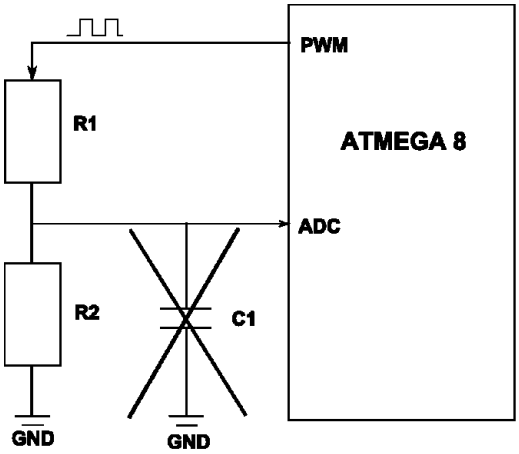
Pokročilé programovanie mikropočítačov v jazyku C. - Diel č.3

Publikované: 23.01.2016, Kategória: Mikroprocesory

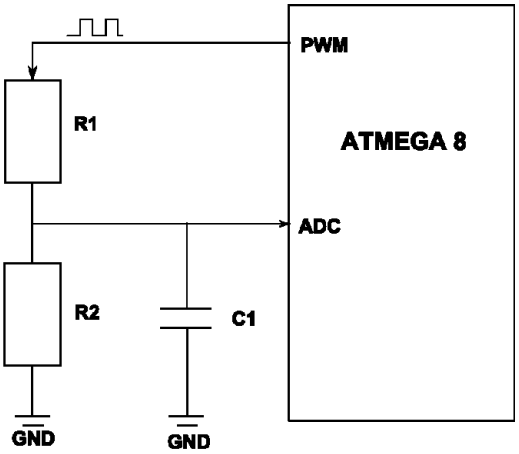
www.svetelektro.com

Filtrujeme meraný signál

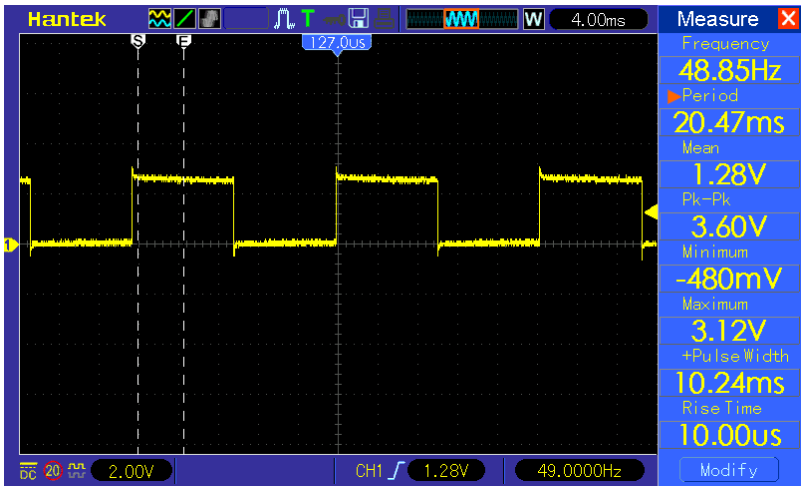
Ahojte programátori. Pripravil som si článok o filtrovaní signálov. Nechcem sa na úvod moc vykecávať o filtrovaní preto hneď skočíme priamo do deja.
Čo sa stane alebo čo budem vidieť na osciloskope, keď pustím PWM (obdĺžnikový) signál do obvodu na obr.1a resp. obr.1b? Odpoveď som si pozrel priamo na osciloskope. PWMmkou som si generoval signál s frekvenciou 50Hz. R1=R2=50 Ω. C1 = 133nF.



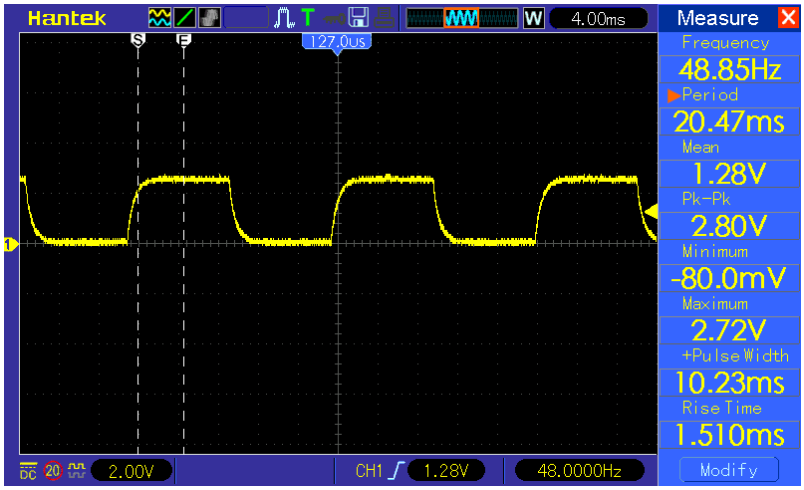
OBR. 1 a



OBR. 1 b

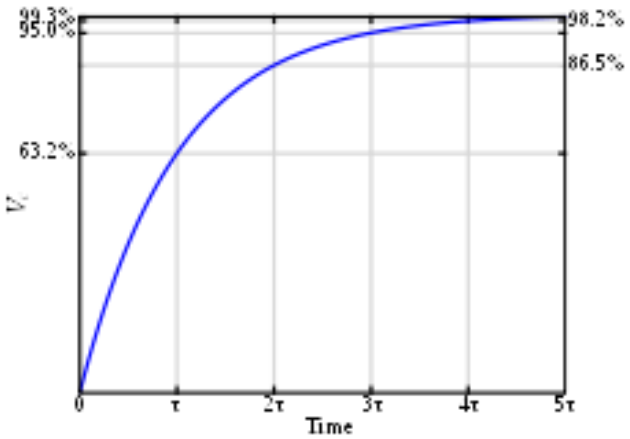


Obr.2 Výsledok pre zapojenie bez kondenzátora (podľa obr.1a)



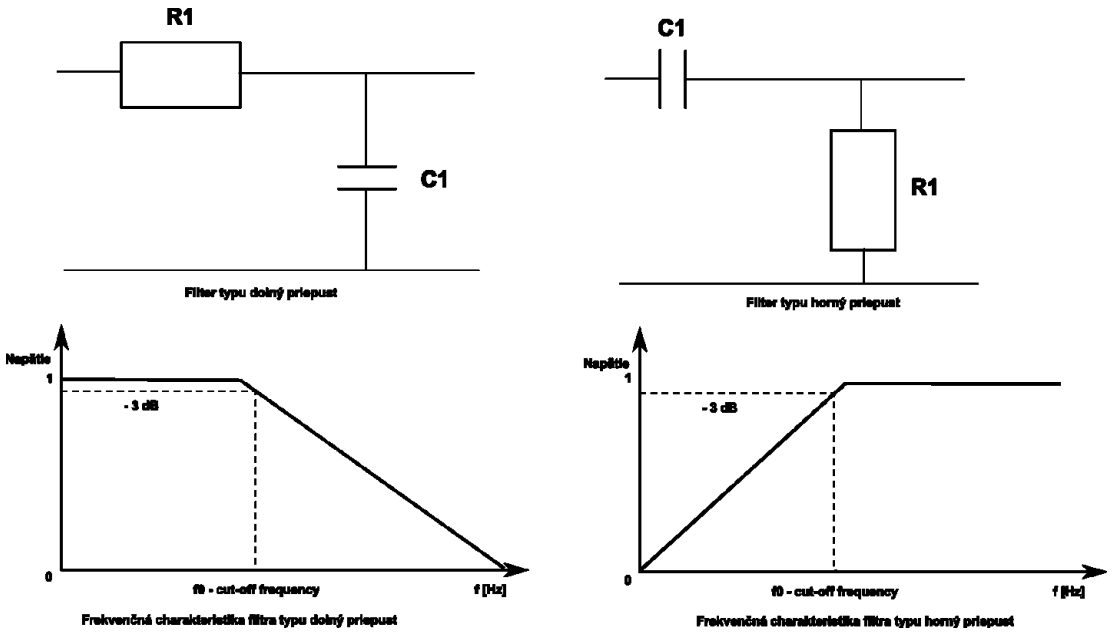
Obr.3 Výsledok pre zapojenie s kondenzátorom (podľa obr.1b)

Teda v krátkosti:
Kondenzátor v obvode funguje ako filter pre vyššie frekvencie. Reaktancia kondenzátora klesá so zvyšujúcou frekvenciou. Zjednodušene môžeme povedať, že kondenzátor sa javí pre vyššie frekvencie ako skrat. Vyhladzuje ostré hrany obdĺžnikového signálu . Prečo? Ak už niekto počul o Fourierovi, príde mu na um, že obdĺžnikový signál sa dá vyskladať pomocou nepárnych harmonických sinusového signálu. Teda obdĺžnikový signál je vlastne zhluk sínusoviek, pričom frekvencia prvej harmonickej je práve frekvencia obdĺžnikového signálu. To, ako rýchlo sa kondenzátor nabije na ustálenú hodnotu napätia, záleží od parametrov R1 a C . Z parametrov R1 a C sa dá vypočítať časová konštanta obvodu. Časová konštanta obvodu je čas, za ktorý sa kondenzátor nabije na 63,2 % ustálenej hodnoty. Trojnásobok časovej konštanty je čas, za ktorý sa kondenzátor nabije na 95,0 % ustálenej hodnoty. Odpor R2 v obvode slúži len ako delič napätia. Nemá žiadny vplyv na RC filter tvorený R1 a C.



Obr.4 Priebeh napätia na kondenzátore v RC obvode (exponenciálny priebeh)

RC obvod funguje aj ako filter typu dolný resp. horný priepust v závislosti od toho ako je v obvode zapojený.



Obr.4 Filter typu dolný priepust resp. horný priepust a ich frekvenčné charakteristiky

Viac o filtroch sa dá nájsť hocikde internete. Zase len v krátkosti, že cut-off frequency je po slovensky medzná frekvencia filtra. Zlomová frekvencia filtra, je taká frekvencia signálu pri ktorej je pomer amplitúdy signálu na výstupe k amplitúde signálu na vstupe 0,707 (útlm 3 dB). Medzná frekvencia RC článku sa počíta podľa nasledovného vzťahu: f0 = 1/(2πRC)

Teda v mojom prípade, kde R=10kΩ a C=133nF, je medzná frekvencia f0=120Hz.
Dalo by sa o tom ešte veľa písať, ale mne ide o niečo iné. Čo ak práve nemám pri sebe tú správnu hodnotu kondenzátora na vyfiltrovanie signálu? Ak mám k dispozícii dostatočný výpočtový výkon môžem si obdĺžnikový signál vyfiltrovať softwarovo. Ako na to?
Ako prvé musíme poznať prenosovú funkciu dolnopriepustného filtra 1. rádu. Niekto to vie z hlavy a niekto si to nájde na internete. Prenosová funkcia dolnopriepustného filtra 1. rádu je:
 H(s) = wc/(s+wc),
 kde wc je medzná frekvencia filtra, wc = 2πf0.
Túto prenosovú funkciu si musíme najskôr zdiskretizovať. Použijem bilineárnu transformáciu a ten istý postup ako v tomto dokumente na strane 13. Výsledkom je rovnica dolnopriepustého filtra 1. rádu v diskrétnom tvare.
y(n) = x(n)*b1 +x(n-1)*b2 – y(n-1)*a2
y(n) je aktuálna hodnota alebo vzorka na výstupe z filtra
x(n) je aktuálna vstupná hodnota alebo vzorka na vstupe filtra
x(n-1) je predchádzajúca vstupná vzorka
y(n-1) je predchádzajúca výstupná vzorka
b1,b2 a a2 sú koeficienty filtra , ktoré sa vypočítajú podľa vzťahov uvedených v spomínanom dokumente na strane 13.
Keď poznáme koeficienty filtra, môžeme prejsť k samotnému programovaniu a otestovaniu filtra.

Popis programu:

Pomocou časovača si generujem PWM signál s frekvenciou 50 Hz a s triedou 0,5. Tento PWM signál privádzam na napäťový delič R1 = R2 = 10 kΩ bez kondenzátora. Signál z napäťového deliča snímam ADC prevodníkom. ADC prevodník vzorkuje s frekvenciou 10 kHz. V prerušení od konca prevodu si uložím do poľa nameranú hodnotu a ešte aj stihnem vypočítať aktuálnu hodnotu na výstupe filtra. Keď si takto odmeriam a vyfiltrujem 150 vzoriek , pošlem si ich cez sériovú linku do PC a zobrazím si nameraný a vyfiltrovaný signál v exceli.

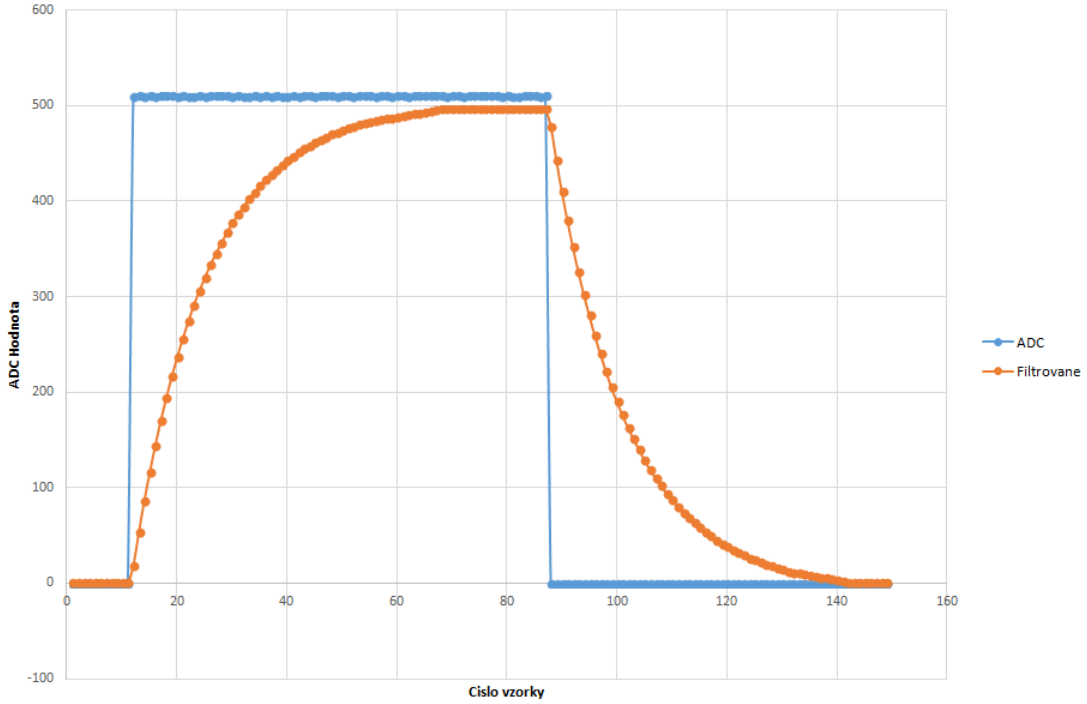
Ukážka kódu:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include "uart.h"

// konstanty filtra
#define A1 -0.9273
#define B1 0.0363
#define B2 B1

#ifndef F_CPU
#define F_CPU 8000000
#endif

//globalne premenne
volatile unsigned int adc;
volatile unsigned int result[150];
volatile unsigned int resultFilt[150];

volatile char counter = 0;

//deklaracie funcie filtruj
int filtruj(int input);

// prerušenie na akciu skoncenia AD prevodu
ISR(ADC_vect){
  result[counter]=ADC; // zapis vysledok prevodu
  resultFilt[counter] = filtruj(ADC);
        counter++;

}

int main(){

    char text[32];
    int i;


OSCCAL= 0xBE;

        // Nastavenie Timera 1 -  generovanie 50 Hz obdlznikovy PWM signal
             DDRB |= (1 << PB1); // OC1A ako výstupný
        TCCR1A |= (1 << COM1A0); // CTC rezim
        TCCR1B |= (1 << WGM12) | (1 << CS11); // preddelička 8 pri 8 MHz = (1us)
        OCR1A |= 9999;

        // Nastavenie ADC
        ADMUX |=  (1<< REFS0) | (1 << REFS1) | (1 << MUX0); //referencia 2.56V

//zapnutie AD prevodnika, Free runing, delicka 64 pri frek. hodin 8MHz, cca 10kHz vzorkovacia frekvencia
    ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADFR) | (1 << ADIE) | (1 <<ADPS2) | (1 <<ADPS1)  | (0
<<ADPS0);

 // inicializacia uart na rychlost 9600bd
    uart_init(19200);

    sei();  //povol globalne prerusenia

    while(1){
//vypis datovych poli, vypnutie ADC
    if (counter == 149)
    {
         ADCSRA = 0x...
```

Popis funkcie „filtruj“:

Funkcia filtruj filtruje nameraný signál presne podľa predpisu filtra prvého rádu, teda nasledovne:
y(n) = x(n)*b1 +x(n-1)*b2 – y(n-1)*a2
Aktuálna výstupná hodnota sa rovná aktuálna nameraná hodnota krát koeficient B1 plus predchádzajúca vstupná hodnota krát koeficient B2 mínus predchádzajúca výstupná hodnota krát koeficient A2.

Výsledok programu:



Obr.5 Výstup softwarového filtra

Záver:

Pomocou digitálneho filtra som dokázal vstupný obdĺžnikový signál vyfiltrovať rovnako ako s použitím analógového RC filtra. Výpočet jednej iterácie filtra som na ATMEGE8 taktovanej na 8 MHz dokázal len tak tak vtesnať do 100us (10kHz), teda do času medzi dvoma vzorkami. Samozrejme, celý výpočet filtra by sa dal ešte optimalizovať. Možností je viacero. Možno sa v budúcnosti ešte vrátim k tomuto príkladu a skúsim výpočet trochu zoptimalizovať, urýchliť. Chcel som v tomto článku ukázať, aj keď tak trochu z rýchlika, že meraný signál sa dá veľmi jednoducho filtrovať aj na strane MCU.

Výhody digitálneho filtra:

- Nezávislý na okolitej teplote (a veku filtra), čo v prípade analógových súčiastok nie je možné
- Dá sa jednoducho prekonfigurovať (preladiť) na iné frekvencie jednoduchou zmenou koeficientov. Netreba hľadať vhodné hodnoty a kombinácie R a C.
- Dajú sa jednoducho navrhnúť digitálne filtre vyšších rádov alebo filtre s charakteristikou, ktoré by sa s použitím analógových súčiastok veľmi ťažko vytvorili (potrebné sú len základy matematiky, úprava zlomkov).
- Samozrejme, digitálne filtre majú aj svoje nevýhody alebo obmedzenia, viac pre záujemcov napríklad tu

Použitý hardware:

- R1 = 10kΩ, R2 = 10kΩ, C1 = 133nF a pár káblíkov
- Atmega 8 Development board.
- USB to RS232 prevodník
- AVR STK500 ako programátor

Toľko tento článok. Pri vytváraní tohto článku som dospel k záveru, že Atmega8 je na niektoré moje nápady už prikrátka. Preto zrejme v najbližšom čase prejdem na iný MCU. Najlepšie už 32 bitový. Ešte sa skúsim pohrať s nejakými komunikačnými modulmi a skúsiť roztočiť DC-motor s Atmegou8 a potom prejdem na nejaký 32 bitový MCU.

Odporúčaná literatúra k digitálnym a analógovým filtrom:

1. IIR Filter design using the bilinear transform
2. Low Pass Filter Introduction

3. Filter – Based Algorithm for Metering applications
4. What is the difference between analogue and digital filters?
5. Filtrácia