

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И
РАДИОЭЛЕКТРОНИКИ
КАФЕДРА РАДИОТЕХНИЧЕСКИХ СИСТЕМ

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

«Цифровая фильтрация и реализация фильтров на микропроцессорах семейства
dsPIC33F»

Выполнил:

ст. гр. 840101

Коржун В.В.

Проверил:

Казека А.А.

ЦЕЛЬ РАБОТЫ:

Ознакомится с программами dsPIC Filter Design и dsPICworks.

С помощью программы dsPIC Filter Design спроектировать цифровой фильтр.

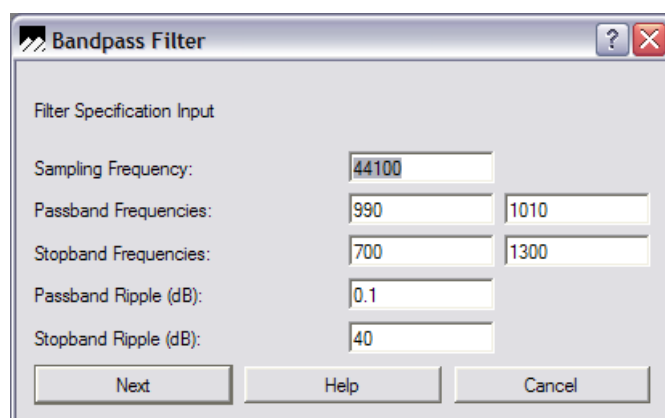
С помощью программы dsPICworks произвести фильтрацию зашумленного сигнала.

Реализовать цифровой фильтр на микропроцессоре семейства dsPIC33F

ХОД РАБОТЫ:

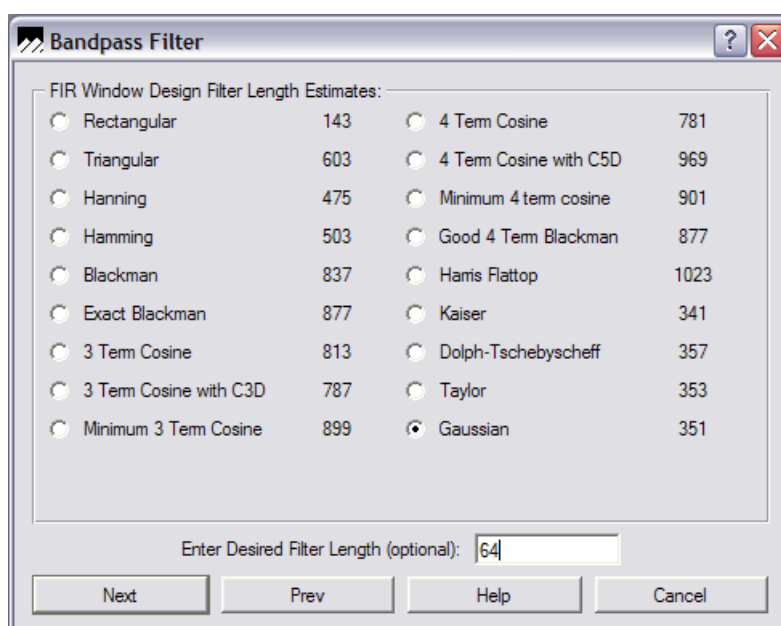
1. Проектирование фильтра

Спроектируем КИХ-фильтр Чебышева со средней частотой равной 1000 Гц и характеристиками, приведенными на рисунке 1.



Filter Specification Input	
Sampling Frequency:	44100
Passband Frequencies:	990 1010
Stopband Frequencies:	700 1300
Passband Ripple (dB):	0.1
Stopband Ripple (dB):	40

Рисунок 1. – Параметры проектируемого фильтра.



FIR Window Design Filter Length Estimates:	
<input type="radio"/> Rectangular	143
<input type="radio"/> Triangular	603
<input type="radio"/> Hanning	475
<input type="radio"/> Hamming	503
<input type="radio"/> Blackman	837
<input type="radio"/> Exact Blackman	877
<input type="radio"/> 3 Term Cosine	813
<input type="radio"/> 3 Term Cosine with C3D	787
<input type="radio"/> Minimum 3 Term Cosine	899
<input type="radio"/> 4 Term Cosine	781
<input type="radio"/> 4 Term Cosine with C5D	969
<input type="radio"/> Minimum 4 term cosine	901
<input type="radio"/> Good 4 Term Blackman	877
<input type="radio"/> Harris Flattop	1023
<input type="radio"/> Kaiser	341
<input type="radio"/> Dolph-Tschebyscheff	357
<input type="radio"/> Taylor	353
<input checked="" type="radio"/> Gaussian	351

Enter Desired Filter Length (optional): 64

Рисунок 2. –Выбор окна и длины фильтра.

Фильтр имеет показанные на рисунке 3 характеристики. На рисунках 4 и 5 показана частотная характеристика соответственно в линейном и логарифмическом масштабе.

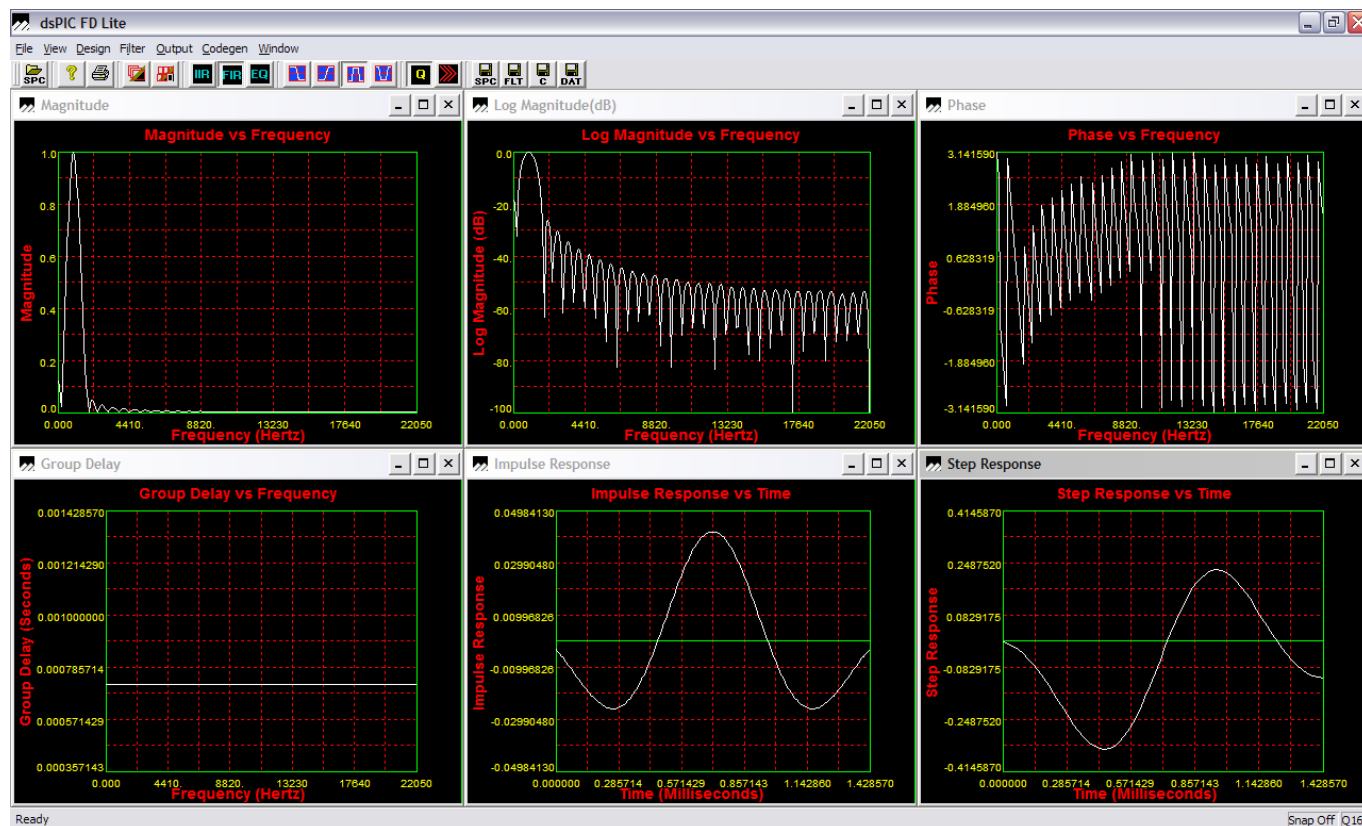


Рисунок 3. – Рабочее окно программы dsPIC Filter Design

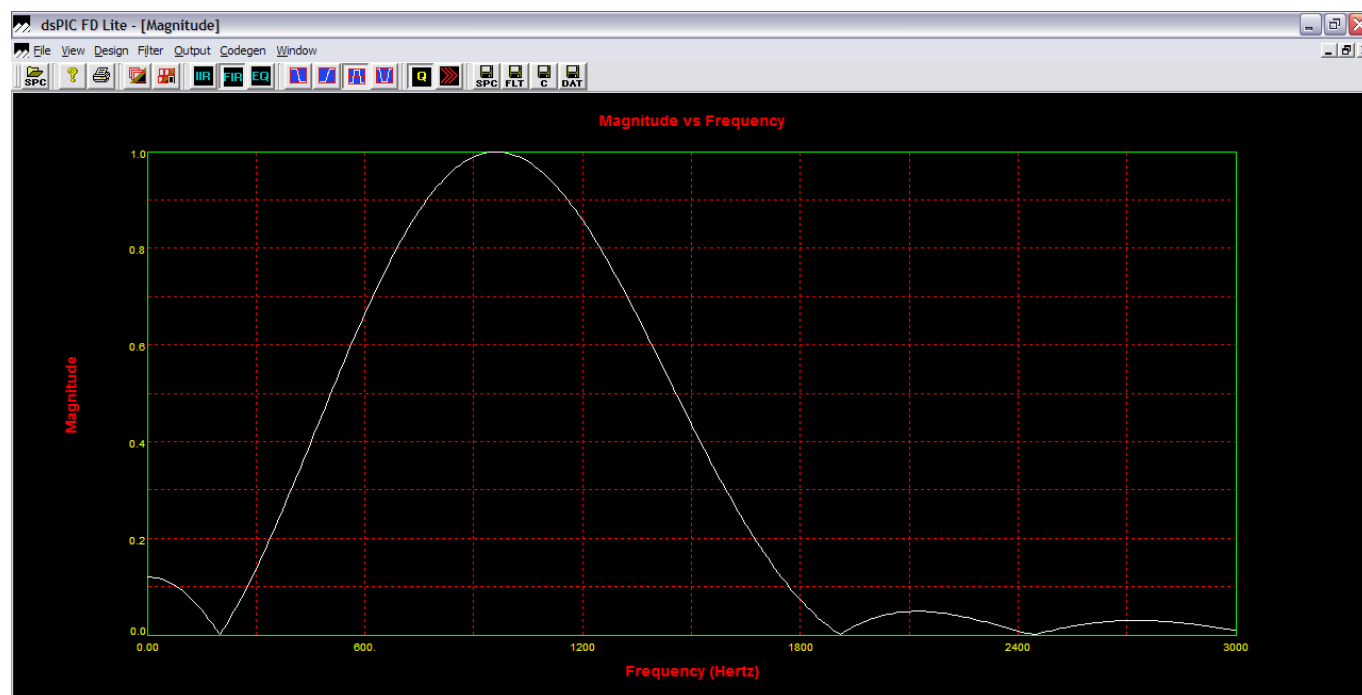


Рисунок 4. – Частотная характеристика фильтра в линейном масштабе.

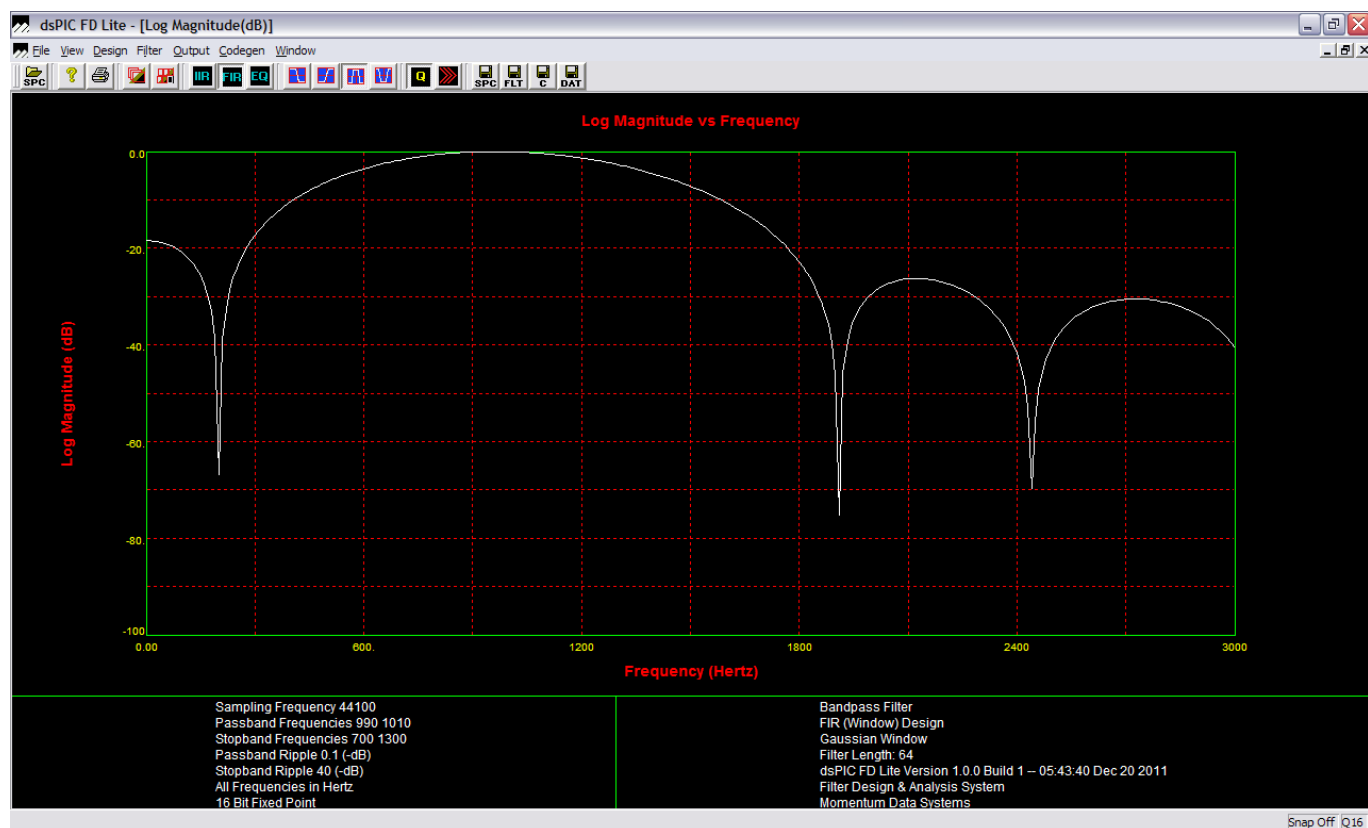


Рисунок 5. – Частотная характеристика фильтра в логарифмическом масштабе.

2. Фильтрация синусоидального сигнала

Сначала зададим синусоидальный сигнал с частотой сигнала равной 1000Гц и частотой дискретизации равной 44.1 кГц (рис. 6), а также аддитивный гауссовский шум. Осциллограммы сигналов приведены на рисунках 7 и 8.

Generating Sinusoidal Wave

Signal Frequency: 1000

Sampling Rate: 44100

Number of Samples: 1024

Angular Phase Delay: 0

Peak Amplitude (from zero): 1

DC offset: 0

Output File Format: Binary

Output Number Type: 32 bit floating point

Random Noise Type: no noise

Frequency Unit: ☒ Hertz ☐ Radians/second

Phase unit: ☒ Degrees ☐ Radians

Output File Name: output.tim

Help Ok Cancel

Рисунок 6. – Параметры синусоидального сигнала.

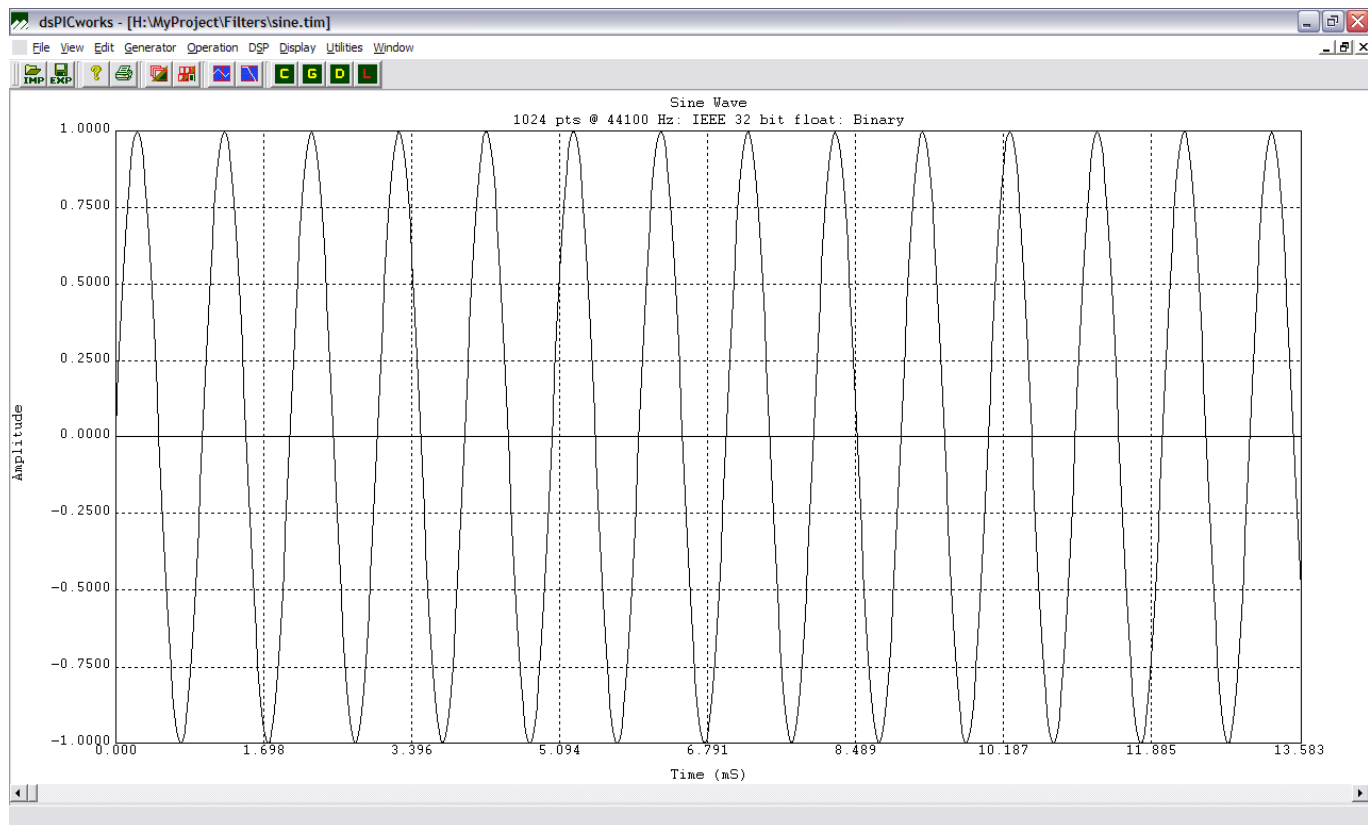


Рисунок 7. – Синусоидальной сигнал частоты 1000Гц.

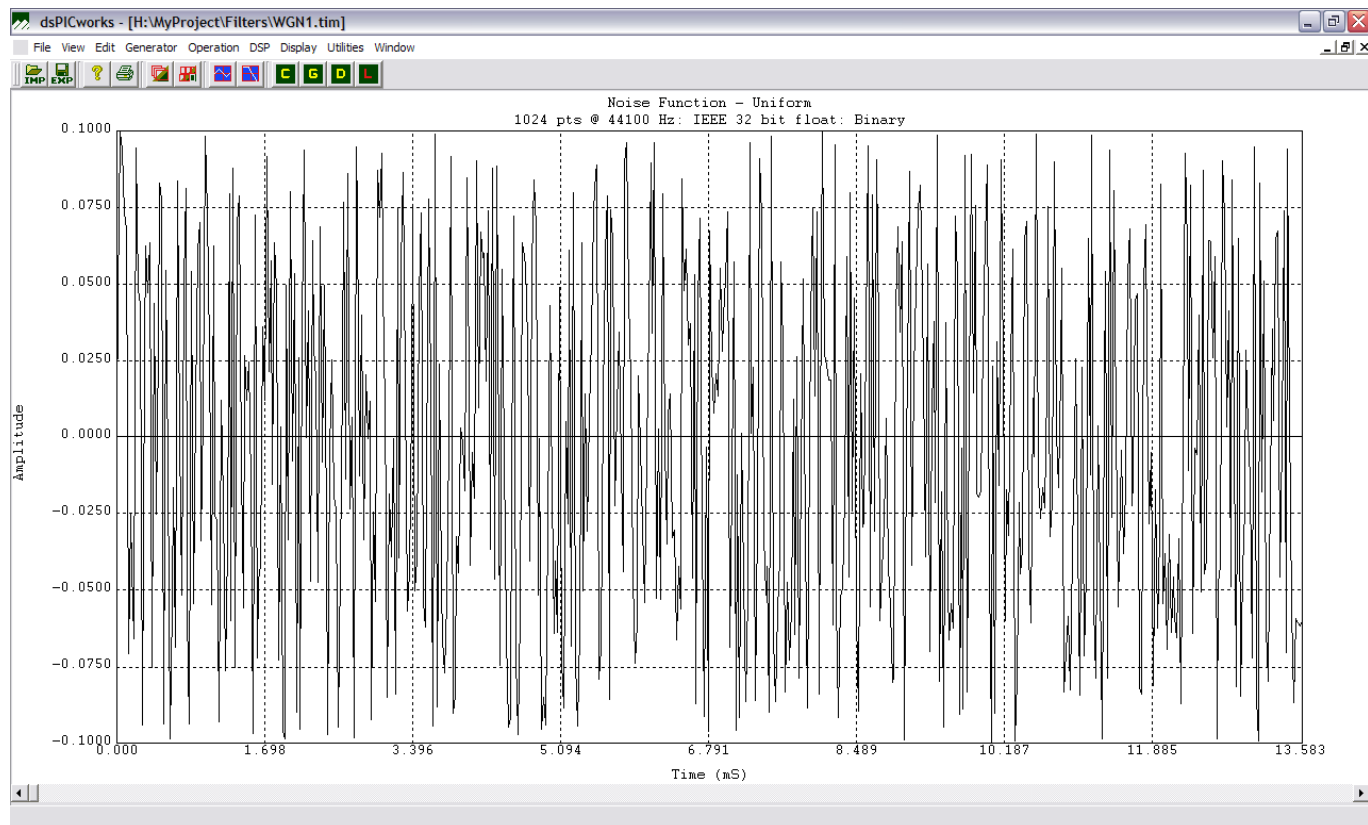


Рисунок 8. – Аддитивный белый гауссовский шум.

Теперь создадим зашумленный сигнал, путем суммирования сигналов, осциллограмма которого показана на рисунке 10.

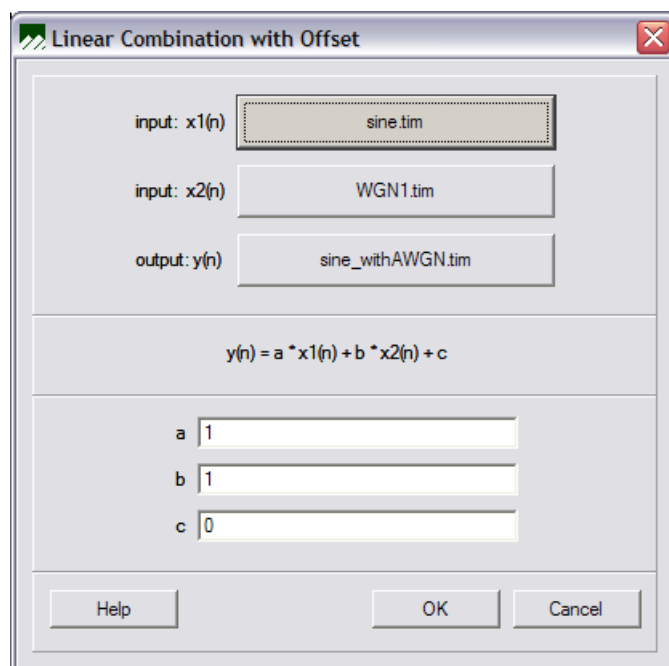


Рисунок 9. – Процесс суммирования двух сигналов.

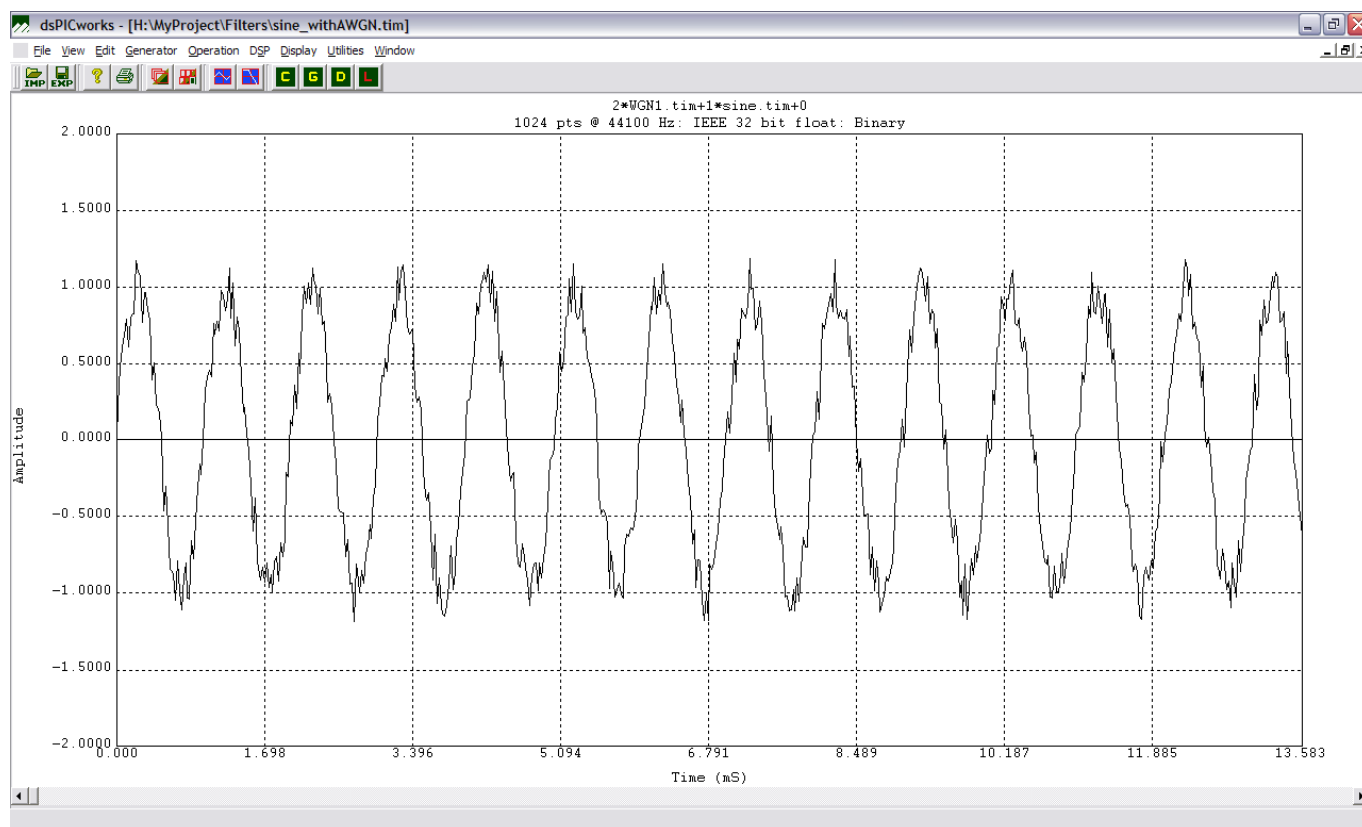


Рисунок 10. – Зашумленный синусоидальный сигнал.

Теперь непосредственно выполним процесс фильтрации с коэффициентами полученными ранее. Процесс фильтрации показан на рисунке 11. Выходной отфильтрованный сигнал показан на рисунке 12.

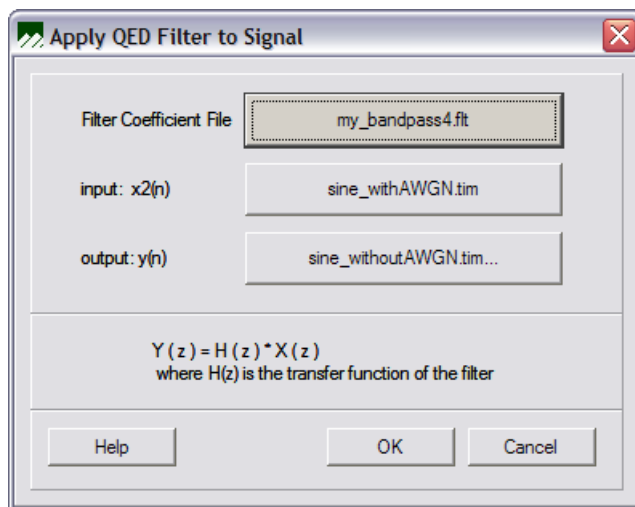


Рисунок 11. – Процесс фильтрации.

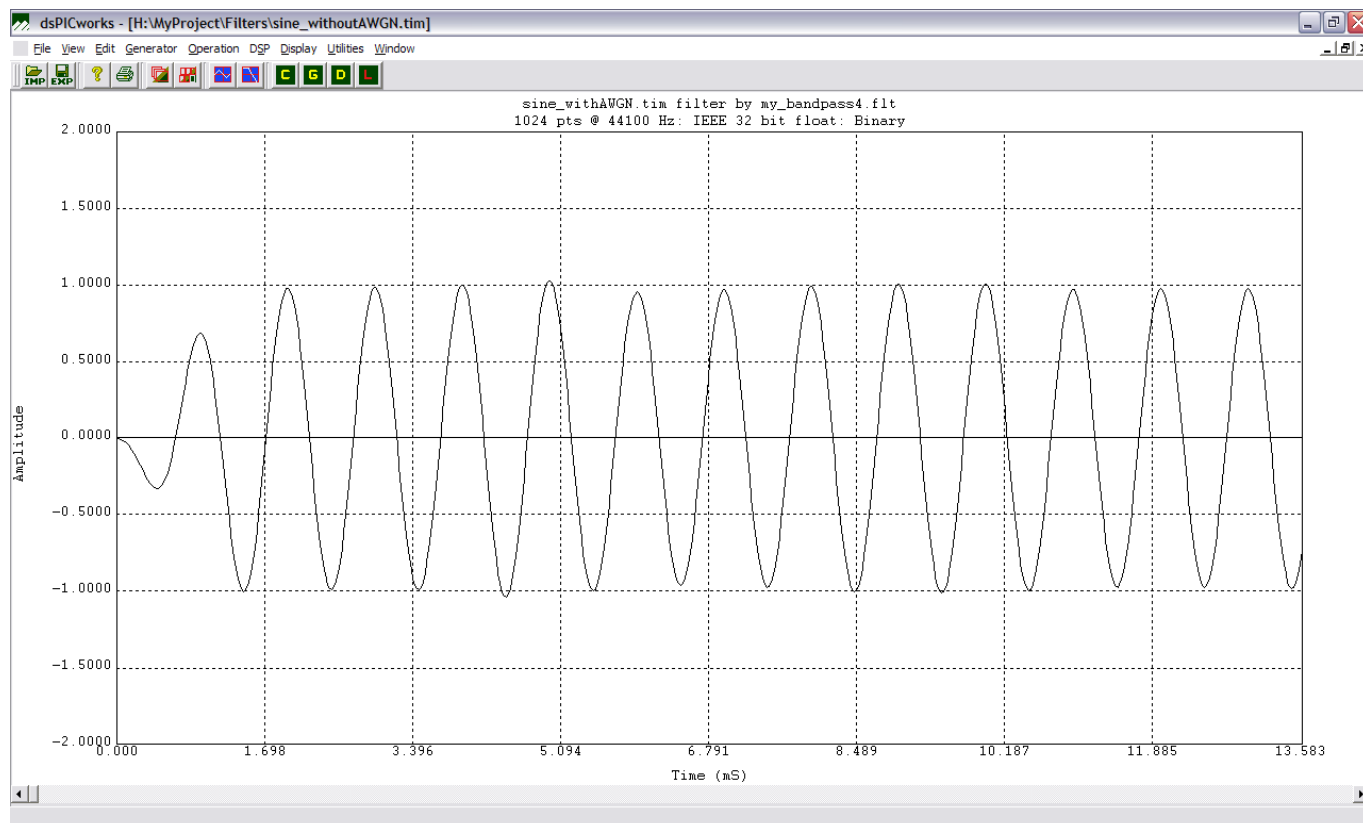


Рисунок 12. – Сигнал на выходе фильтра.

3. Реализация фильтрация на dsPIC33F(листинг кода)

```
#include "p33Fxxxx.h"

#include "dsp.h"

int incr;

int countCycles;

extern void initOC1(void);

extern void __attribute__((__interrupt__)) _T20Interrupt(void);


_FOSCSEL(FNOSC_FRC);

_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_XT);

_FWDT(FWDTEN_OFF);

_FPOR(FPWRT_PWR1);

_FGS(GCP_OFF);

#define BLOCK_LENGTH      256

#define LOWPASSFILTER

#define FILTERCOEFFS_IN_PROGMEM

extern fractional square1k[256];

#ifdef LOWPASSFILTER

    #ifdef FILTERCOEFFS_IN_PROGMEM

        extern FIRStruct lowpassexample_psvFilter;

    #else

        extern FIRStruct lowpassexampleFilter;

    #endif

#else

    #ifdef FILTERCOEFFS_IN_PROGMEM

        extern FIRStruct bandpassexample_psvFilter;

    #else

        extern FIRStruct bandpassexampleFilter;

    #endif

#endif

fractional FilterOut[BLOCK_LENGTH] ;


int main(void) {

    PLLFBD=38;                                // M=40

    CLKDIVbits.PLLPOST=0;                      // N1=2
```

```

CLKDIVbits.PLLPRE=0;          // N2=2
OSCTUN=0;                     // Tune FRC oscillator, if FRC is used
TRISD=0xFF00;
incr=0;
countCycles=0;
initOC1();
RCONbits.SWDTEN=0;
__builtin_write_OSCCONH(0x03);
__builtin_write_OSCCONL(0x01);
while (OSCCONbits.COSC != 0b011);
    while(OSCCONbits.LOCK!=1) {};
        #ifdef LOWPASSFILTER
            #ifdef FILTERCOEFFS_IN_PROGMEM
                FIRDelayInit(&lowpassexample_psvFilter);
FIR(BLOCK_LENGTH,&FilterOut[0],&square1k[0],&lowpassexample_psvFilter);
            #else
                FIRDelayInit(&lowpassexampleFilter);
FIR(BLOCK_LENGTH,&FilterOut[0],&square1k[0],&lowpassexampleFilter);
            #endif
        #else
            #ifdef FILTERCOEFFS_IN_PROGMEM
                FIRDelayInit(&bandpassexample_psvFilter);
FIR(BLOCK_LENGTH,&FilterOut[0],&square1k[0],&bandpassexample_psvFilter);
            #else
                FIRDelayInit(&bandpassexampleFilter);
FIR(BLOCK_LENGTH,&FilterOut[0],&square1k[0],&bandpassexampleFilter);
            #endif
        #endif
    #endif

```

```

        while (1)
    {
        if (countCycles==10)
        {
            incr++;

            if(incr==256)
            {
                incr=0;
            }

            countCycles=0;
        }
    }
    return 0;
}

void __attribute__((__interrupt__)) _T2Interrupt( void )
{
    countCycles++;

    OC1RS = FilterOut[incr];

    IEC0bits.T2IE = 1;

    IFS0bits.T2IF = 0;
}

void initOC1(void)
{
    OC1CONbits.OCM = 0b000;

    OC1R = FilterOut[incr];

    OC1RS = FilterOut[incr];

    OC1CONbits.OCTSEL = 0;

    OC1CONbits.OCM = 0b110;

    T2CONbits.TON = 0;

    T2CONbits.TCS = 0;

    T2CONbits.TGATE = 0;

    T2CONbits.TCKPS = 0b00;

    TMR2 = 0x00;

    PR2 = 65535;

    IPC1bits.T2IP = 0x01;

    IFS0bits.T2IF = 0;

    IEC0bits.T2IE = 1;

    T2CONbits.TON = 1;
}

```

Результат можно проверить если построить сигнала по выборкам, полученным из окна Watch.

```
data=ReadList["FilterOut.txt"]
{0,-1,-4,-6,-4,3,17,33,44,42,20,-17,-58,-81,-72,-20,64,150,202,182,78,-91,-265,-367,-330,-
131,187,515,706,628,228,-424,-1112,-1520,-1300,-
172,1993,5111,8907,12978,16893,20310,23051,25125,26678,27918,29013,30019,30857,31341,31253,30
416,28756,26311,23210,19607,15624,11319,6698,1747,-3495,-8901,-14233,-19181,-23433,-26761,-
29073,-30442,-31069,-31204,-31066,-30759,-30244,-29351,-27853,-25535,-22284,-18127,-13243,-
7916,-
2483,2742,7525,11733,15339,18395,20994,23241,25211,26943,28425,29608,30408,30736,30505,29640,
28107,25896,23033,19557,15531,11014,6089,864,-4509,-9832,-14858,-19334,-23044,-25848,-27735,-
28815,-29298,-29434,-29440,-29421,-29342,-29036,-28244,-26708,-24243,-20810,-16512,-11584,-
6325,-
1031,4064,8825,13200,17197,20831,24106,26981,29389,31254,32503,32767,32767,32205,30715,28524,
25643,22112,18015,13489,8711,3869,-864,-5362,-9556,-13421,-16949,-20134,-22943,-25326,-27229,-
28625,-29530,-29998,-30097,-29863,-29270,-28207,-26497,-23959,-20469,-16039,-10840,-
5199,476,5789,10440,14302,17421,19979,22207,24303,26356,28321,30032,31263,31797,31486,30292,2
8273,25563,22326,18705,14803,10676,6348,1840,-2798,-7481,-12075,-16429,-20385,-23820,-26656,-
28877,-30511,-31616,-32241,-32411,-32108,-31272,-29820,-27677,-24805,-21218,-17002,-12290,-
7247,-
2042,3166,8245,13081,17577,21633,25161,28085,30356,31960,32767,32767,32767,32372,31106,29247,
26731,23513,19595,15049,10015,4691,-695,-5906,-10743,-15039,-18691,-21646,-23920,-25582,-
26756,-27589,-28235,-28804,-29325,-29725,-29835,-29427,-28267,-26187,-23139,-19214,-14621}
```

```
ListLinePlot[data,InterpolationOrder→4,PlotStyle→Thick]
```

