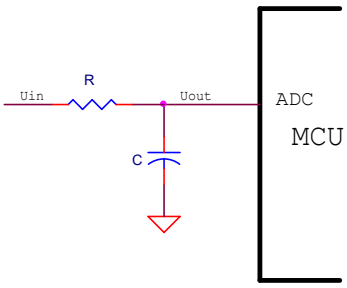
ПРАКТИЧНО И ЈЕДНОСТАВНО НИСКОФРКВЕНТНО ФИЛТРИРАЊЕ У АПЛИКАЦИЈАМА СА ОСМОБИТНИМ МИКРОКОНТРОЛЕРИМА

Честа је поотреба за филтрирањем улазних сигнала у микроконтролер (МЦУ), са циљем да се отклоне сметње пре увођења у АД конвертор.



Поступак: филтрирање па конверзија, може се заменити са: конверзија па филтрирање. Други поступак не захтева спољне компоненте, а омогућује лако прилагођавање филтрирања постојећим сметњама. На слици је НФ првог реда.

Ако посматрамо овај филтер у довољно кратком времену *ΔT* за које се може сматрати да су *Uin* и *Uout* константне вредности, и ако (ради једноставности) *Uin* заменимо са *X*, а *Uout* са *Y*, онда ће прираштај *Y* (*ΔY*) бити једнак количнику протекле количине електрицитета кроз R у кондензатор C подељено са вредношћу кондензатора C. То се може написати као:

$$\Delta Y = \frac{\frac{X-Y}{R}*\Delta T}{C} = \Delta T * \frac{X-Y}{R*C}$$

Ако се конверзија врши периодично, са периодом *ΔT*, онда ће филтрирана вредност бити:

$$Y_n = Y_{n-1} + \frac{X_n - Y_{n-1}}{\frac{\tau}{\Delta T}}$$

Или у практичном облику:

$$Y = Y + \frac{X-Y}{a}$$

где је временска константа филтра $\tau = \Delta T * a$. Згодно је da буде $a = 2^N$ тако да се дељење своди на шифтање удесно.

Овај филтер, осим за *Y* варијаблу не захтева ни једну додатну. Ово је, такозвани филтер са бесконачним импулсним одзивом (ИИР), али постоје и филтри са коначним импулсним одзивом (ФИР). ФИР филтер се базира на сумирању *N* вредности конверзије, а онда дељење суме са *N*.

Неке упоредне особине поменутих филтера:

- ИИР филтер троши мање РАМ од ФИР филтра;
- ФИР филтер је увек стабилан, док лоше пројектован ИИР може бити нестабилан;
- За исти ефекат филтрирања ИИР филтер је нижег реда од ФИР филтра.

Ове особине препоручују ИИР филтер за 8-о битне МЦУ.

За филтре вишег реда мора се користити много снажнији МЦУ, или по правилу ДСП.

Међутим, ако вежемо на ред два RC филтра са слике, добићемо филтер другог реда који ће бити стабилан. Овакав филтер је лак за реализацију, али вероватно неће бити најефикаснији филтер другог реда. Важно једа се може реализовати на 8-о битном МЦУ.

Следи приказ реализације оваквих филтера вишег реда. Почнимо од филтра другог реда:

$$y_n = C_0 \cdot x_n + C_1 \cdot x_{n-1} + C_2 \cdot x_{n-2} - D_1 \cdot y_{n-1}$$

C и *D* су константе које се одређују приликом пројектовања филтра. Да би упростили поступак пођимо од филтра првог реда:

$$y_n = y_{n-1} + \frac{x_n - y_{n-1}}{a} = y_{n-1} + \frac{1}{a} * x_n - \frac{1}{a} * y_{n-1} = \frac{1}{a} * x_n - (\frac{1}{a} - 1) * y_{n-1}$$

одаке следи да је:

$$C_0 = \frac{1}{a}$$

и

$$D_1 = C_0 - 1$$

Филтер другог реда се добија и везивањем на ред два RC филтра, која су приказана на слици, на ред. Такав филтер, можда неће имати најбоље перформансе, али ће бити бољи од филтра првог реда. На сличан начин се могу добити филтри трећег и четвртог реда. Пгактичне једначине за реализацију ових филтера су:

$$y1_n := C_0 \cdot x - D_1 \cdot y$$

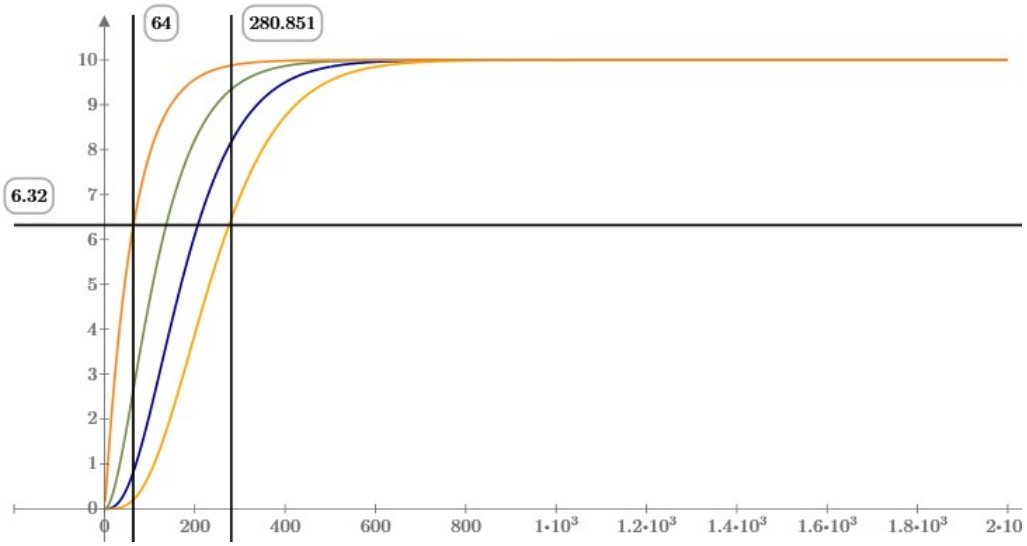
$$y2_n := C_0 \cdot y1_n - D_1 \cdot y$$

$$y3_n := C_0 \cdot y2_n - D_1 \cdot y$$

$$y4_n := C_0 \cdot y3_n - D_1 \cdot y$$

где је *x* уствари x_n.

Одзиви на одскочну функцију ових филтера су:

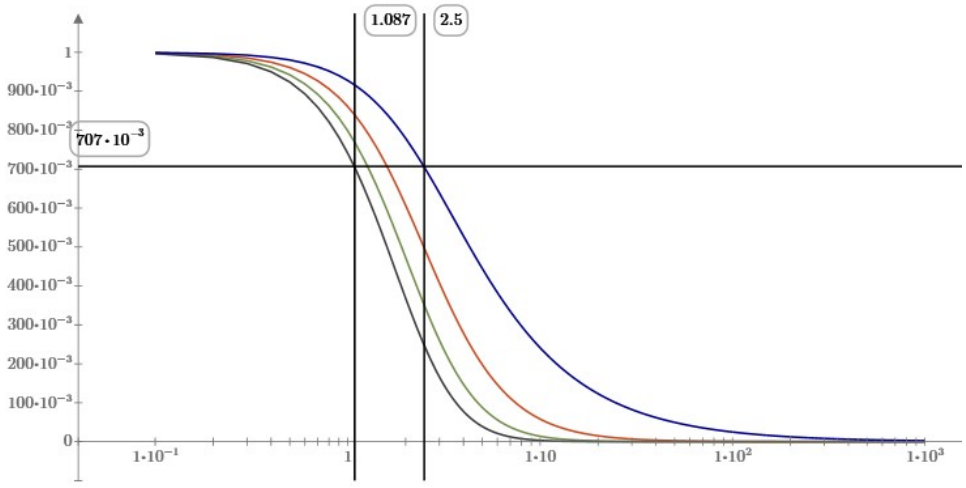


Филтри су пројектовани са a = 64, са конверзијом на *ΔT = 1ms*. На x оси је време ms, а на y оси је напон са максималном вредношћу 10V. Напон може да има и неку другу вредност, али ће графици бити исти. Када напон расте, у тренутку једнаком временској константи, он ће бити једнак:

$$10 * (1 - \frac{1}{e}) = 6.321$$

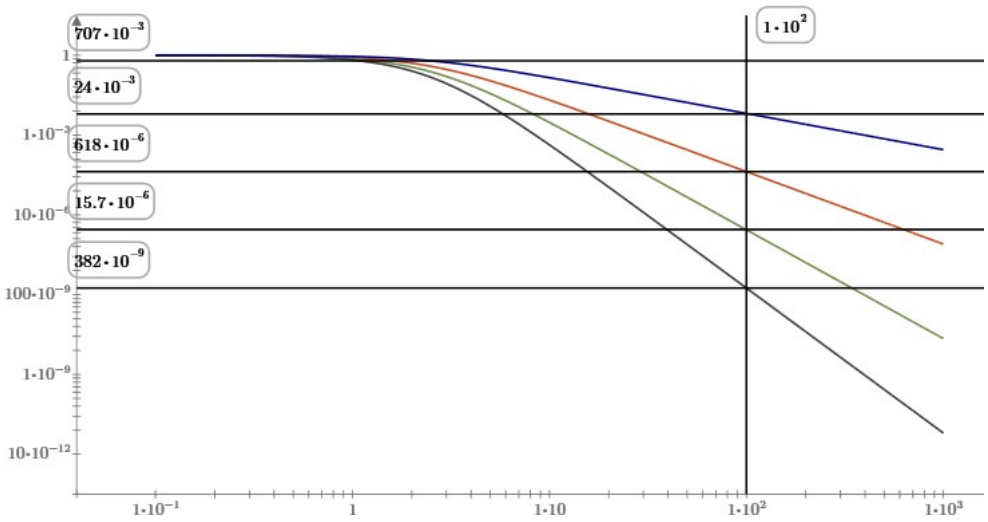
Тако очитавамо да је временска константа филтра првог реда 64ms, а филтра четвртог реда 281ms. Временске константе остала два филтра су између ових вредности.

Фреквентни пропусни опсези филтера су на следећој слици:



X оса је логаритамска, а *H* је преносна функција. Гранична фреквенција филтра птвог реда је 2.5Hz, а четвртог реда 1.1Hz. Филтер другог реда има граничну фреквенцију од 1.6Hz. Филтер другог реда има ужи пропусни опсег од филтра првог реда за 0,9Hz, а филтер четвртог реда има ужи опсег од филтра другог реда за 0.5Hz. Већи је добитак, када се пређе са филтра првог реда на филтер другог реда него при преласку са филтра другог реда на филтер четвртог реда.

Међутим, стварни добитак се има на потискивању сметњи. Тако, на пример, потискивање сметњи које су на 100Hz се виде се на слици на којој је и у оса логаритамска:



1. Филтер првог реда -16dB (0,025);
2. Филтер другог реда -32dB (0.00062);
3. Филтер трећег реда -48dB (0.000015);
4. Филтер четвртог реда -64dB (0.00000038).

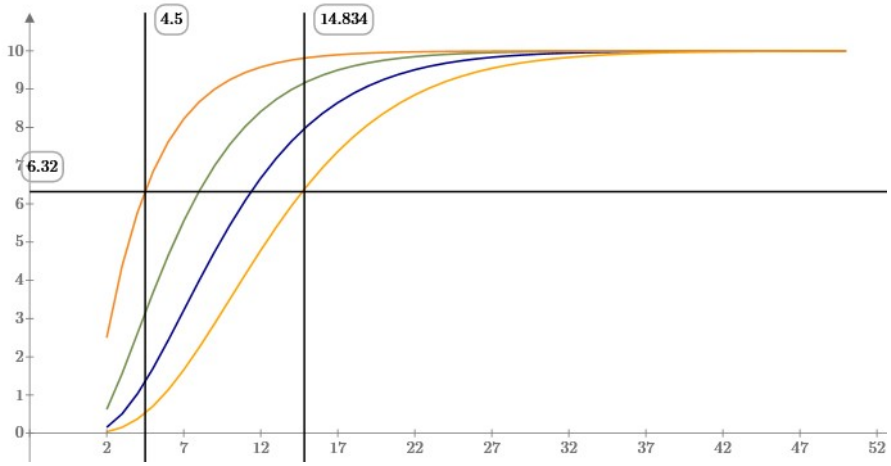
У загради је слабљење филтра на поменутој фреквенцији.

Ово нису идеални резултати, али имајући у виду једноставност реализације, што је био примарни циљ, они су задовољавајући.

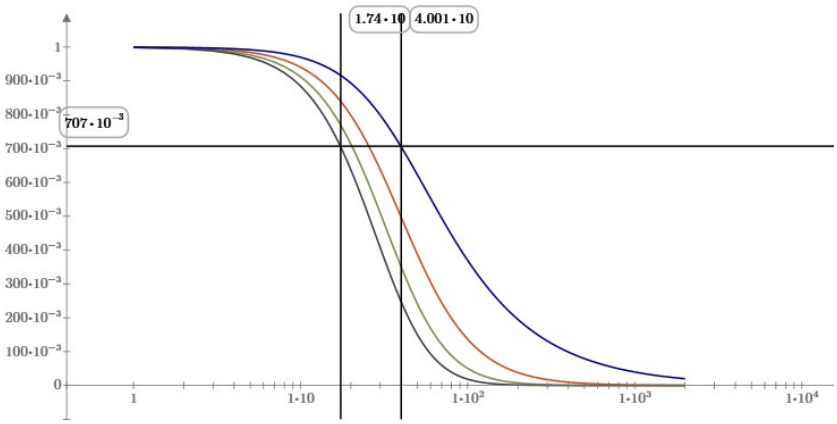
У реализацији се користе 4 варијабле, и само једна константа којом се подешава филтер, константа a.

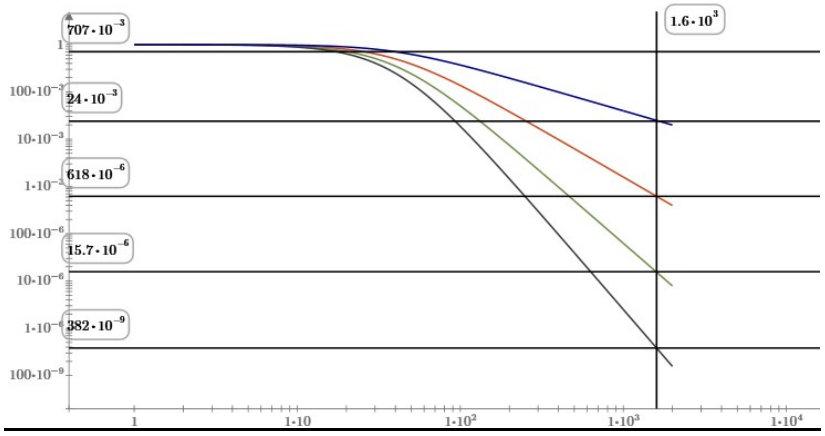
За филтер другог реда постоје 5 константе, а за филтер четвртог реда 9 константе. Оне се могу подесити да дају оптималан филтер, али тешко да би могле да се представе у облику степена двојке, па би израчунавање филтра захтевало пуно процесорског времена.

Следећа три графика су за константе $a = 4$ и $\Delta T = 1ms$. Коментар није потребан.



Временска оса је у ms, а фреквентна у Hz.





Пример могуће реализације.

```
#define Nsh 6  // a = 64

word X;
word Y1 = 0;
word Y2 = 0;
word Y3 = 0;
word Y4 = 0;
word Nout;
//---------------------------
void FilterIreda(word * Y,word X)
{
        *Y += (int)(X - *Y) >> Nsh;
}
//---------------------------
void TI1_OnInterrupt(void)             // poziva se u interrupt-u na 1ms
{
        FilterIreda(&Y1,X);
        FilterIreda(&Y2,Y1);
        FilterIreda(&Y3,Y2);
        FilterIreda(&Y4,Y3);

        Nout = Y3 >> 6;

        AD1_GetValue16(&X);
}
```

Функција ***TI1_OnInterrupt*** се позива периодично у некој тајмерској сервисној функцији која обрађује догађаје на 1ms. Са МЦУ-ом фамилије 9S08 на 19.6608MHz израчунавање траје 42.7µs.

Обзиром да домиирају математичке операције, ово израчунавање филтра на асемблеру не би драстично скратило време. Асемблерска верзија истог програма, написана „из прве руке “ би била:

```
const
  Nsh = 6; // a = 64
var
  X, Y1, Y2, Y3, Y4 : word;
  Nout : word;
  ADC_R : word absolute $14;
//-------------------------
procedure FilterIreda(Y:word;X:word);
var
  priv : word;
  br : byte;
begin
  tshx;

  ldaa Nsh;
  staa x[br-1];

  ldaa x[X.1-1];
  suba x[Y.1-1];
  staa x[priv.1-1];

  ldaa x[X-1];
  sbca x[Y-1];

  ldx x[priv.1-1];    //A:X

  repeat
    asra;
    rorx;
  until decr(s[br]) =0;

  staa s[priv];
  txa;
  tshx;

  adda x[Y.1-1];
  staa x[Y.1-1];
  ldaa x[priv-1];
  adca x[Y-1];
  staa x[Y-1];
end;

//-------------------------
procedure TI1_OnInterrupt;
var
  br : byte;
begin
  FilterIreda(Y1 reload,X);
  FilterIreda(Y2 reload,Y1);
  FilterIreda(Y3 reload,Y2);
  FilterIreda(Y4 reload,Y3);

  ldaa 6;
  staa s[br];

  ldhx [Y4];
  pshh;
  pula;

  repeat
    asra;
    rorx;
  until decr(s[br]) =0;

  psha;
  pulh;
  sthx [Nout];

  ldhx [ADC_R];
  sthx [X];
end;
```

Под истим условима трајање израчунавања је 36.9μs.