T_C = 1,6 μs. Což by odpovídalo hypotetickému vzorkovacímu kmitočtu až 625 kHz, tedy interval je zanedbatelný. Jelikož je však nutné zajistit i aktivitu vodiče $\overline{CS}$, bude skutečný interval přenosu dat do D/A převodníku poněkud delší. Pro optimální využití možností mikrokontroléru budeme řešit odeslání dvou 8bitových údajů pomocí přerušení (SPI jednotka neposkytuje možnost odesílání 16bitových dat přímo, přenos probíhá vždy po 8 bitech).

Zavedené symboly
Pro vyšší přehlednost zavedeme tyto symboly:

- **CS** – číslo bitu (4), který ovládá signál $\overline{CS}$ pro D/A převodník.
- **F0_8** – zvolená předdělička čítače/časovače 0 (dělení systémového kmitočtu osmi; čítá se kmitočet 2,5 MHz; perioda 400 ns).
- **WGM_CTC** – režim činnosti čítače/časovače 0 (CTC, tedy 2).
- **OC0APRE** – délka periody čítače/časovače 0 snížená o 1 (249; perioda má délku 100 μs což odpovídá požadovanému vzorkovacímu kmitočtu 10 kHz).
- **ADP_64** – zvolená předdělička A/D převodníku (dělení systémového kmitočtu 64; kmitočet pro A/D převod je 312,5 kHz).
- **ADT_CTC0** – zvolený zdroj spouštění A/D převodu (perioda daná CTC režimem čítače/časovače 0).

Používané globální proměnné
Výpočet používá tyto globální proměnné:

- **int x** – vstupní vzorek (upravený výsledek A/D převodu),
- **int y** – výstupní vzorek (hodnota použitá pro D/A převodník),
- **unsigned dataDAC** – 12bitová data odesílaná na D/A převodník (upravená hodnota y),
- **unsigned char cntDAC** – počítadlo bajtů pro odesílání na D/A převodník.

Zápis programu

Zdrojový text programu pro transparentní režim je vypsán níže.

Na začátku programu jsou uvedeny obvyklé definice (volba typu mikrokontroléru) a vloženy potřebné hlavičkové soubory (definice registrů, funkce pro řízení přerušení).

Následuje definice symbolů, které používáme pro zvýšení přehlednosti konfigurace jednotlivých registrů.

Jsou deklarovány globální proměnné. Připomeňme, že tyto proměnné jsou označeny klíčovým slovem **volatile**, které zajišťuje správné chování překladače při optimalizaci kódu. Jako volatile označujeme proměnné, jejich hodnota se obvykle mění pomocí obsluhy přerušení, což je i náš případ.

Funkce **StartDAC** slouží pro spuštění přenosu dat do D/A převodníku. Nejdříve aktivujeme signál $\overline{CS}$ ($\overline{CS} = 0$). Data pro odesílání jsou uložena v globální proměnné **dataDAC**. Odebereme z těchto dat bity 11 až 8 (horní 4 bity 12bitového čísla) a sloučíme je s řídicími bity dle obr. 5.4. Tuto hodnotu uložíme do registru **SPDR** čímž je odstartován přenos. Po přenosu těchto 8 bitů bude aktivováno přerušení, které zajistí odeslání druhého bajtu. Počítadlo **cntDAC**, které je používáno obsluhou přerušení pro počítání bajtů, musí být vynulováno. Funkce je zapsána jako inline, aby její volání bylo časově úsporné (funkce se fakticky nevolá, ale její kód se rozvine v místě použití; příkazy seskupujeme do formy funkce pro zvýšení přehlednosti zápisu).

Obsluha přerušení jednotky SPI. Jak bylo naznačeno, je odesílání 16bitových dat na D/A převodník řešeno pomocí přerušení. To je vhodné řešení zvláště v případě,

že je nutno odesílat data nadvakrát (po 8bitech) a ještě řídit aktivitu signálu $\overline{CS}$. Pro rozeznání jednotlivých bajtů a konce přenosu používáme počítadlo – globální proměnnou **cntDAC**. Toto počítadlo je při každé aktivaci obsluhy přerušení zvýšeno o 1, obsluha přerušení je totiž vyvolána vždy po vyprázdnění vysílacího bufferu (tedy po odeslání 8 bitů). Po odeslání 4 bitů pro konfiguraci a horních 4 datových bitů je obsluha přerušení aktivována poprvé a počítadlo má hodnotu **cntDAC = 1**, odešleme tedy nižší bajt dat získaný bitovým součinem obsahu proměnné **dataDAC** s hodnotu 0xFF (255 desítkově). Podruhé bude obsluha přerušení aktivována po odeslání druhého bajtu (**cntDAC = 2**). Nyní musíme ukončit komunikaci deaktivací signálu $\overline{CS}$. Přechodem $\overline{CS}=1$ dojde k přepisu vnitřního posuvného registru do záchytného registru D/A převodníku a tedy ke změně výstupního napětí.

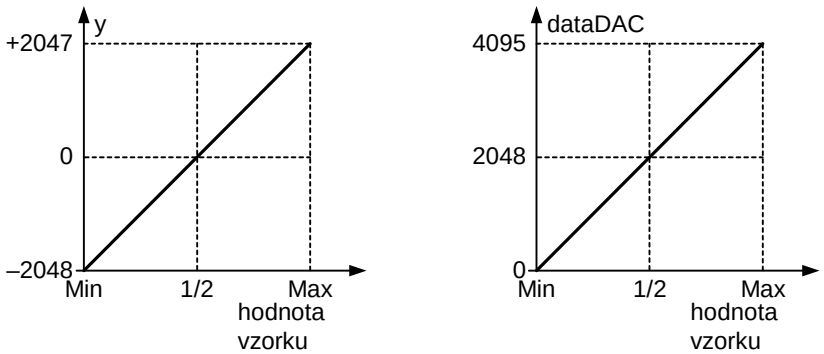
Obsluha přerušení A/D převodníku. Vždy po uplynutí intervalu 100 μs je automaticky spuštěn nový A/D převod a po jeho dokončení je aktivována obsluha přerušení. Tato obsluha zjišťuje řízení obou převodníků.

Jako první nulujeme příznak přerušení čítače/časovače 0 (nulování se provádí zápisem 1 do bitu OCF0A registru TIFR0). Tato operace je nezbytná, protože automatické spouštění A/D převodů je hranově citlivé a spouštěcí příznak musí být po svém použití vždy vynulován, aby mohla vzniknout nová spouštěcí hrana. Viz příklad PROG_23 z [2].

Velmi důležité je povolení přerušení pomocí makra **sei**. Tato operace dovolí **vnořené přerušení**. Jelikož má obsluha přerušení jednotky SPI vyšší prioritu než obsluha přerušení A/D převodníku, bude možné provést obsluhu jednotky SPI přednostně při vykonávání obsluhy A/D převodníku. Viz tab. 8.1 z [2].

Nejdříve jsou sestavena data pro D/A převodník a uložena do globální proměnné **dataDAC** (použití globální proměnné je nanejvýš vhodné, protože k obsahu této proměnné přistupuje jak funkce StartDAC tak i obsluha přerušení jednotky SPI).

D/A převodník se obsluhuje jako první proto, aby bylo zajištěno časově přesné taktování výstupního signálu. Při výpočtu vzorků může totiž vznikat předem nedefinované zpoždění. Navíc výpočet může dokonce probíhat různě dlouhou dobu v závislosti na konkrétních hodnotách vzorků. Umístěním obsluhy D/A převodníku na konec obsluhy přerušení by vznikala nejistota (jitter) vzorkování výstupu. Odesílání dat na D/A převodník je odstartováno pomocí funkce **StartDAC**.



Čísla se znaménkem vyjádřená ve dvojkovém doplňku:

–2048=1111 1000 0000 0000
0=0000 0000 0000 0000
+2047=0000 0111 1111 1111

Čísla se bez znaménka a jejich binární kódy:

0=0000 0000 0000 0000
2047=0000 0111 1111 1111
4095=0000 1111 1111 1111

Obr. 5.5 Převod mezi vzorkem se znaménkem a vzorkem bez znaménka pro 12 bitů

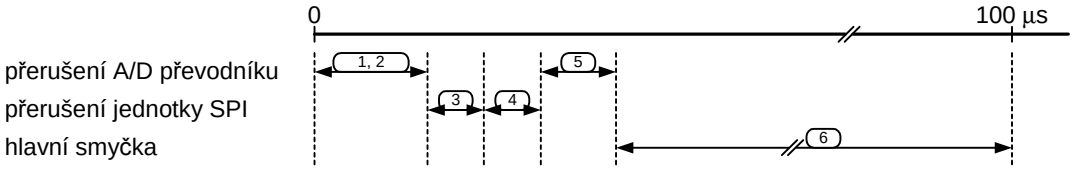
Hodnota vzorku uložená v proměnné **y** je ve formátu celého čísla se znaménkem. Provedeme převod na číslo bez znaménka prostým posunutím hodnoty o polovinu rozsahu. 12bitové číslo bez znaménka má rozsah hodnot 0 až 4095. 12bitové číslo se znaménkem má rozsah hodnot –2048 až +2047. Proto číslo se znaménkem posuneme o 2048, abychom získali odpovídající hodnotu čísla bez znaménka. Tento přepočet je nutný proto, že při výpočtu je na vzorek nahlíženo jako na číslo se znaménkem. Samotný D/A převodník však nepracuje v doplňkovém kódu. Viz obr. 5.5.

Jako druhý krok následuje **získání vzorku z A/D převodníku**. Pro použitý 8bitový režim čteme obsah registru **ADCH**. S ohledem na skutečnost, že D/A převodník pracuje s rozlišením 12bitů, posuneme získanou 8bitovou hodnotu o 4 bity doleva (tedy rozšíříme z 8 bitů na 12 bitů, dolní 4 bity jsou vynulovány). Takto rozšířenou hodnotu poté posuneme o 2048 dolů pro provedení přepočtu na číslo se znaménkem.

Vlastní výpočet je v transparentním režimu velmi jednoduchý. Hodnotu vzorku z A/D převodníku předáme na D/A převodník (fakticky bude tento vzorek podržen v proměnné y a použije se až při další aktivaci obsluhy přerušení). Přiřazení **y = x** popisuje funkci transparentního režimu velmi výstižně.

Běh celého kódu, který je rozdělen do dvou přerušení, je na představu poněkud složitější a je proto znázorněn formou obr. 5.6:

1. Na začátku obsluhy přerušení čítače/časovače 0 je uložena hodnota 12bitových dat do globální proměnné **dataDAC** a následně volána funkce **StartDAC**. Tato funkce však pouze spustí přenos bitů a poté se okamžitě vrací.
2. Následuje čtení výsledku A/D převodu a vstupní vzorek je uložen do proměnné **x**.
3. Jakmile je detekováno odeslání prvního bajtu, je aktivována obsluha přerušení jednotky SPI, **cntDAC = 1**.
4. Jakmile je detekováno odeslání druhého bajtu, je aktivována obsluha přerušení jednotky SPI, **cntDAC = 2**. Nová hodnota vzorku vytvoří novou hodnotu výstupního napětí.
5. Je proveden výpočet hodnoty výstupního vzorku a hodnota je uložena do proměnné **y**. Tím obsluha přerušení čítače/časovače 0 končí.
6. Provádění programu se vrací do hlavní smyčky. Hlavní smyčka bude přerušena obsluhou dalšího vzorku. Činnost se tedy opakuje.



Obr. 5.6 Běh výpočtu

Vidíme tedy, že obsluha přerušení A/D převodníku nejdříve spustí přenos vzorku předchozího výpočtu do D/A převodníku. Poté pokračuje ve svém běhu, ale tento běh je 2× přerušen obsluhou jednotky SPI.

Tím je zajištěno souběžné provádění obsluhy A/D převodníku a přenos dat na D/A převodník (jednotka SPI provádí přenos nezávisle na jádru procesoru).

Hlavní program zajistí konfiguraci jednotlivých jednotek výše popsaným způsobem. Je povoleno přerušení A/D převodníku a jednotky SPI. Přerušení čítače/časovače 0 není povoleno, nulování příznaku zajišťujeme programově v obsluze přerušení A/D převodníku.

Nakonec je povoleno přerušení a hlavní program přechází do nekonečné smyčky. Tato smyčka je vždy přerušena v okamžiku požadavku obsluhy přerušení.

TRANSPARENTNI.C:

```
#define __AVR_ATmega644__ 1
#include <avr/io.h>
#include <avr/interrupt.h>
```

```
#define CS       4    //PB4 je CS signal pro MCP4921
#define F0_8     2    //preddelicka c/c 0 (f0/8)
#define WGM_CTC  2    //rezim c/c 0 (CTC)
#define OC0APRE  (250-1)  //delka citace -1 pro CTC rezim
#define ADP_64   6  //preddelicka A/D prevodniku
#define ADT_CTC0 3  //zdroj spousteni A/D prevodniku
```

```
//promenne pro vypocet a rizeni D/A prevodniku:
volatile int x;    //vstupni vzorek
volatile int y=0;  //vystupni vzorek
volatile unsigned dataDAC;  //12bitova data do D/A prevodnik
volatile unsigned char cntDAC=0;  //pocitadlo pro rizeni
                                  //SPI komunikace
```

```
//spusteni prenosu na MCP4921:
inline void StartDAC()
{
  PORTB&=~(1<<CS);  //CS=0, aktivuj obvod
  SPDR=((dataDAC>>8)&0x0f)|0b00110000;  //odesli 4 bity rezimu
                                        //+4 datove bity
  cntDAC=0;  //nuluj ridici pocitadlo
}
```

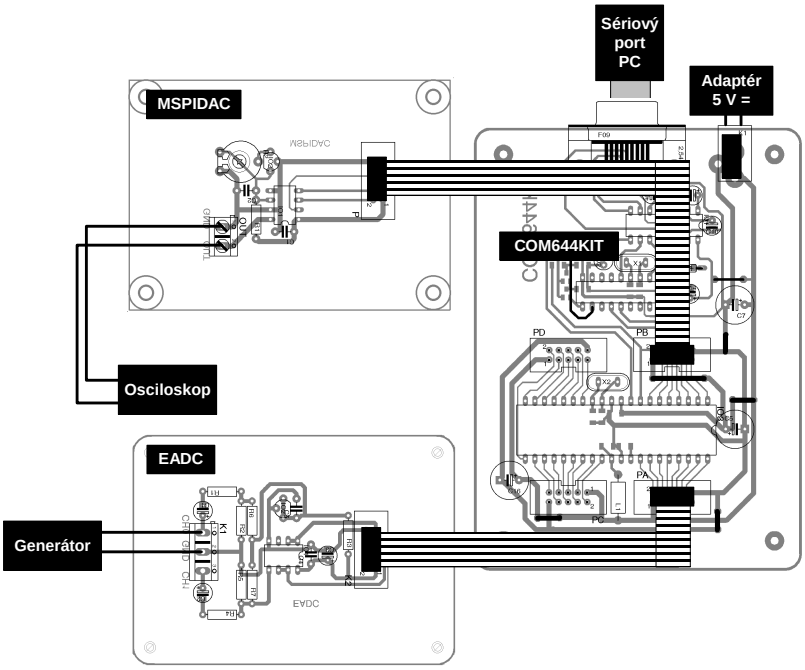
```
//obsluha preruseni SPI jednotky pro rizeni MCP4922:
ISR(SPI_STC_vect)
{
  cntDAC++;  //pocitadlo aktivaci pro rizeni komunikace

  if(cntDAC==1)  //po poslani hornich bitu posli dolnich bity
    SPDR=dataDAC&0xff;
  else         //nakonec dej CS do neaktivniho stavu
    PORTB|=(1<<CS);   //CS=1
}
```

```
//obsluha preruseni A/D prevodniku:
ISR(ADC_vect)
{
  TIFR0=1<<OCF0A;  //nulovani priznaku c/c 0
  sei();           //povoleni vnoreneho preruseni
  //odeslani predochozi vysledku:
  dataDAC=y+2048;  //odebrani znamenka, rozsireni na 12 bitu
  StartDAC();          //start D/A prevodu (vysilani pres SPI)

  //cteni vstupniho vzorku:
  x=(ADCH<<4)-2048;          //uprava na cislo se znamenkem

  //vlastni vypocet:
  y=x;
}
```

```
//hlavní program-konfigurace jednotek a nekonečná smyčka:
int main()
{
  //konfigurace c/c 0:
  TCCR0A=WGM_CTC;  //rezim CTC
  TCCR0B=F0_8;     //preddelicka 8
  OCR0A=OC0APRE;   //predvolba pro OCR0A

  //konfigurace A/D prevodniku:
  ADMUX=(1<<REFS1)|(1<<REFS0)|(1<<ADLAR); //vnitrni ref., 8bitu
  ADCSRA=(1<<ADEN)|(1<<ADATE)|(ADP_64)|(1<<ADIE);  //auto,
                                                   //P=64,
                                                   //preruseni
  ADCSRB=ADT_CTC0;                          //spousteni pres CTC0

  //inicializace SPI:
  DDRB=0xFF;   //aktivuj výstupy
  SPCR=(1<<SPIE)|(1<<SPE)|(1<<MSTR);  //režim master
  SPSR=(1<<SPI2X);                    //rychlost f0/2 (10 MHz)

  sei();  //globalni povoleni preruseni

  while(1);  //nekonecna smycka
}
```

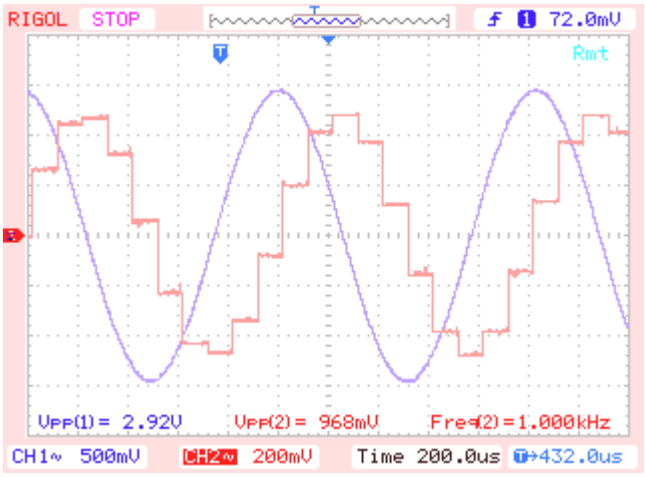
Funkci transparentního režimu ověříme tak, že přípravek **EADC** připojíme na port **A** vývojového kitu **COM644KIT (konstrukce tohoto kitu je popsána v příloze v kapitole A.1)**. Na kanál **CH0** pak přivedeme signál ze sinusového generátoru. Přípravek **MSPIDAC** připojíme na port **B** vývojového kitu a výstup sledujeme pomocí osciloskopu.

Vývojový kit připojíme k počítači a zapojíme i napájecí adaptér. Do mikrokontroléru nahrajeme program, který nalezneme na doprovodném CD-ROM v adresáři **PROGRAMY\FILTRY\ATMEGA644\TRANSPARETNI**.



Obr. 5.7 Sledovaná sestava

Výsledek ověření transparentního režimu je dokumentován formou oscilogramu, který je uveden na obr. 5.8. Měření bylo provedeno pro vstupní signál kmitočtu 1 kHz. Pro kontrolu je na prvním kanálu osciloskopu vstupní signál a na druhém kanálu výstupní signál. Oba kanály byly vázány střídavě. Rozkmit obou signálů je na oscilogramu také zobrazen.



Obr. 5.8 Vstupní a výstupní signál v transparentním režimu

5.4 Realizace filtru dle kapitoly 4.2

V kapitole 4.2 byl navržen filtr se vzorkovacím kmitočtem 10 kHz s koeficienty, které lze popsat následujícím souborem rovnic:

```
v=x+0.7478*v1-0.2722*v2
p=z-0.7478*p1-0.2722*p2
z=0.50505*v-1.0101*v1+0.50505*v2
y=0.50505*p+1.0101*p1+0.50505*p2

//posun pamětí
v2=v1
v1=v
p2=p1
p1=p
```

V následujícím textu budeme tento filtr realizovat různými způsoby za účelem porovnání časové náročnosti a získaných parametrů filtru.

Realizace pomocí float-point aritmetiky

Z hlediska programátora je nejsnazší realizací použití aritmetiky plovoucí řádové čárky (float-point). Tato varianta však naráží na požadavek vysoké výpočetní rychlosti použitého mikrokontroléru. Ověříme, zda mikrokontrolér ATmega644 taktovaný 20 MHz je schopen výpočet provést v čase kratším než 100 µs. Pokud ano, lze tuto realizaci filtru provést.

Zápis programu vychází z předchozího transparentního režimu. Výpis je proto zkrácen. Celý program naleznete na doprovodném CD-ROM v adresáři **PROGRAMY\FILTRY\ATMEGA644\PP_DLE_42\FLOAT**.

Program je pouhým přepsáním výše uvedeného souboru rovnic do jazyka C. Pochopitelně musely být deklarovány proměnné pro uložení mezivýsledků a pro realizaci paměťových vzorků.