Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

Институт Радиотехники и электроники им. В.А. Котельникова

Кафедра Формирования и обработки радиосигналов

КУРСОВОЙ ПРОЕКТ

по дисциплине «Методы и устройства цифровой обработки сигналов»

Тема: «Цифровая обработка сигналов на микроконтроллере»

Студент, гр. ЭР-15-15 _____ Жеребин В.Р.
(подпись)

Руководитель, ст. преподаватель _____ Филатов В.А.
(подпись)

Москва

2018

СОДЕРЖАНИЕ

ЗАДАНИЕ

Цель проекта – познакомится с аппаратным построение цифровых систем обработки сигналов с использованием механизма прерываний и встроенных в микроконтроллер устройств ввода-вывода (АЦП, таймер, последовательный порт USART), практическое освоение приемов цифровой обработки сигналов с использованием микроконтроллера PIC18F2520 и выполнение программы с использованием отладчика PICkit. Индивидуальное задания предусматривает обработку отрезка сигнала длительностью 1 секунда, состоящего из 256 8-ми битных отчета, которые необходимо записать в массив ***A***. Результат обработки массива A сохранить в массив ***B***, а так же вывести значения массива B через последовательный порт с заданной скоростью.

1. САМОСТОЯТЕЛЬНАЯ ДОМАШНЯЯ ПРОРАБОТКА

1.1. Определение структуры, основных требований и параметров к микропроцессорной системе (МПС) обработки сигналов.

- Канал приема и канал передачи сигналов.
- Тактирование МПС.
- Задействованные аппаратные модули МК и связанные с ним внешние выводы.
- Питание МК.
- Необходимые внешние соединения, цепи обеспечения и согласования с периферийными устройствами.
- Функциональное распределение выводов микросхемы МК.

Для формирования массива A в МПС использую вход ***AN0*** АЦП, с которого по нажатии кнопки подлеченной к выводу (***RB0/INT0***), происходит прерывание основной программы, оцифровка аналогово сигнала с частотой 256 выборок в секунду и запись старших 8 бит в массив A. Во время формирования массива включается светодиод на ***RB2***. После записи 256 отчетов сигнала в массив A светодиод RB2 продолжает гореть и происходит возврат из прерывания в основную программу для обработки этого отрезка сигнала, при этом включается светодиод на ***RB3***. После завершения обработки и записи в массив B, включается светодиод на ***RB4*** и запускается передача массива по USART. По завершении передачи включается светодиод на ***RB5*** и ожидается нажатие кнопки на ***RB1***. Тактирование МПС происходит от внутреннего тактового генератора с стабильной частотой 4 МГц. Питание МК производится от постоянного напряжение +5В. Этот же источник питания используется и для подачи постоянной составляющей сигнала на вход АЦП.

1.2. Полная принципиальная схема МПС.

Полная принципиальная схема МПС представлена на рисунке 1.1

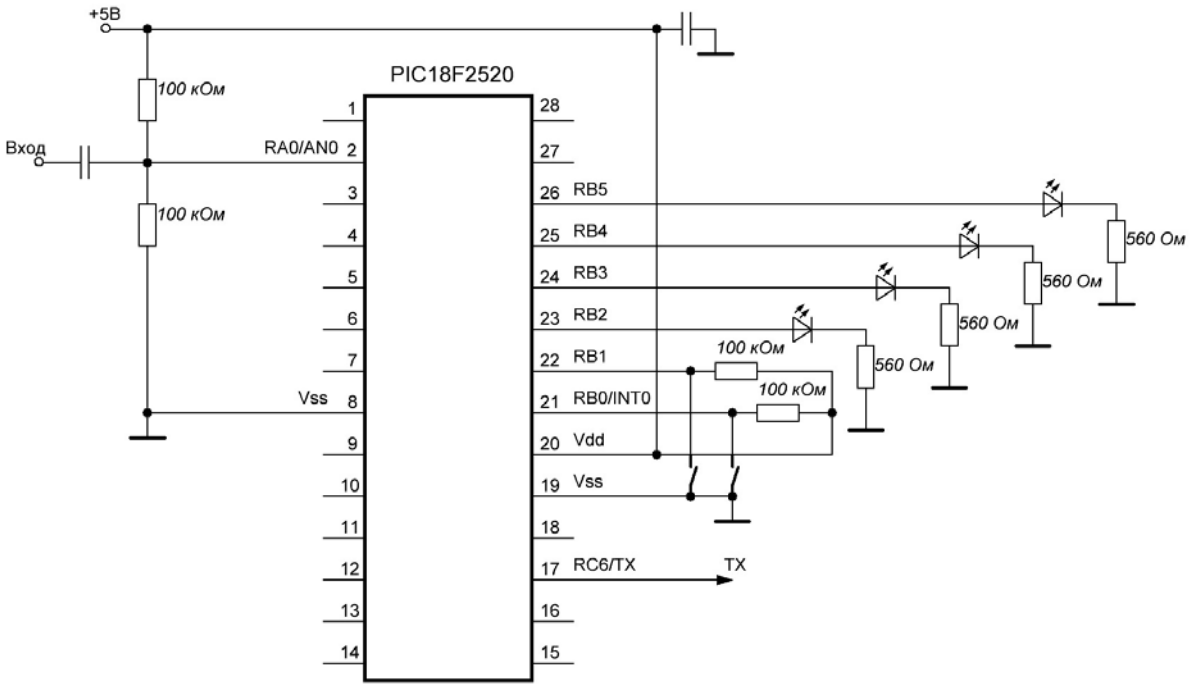
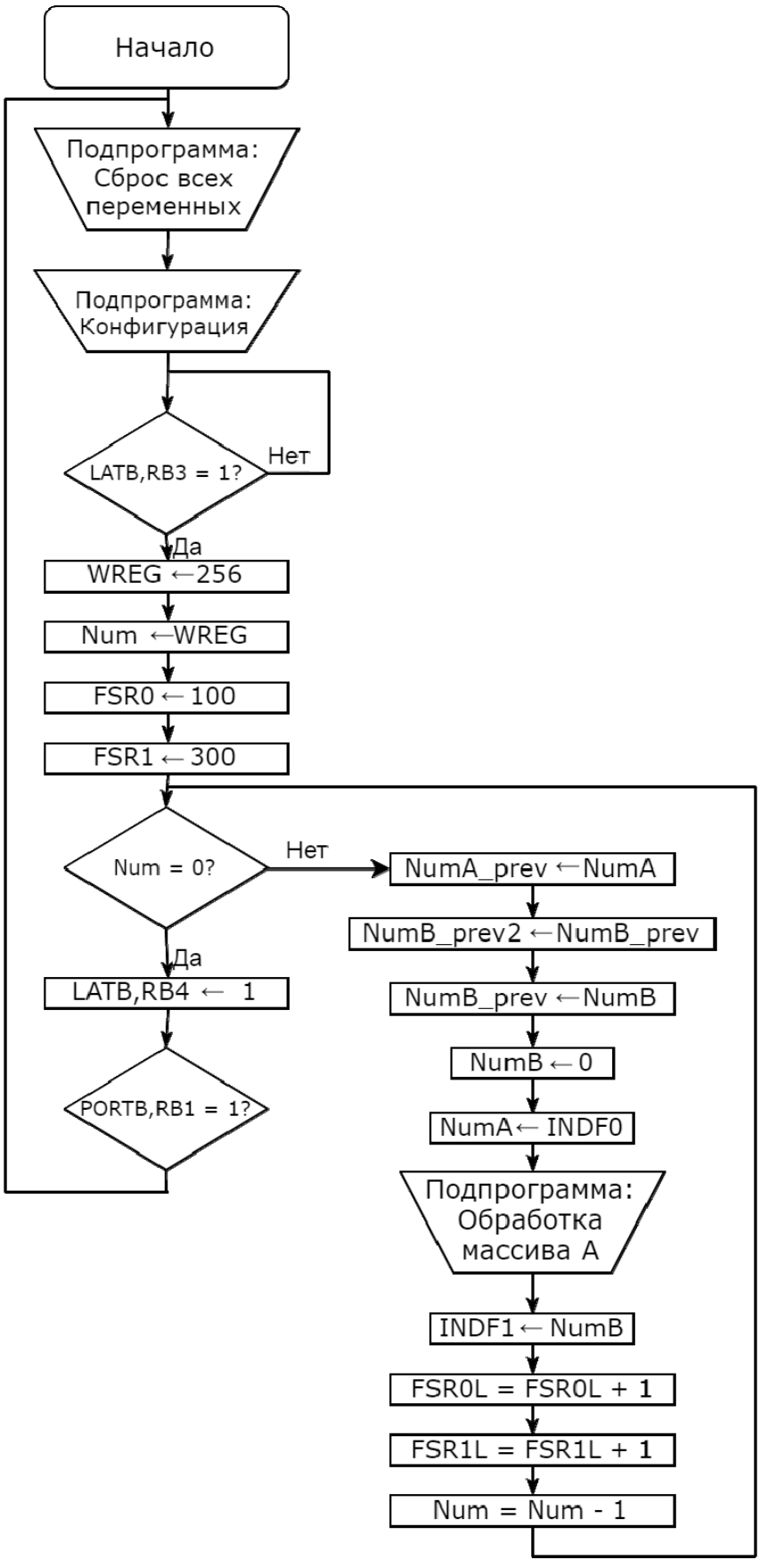


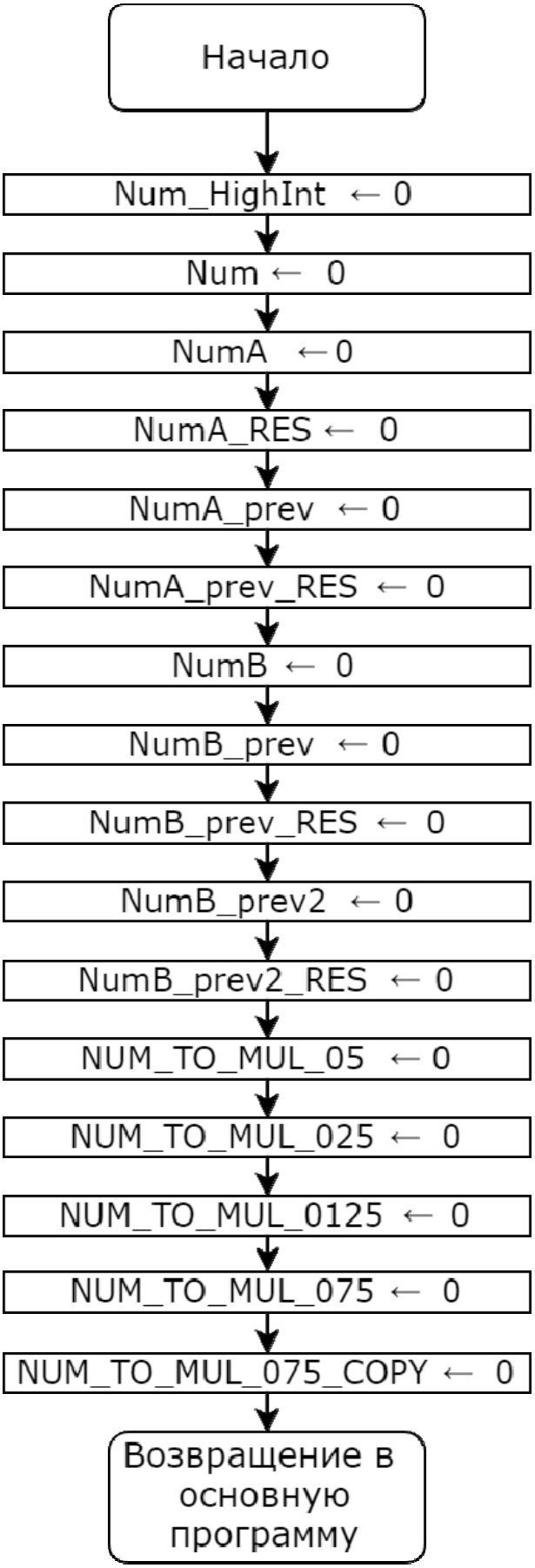
Рисунок 1.1 – Полная принципиальная схема МПС.

1.3. Алгоритмы основной программы и необходимых подпрограмм.

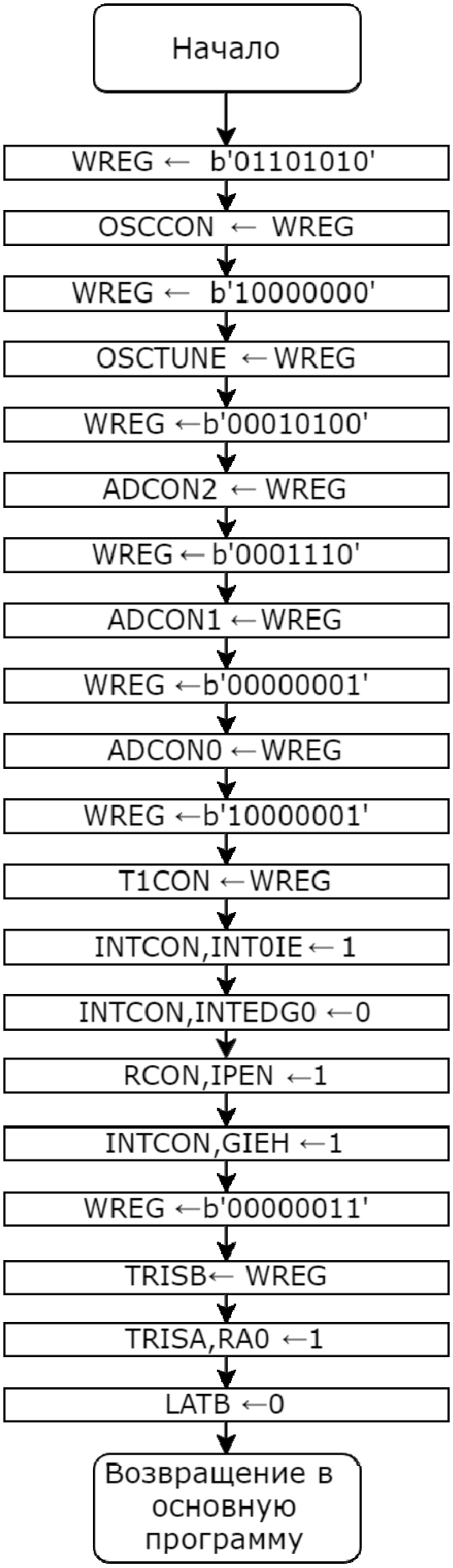
Алгоритм основной программы:



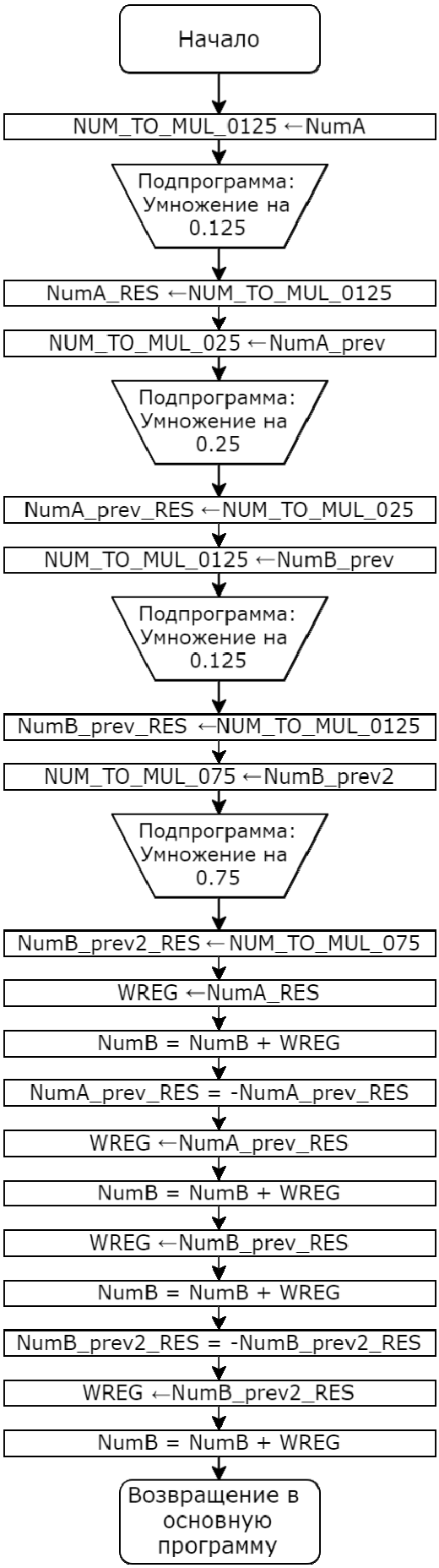
Алгоритм подпрограммы – Сброс всех переменных:



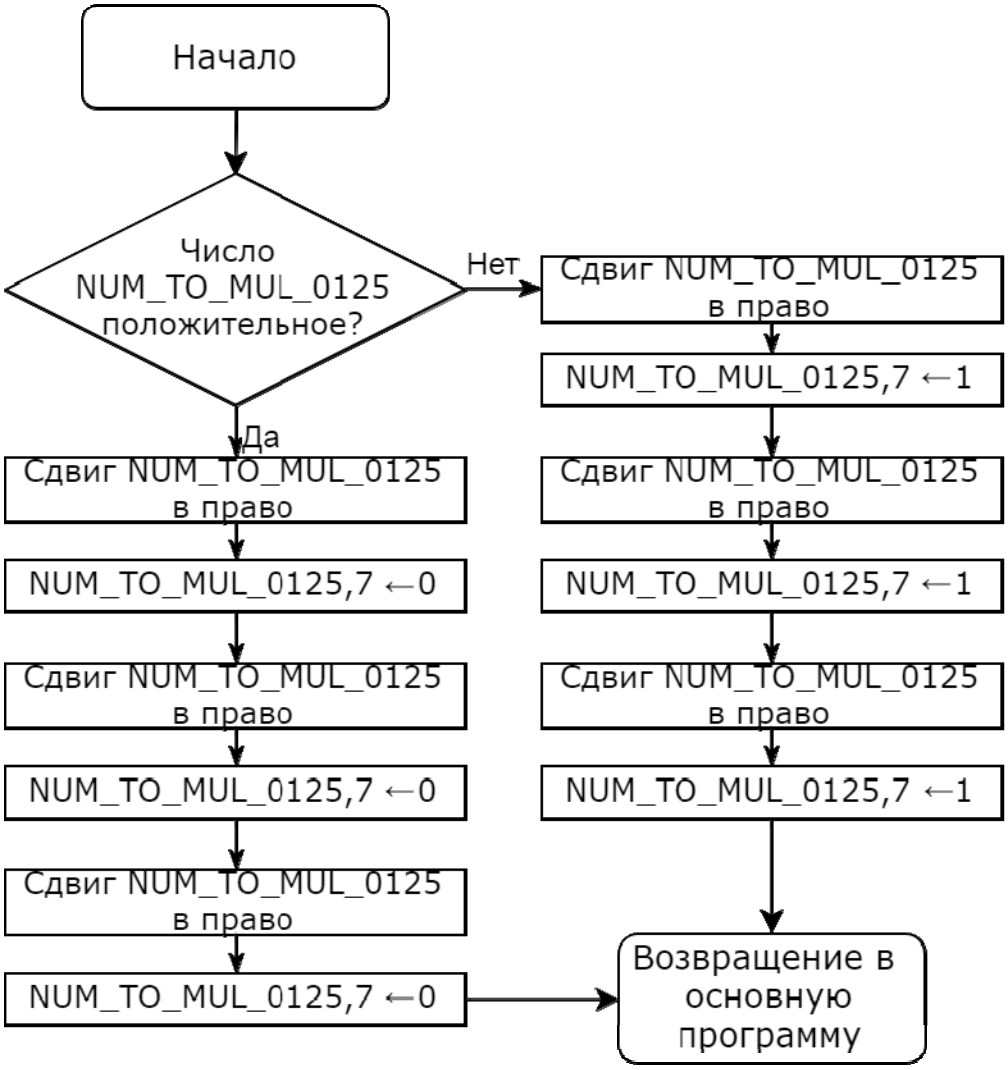
Алгоритм подпрограммы – Конфигурация:



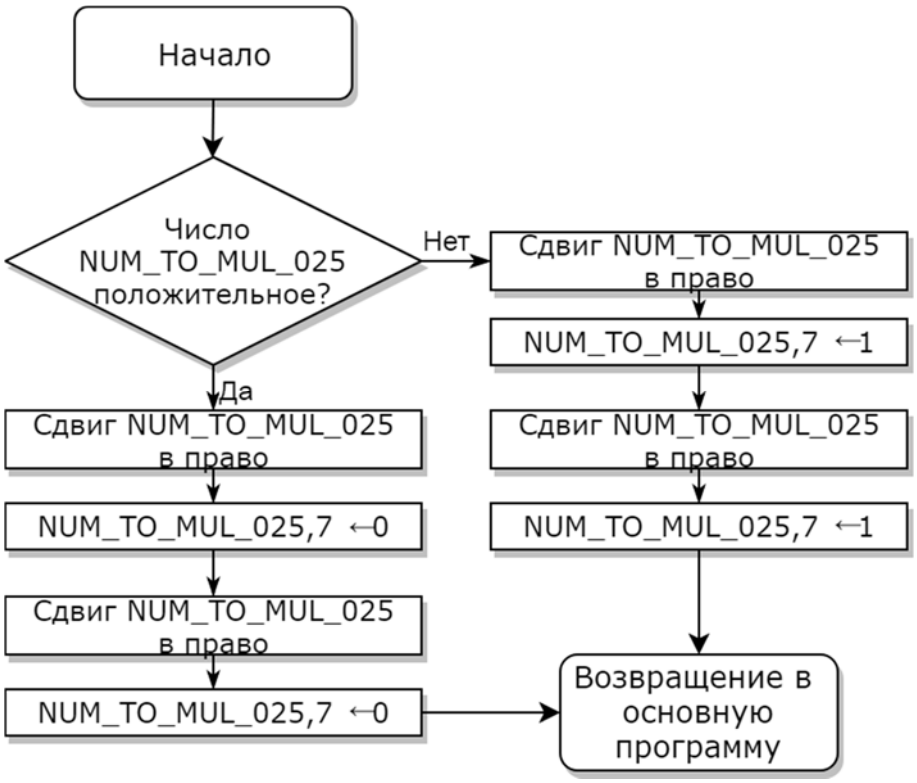
Алгоритм подпрограммы – Обработка массива A:



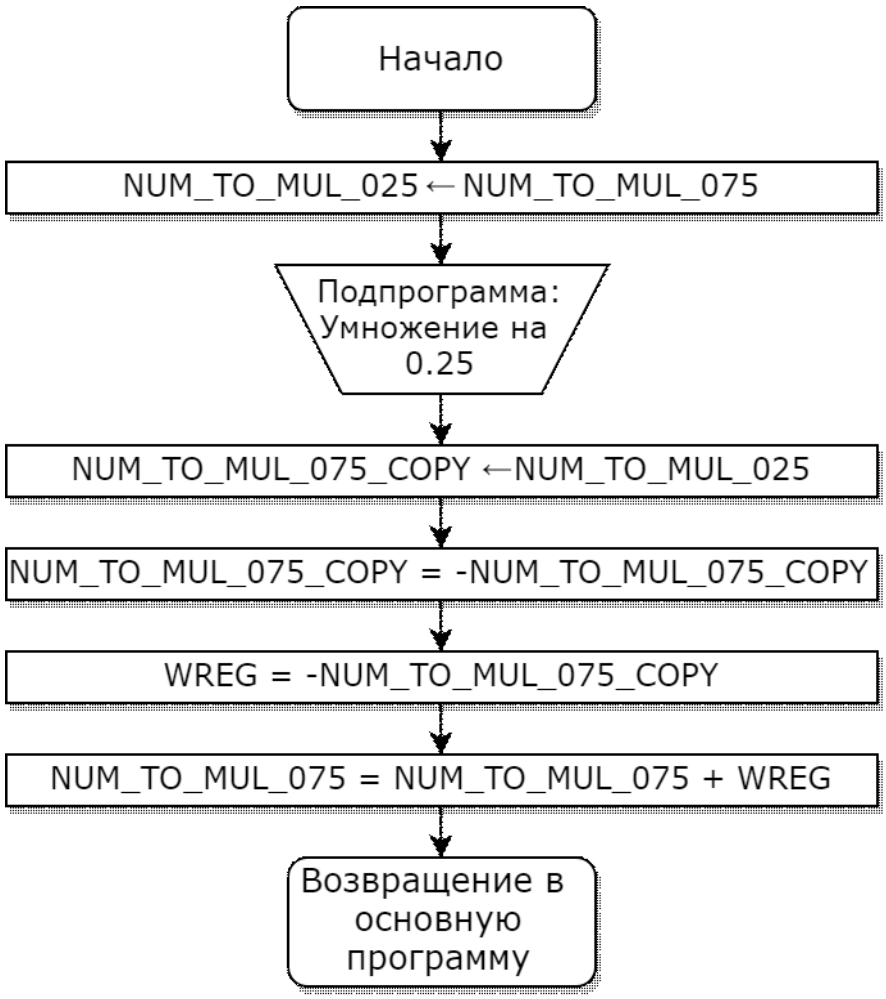
Алгоритм подпрограммы – Умножение на 0.125:



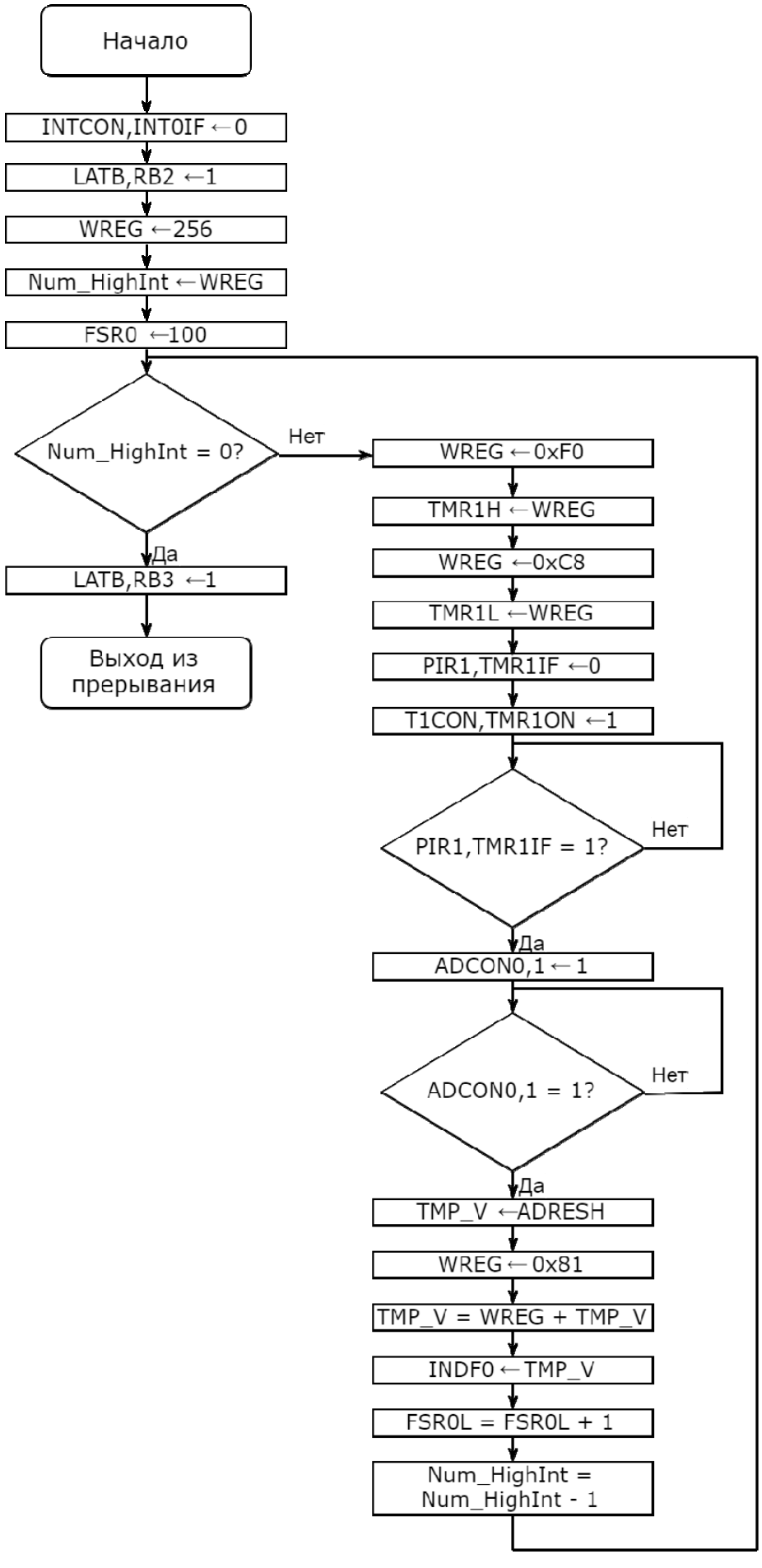
Алгоритм подпрограммы – Умножение на 0.25:



Алгоритм подпрограммы – Умножение на 0.75:



Алгоритм прерывания:



1.4. Теоритический анализ свойств цифрового фильтра.

Заданный алгоритм цифровой обработки сигнала:

$$B_i = 0{,}125A_i - 0{,}25A_{i-1} + 0{,}125B_{i-1} - 0{,}75B_{i-2}$$

Где A_i и B_i элементы массива **A** и **B** соответственно; i – номер элемента массива, изменяется от 0 до 255.

По заданному алгоритму можем получить разносное уравнение ЦФ:

$$Y = 0{,}125X - 0{,}25Xz^{-1} + 0{,}125Yz^{-1} - 0{,}75Yz^{-2}$$

Где Y – отсчеты выходного сигнала, X – отсчеты входного сигнала.

Проведем обратное Z-преобразование, для переходя к временным отсчетам:

$$y(nT) = 0{,}125x(nT) - 0{,}25x(nT - T) + 0{,}125y(nT - T) - 0{,}75y(nT - 2T)$$

По полученному выражению получаем импульсную и передаточную характеристики ЦФ. На рисунке 1.2 представлена импульсная характеристика ЦФ. На рисунке 1.3 представлена передаточная характеристика ЦФ.

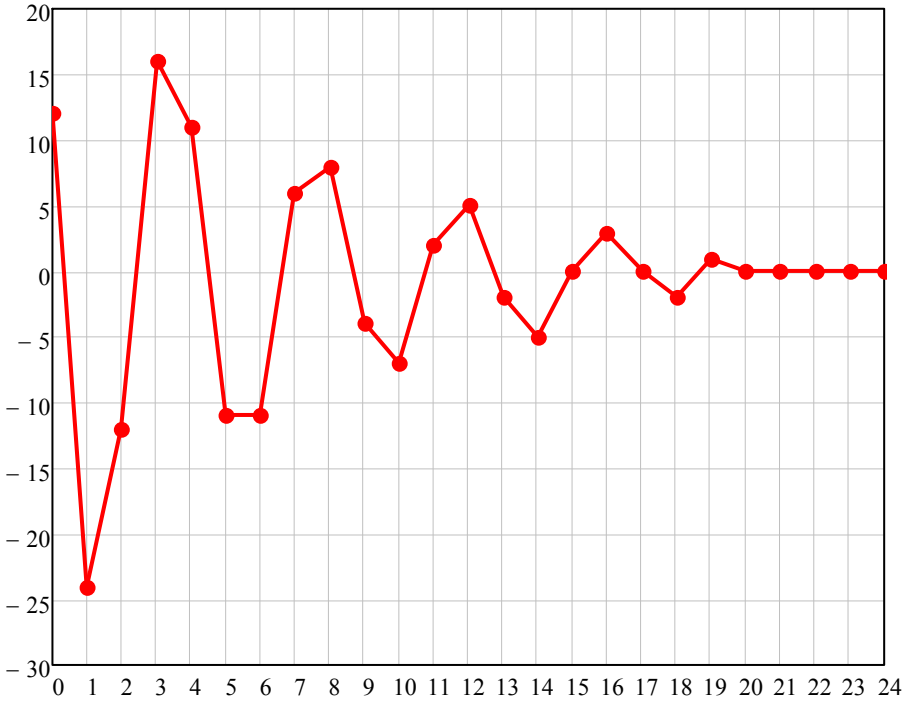


Рисунок 1.2 – Импульсная характеристика ЦФ.

Импульсная характеристика имеет колебательный затухающий характер. Следовательно данный фильтр – полосовой. Период колебания импульсной характеристики составляет 4 такта. При условии, что за 1 секунду АЦП обрабатывает 256 отчетов, резонансная частота фильтра составит $256/4 = 64$ Гц. Импульсная характеристика к 20 такту затухает, значит фильтр имеет конечную импульсную характеристику (КИХ).

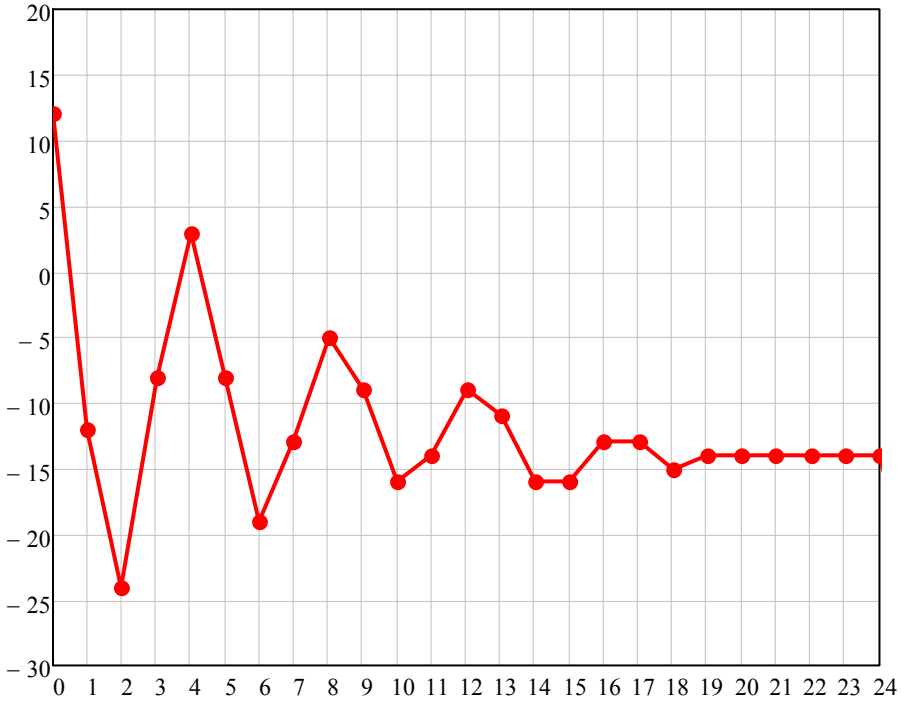


Рисунок 1.3 – Передаточная характеристика ЦФ.

Зная разносное уравнение, получим передаточную функцию ЦФ:

$$H(z) = \frac{0,125 - 0,25z^{-1}}{1 - 0,125z^{-1} + 0,75z^{-2}}$$

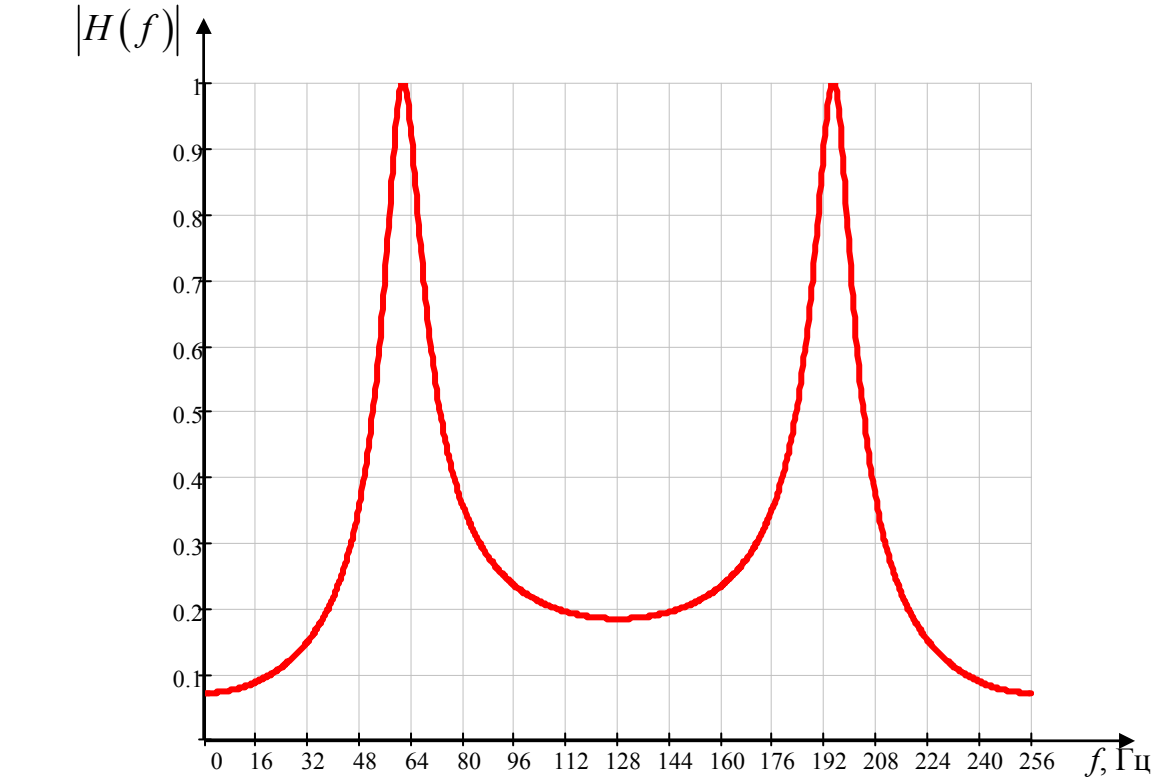
Для получение частотных характеристик, проведем замену: $z^{-m} = e^{-jm\varphi}$

$$H(\varphi) = \frac{0,125 - 0,25e^{-j\varphi}}{1 - 0,125e^{-j\varphi} + 0,75e^{-j2\varphi}}$$

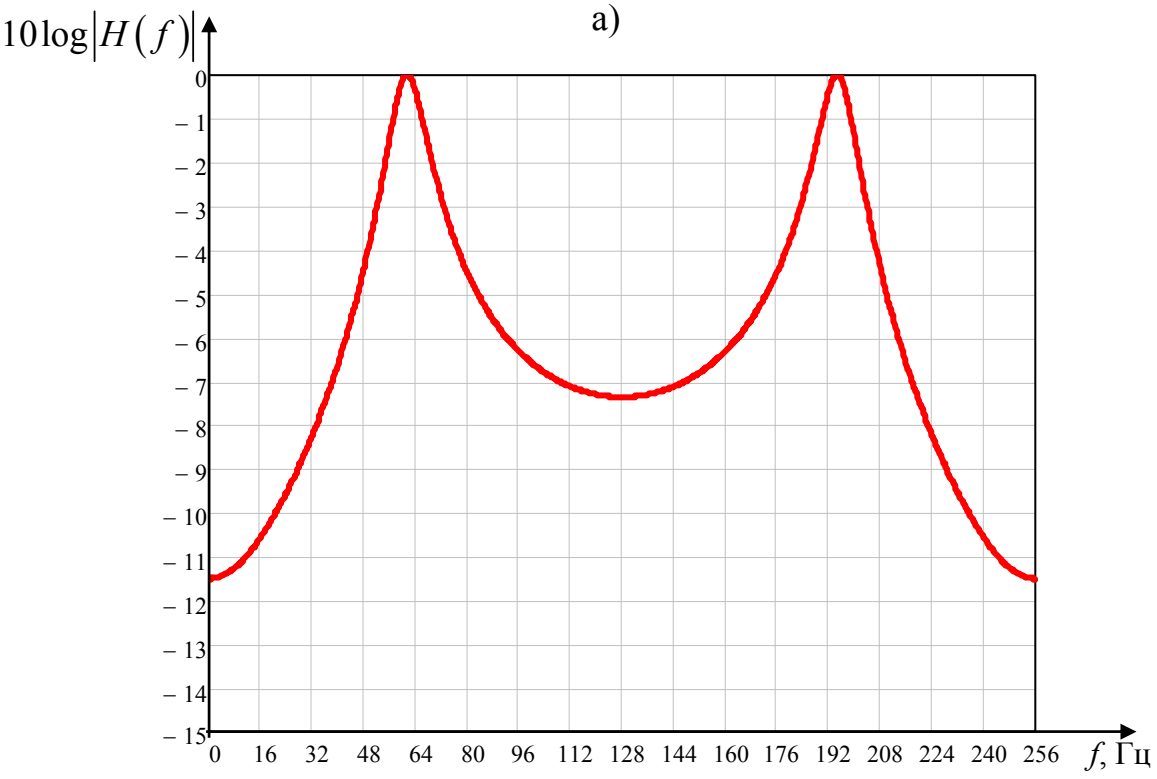
Где $\varphi = 2\pi\frac{f}{f_д}$ – нормированная цифровая частота, $f_д$ = 265 Гц – частота дискретизации. С учетом того, получим следующее выражение:

$$H(f) = \frac{0,125 - 0,25e^{-j2\pi\frac{f}{f_д}}}{1 - 0,125e^{-j2\pi\frac{f}{f_д}} + 0,75e^{-j4\pi\frac{f}{f_д}}}$$

По данной формуле построим нормированную амплитудно-частотную (рисунок 1.4 а), б)) и фазо-частотную (рисунок 1.5) характеристики ЦФ.



а)



б)

Рисунок 1.4 – Нормированная амплитудно-частотная характеристика ЦФ.
а) Шкала амплитуды в разах. б) Шкала амплитуды в децибелах.

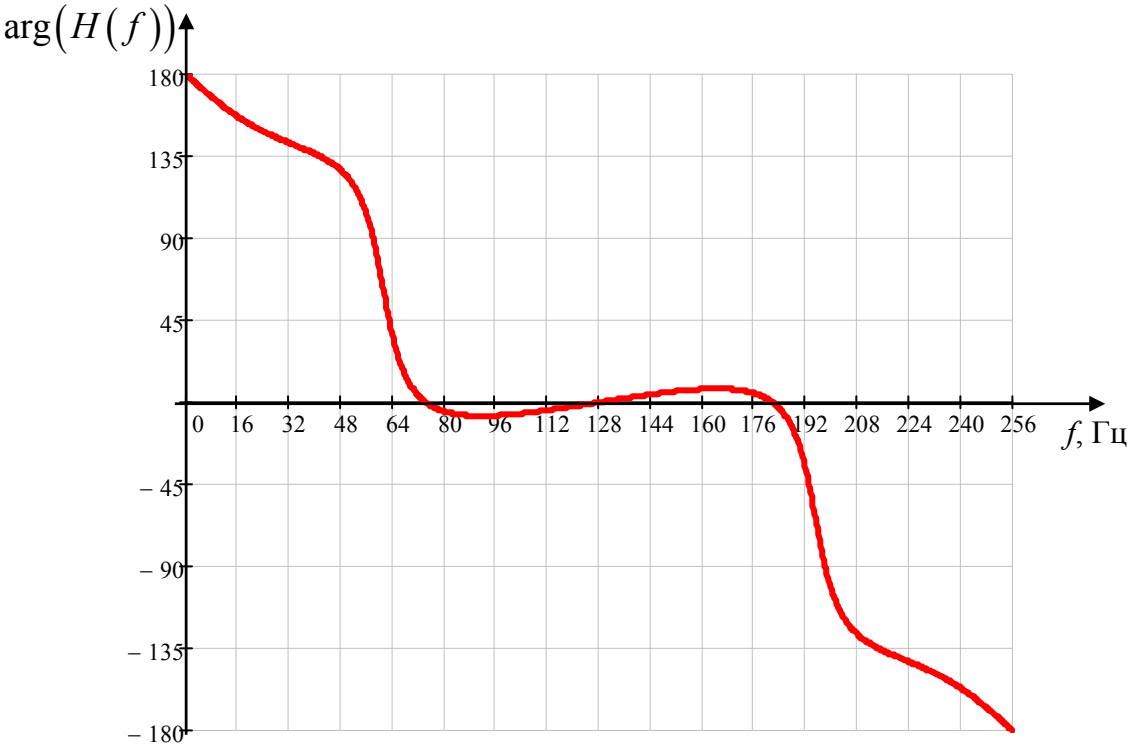


Рисунок 1.5 – Фазо-частотная характеристика ЦФ.

1.5. Реализация и отладка алгоритмов основной программы и подпрограмм на языке Ассемблера и в кодах микроконтроллера PIC18F2520. Полная программная документация.

Реализация алгоритма основной программы:

```
Main:
        CALL CLRF_ALL; Вызов подпрограммы: Сброс всех переменных
        NOP
        CALL CONF; Вызов подпрограммы: Конфигурация

M1:     BTFSS LATB,RB3; Ожидание включения светодиода на RB3
          BRA M1

        MOVLW .256; Запись 256 в WREG
        MOVWF Num; Запись из WREG в переменную Num
        LFSR FSR0,0x100; Задаем базовый адрес для массива A - 0x100
        LFSR FSR1,0x300; Задаем базовый адрес для массива B - 0x300

M2:     MOVFF NumA,NumA_prev; Перенос значения A(n) в A(n-1)
        MOVFF NumB_prev,NumB_prev2; Перенос значения B(n-1) в B(n-2)
        MOVFF NumB,NumB_prev; Перенос значения B(n) в B(n-1)
        CLRF NumB; Обнуление значения B(n)

        MOVFF INDF0,NumA; Перенос значения из памяти МК в A(n)
        CALL MAIN_FUNC; Вызов подпрограммы: Обработка массива A
        MOVFF NumB,INDF1; Перенос значения B(n) в память МК

        INCF FSR1L,F; Переход на следующий адрес массива B
        INCF FSR0L,F; Переход на следующий адрес массива A
        DECFSZ Num,F; Уменьшение на 1 количество операций, и проверка на нулевое значение
          BRA M2

        BSF LATB,RB4; Включение светодиода на RB4

M3:     BTFSC PORTB,RB1; Ожидание нажатия кнопки на RB1
          BRA M3

        GOTO Main; Перезапуск основной программы
```

Реализация алгоритма подпрограммы – Сброс всех переменных:

```
CLRF_ALL:
        CLRF Num_HighInt; Сброс переменной Num_HighInt
        CLRF Num; Сброс переменной Num
        CLRF NumA; Сброс переменной NumA
        CLRF NumA_RES; Сброс переменной NumA_RES
        CLRF NumA_prev; Сброс переменной NumA_prev
        CLRF NumA_prev_RES; Сброс переменной NumA_prev_RES
        CLRF NumB; Сброс переменной NumB
```

```
CLRF NumB_prev; Сброс переменной NumB_prev
CLRF NumB_prev_RES; Сброс переменной NumB_prev_RES
CLRF NumB_prev2; Сброс переменной NumB_prev2
CLRF NumB_prev2_RES; Сброс переменной NumB_prev2_RES
CLRF NUM_TO_MUL_05; Сброс переменной NUM_TO_MUL_05
CLRF NUM_TO_MUL_025; Сброс переменной NUM_TO_MUL_025
CLRF NUM_TO_MUL_0125; Сброс переменной NUM_TO_MUL_0125
CLRF NUM_TO_MUL_075; Сброс переменной NUM_TO_MUL_075
CLRF NUM_TO_MUL_075_COPY; Сброс переменной NUM_TO_MUL_075_COPY
CLRF TMP_V; Сброс переменной TMP_V
RETURN; Возвращение в основную программу
```

Реализация алгоритма подпрограммы – Конфигурация:

```
CONF:
    ;OSCILLATOR
    MOVLW b'01101010'; Fosc = 4 MHz
    MOVWF OSCCON
    MOVLW b'10000000'; F = 8 MHz
    MOVWF OSCTUNE
    ;ADC
    MOVLW b'00010100'; Включаем АЦП
    MOVWF ADCON2
    MOVLW b'00001110'; Аналоговый вход AN0, остальные - цифровые
    MOVWF ADCON1
    MOVLW b'00000001'; Используем канал AD0
    MOVWF ADCON0
    ;TIMER
    MOVLW b'10000001';16-битный, T=1мкс
    MOVWF T1CON

    BSF INTCON,INT0IE; Нажатие кнопки вызывает прерывания
    BCF INTCON2,INTEDG0; По отрицательному фронту
    BSF RCON,IPEN; Разрешение прерываний
    BSF INTCON,GIEH; Разрешение глобальных прерываний
    ; Порты входных индикаторов
    BSF TRISA,RA0
    BSF TRISB,RB0
    BSF TRISB,RB1
    ; Порты выходных индикаторов
    BCF TRISB,RB2
    BCF TRISB,RB3
    BCF TRISB,RB4
    BCF TRISB,RB5
    CLRF LATB; Сброс всех значений в регистре LATB

    RETURN; Возвращение в основную программу
```

Реализация алгоритма подпрограммы – Обработка массива A:

```
MAIN_FUNC:
    MOVFF NumA,NUM_TO_MUL_0125; Запись значения NumA в переменную NUM_TO_MUL_0125
    CALL FUNC_MUL_TO_0125; Вызов подпрограммы: Умножение на 0.125
    MOVFF NUM_TO_MUL_0125,NumA_RES;

    MOVFF NumA_prev,NUM_TO_MUL_025; Запись значения NumA_prev в переменную NUM_TO_MUL_025
```

```
        CALL FUNC_MUL_TO_025; Вызов подпрограммы: Умножение на 0.25
        MOVFF NUM_TO_MUL_025,NumA_prev_RES; Запись значения NUM_TO_MUL_025 в переменную NumA_prev_RES

        MOVFF NumB_prev,NUM_TO_MUL_0125; Запись значения NumB_prev в переменную NUM_TO_MUL_0125
        CALL FUNC_MUL_TO_0125; Вызов подпрограммы: Умножение на 0.125
        MOVFF NUM_TO_MUL_0125,NumB_prev_RES; Запись значения NUM_TO_MUL_0125 в переменную NumB_prev_RES

        MOVFF NumB_prev2,NUM_TO_MUL_075; Запись значения NumB_prev2 в переменную NUM_TO_MUL_075
        CALL FUNC_MUL_TO_075; Вызов подпрограммы: Умножение на 0.75
        MOVFF NUM_TO_MUL_075,NumB_prev2_RES; Запись значения NUM_TO_MUL_075 в переменную NumB_prev2_RES

        MOVF NumA_RES,W; Запись значения NumA_RES в WREG
        ADDWF NumB,F; Операция сложения WREG и NumB. Результат записывается в NumB

        NEGF NumA_prev_RES; Перевод NumA_prev_RES в дополнительный код
        MOVF NumA_prev_RES,W; Запись значения NumA_prev_RES в WREG
        ADDWF NumB,F; Операция сложения WREG и NumB. Результат записывается в NumB

        MOVF NumB_prev_RES,W; Запись значения NumB_prev_RES в WREG
        ADDWF NumB,F; Операция сложения WREG и NumB. Результат записывается в NumB

        NEGF NumB_prev2_RES; Перевод NumA_prev_RES в дополнительный код
        MOVF NumB_prev2_RES,W; Запись значения NumB_prev2_RES в WREG
        ADDWF NumB,F; Операция сложения WREG и NumB. Результат записывается в NumB

        RETURN; Возвращение в основную программу
```

Реализация алгоритма подпрограммы – Умножение на 0.125:

```
FUNC_MUL_TO_0125:
        BTFSS NUM_TO_MUL_0125,7; Проверка старшего бита в переменной NUM_TO_MUL_0125
          BRA K1_0125
        BRA K2_0125
K1_0125:RRCF NUM_TO_MUL_0125,F; Сдвиг переменной NUM_TO_MUL_0125 на 1 разряд
        BCF NUM_TO_MUL_0125,7; Обнуление старшего разряда
        RRCF NUM_TO_MUL_0125,F; Сдвиг переменной NUM_TO_MUL_0125 на 1 разряд
        BCF NUM_TO_MUL_0125,7; Обнуление старшего разряда
        RRCF NUM_TO_MUL_0125,F; Сдвиг переменной NUM_TO_MUL_0125 на 1 разряд
        BCF NUM_TO_MUL_0125,7; Обнуление старшего разряда
        BRA K3_0125
K2_0125:RRCF NUM_TO_MUL_0125,F; Сдвиг переменной NUM_TO_MUL_0125 на 1 разряд
        BSF NUM_TO_MUL_0125,7; Установка старшего разряда
        RRCF NUM_TO_MUL_0125,F; Сдвиг переменной NUM_TO_MUL_0125 на 1 разряд
        BSF NUM_TO_MUL_0125,7; Установка старшего разряда
        RRCF NUM_TO_MUL_0125,F; Сдвиг переменной NUM_TO_MUL_0125 на 1 разряд
        BSF NUM_TO_MUL_0125,7; Установка старшего разряда
        BRA K3_0125
K3_0125:RETURN; Возвращение в основную программу
```

Реализация алгоритма подпрограммы – Умножение на 0.25:

```
FUNC_MUL_TO_025:
```

```
        BTFSS NUM_TO_MUL_025,7; Проверка старшего бита в переменной NUM_TO_MUL_025
          BRA K1_025
        BRA K2_025
K1_025:     RRCF NUM_TO_MUL_025,F; Сдвиг переменной NUM_TO_MUL_025 на 1 разряд
        BCF NUM_TO_MUL_025,7; Обнуление старшего разряда
        RRCF NUM_TO_MUL_025,F; Сдвиг переменной NUM_TO_MUL_025 на 1 разряд
        BCF NUM_TO_MUL_025,7; Обнуление старшего разряда
        BRA K3_025
K2_025:     RRCF NUM_TO_MUL_025,F; Сдвиг переменной NUM_TO_MUL_025 на 1 разряд
        BSF NUM_TO_MUL_025,7; Установка старшего разряда
        RRCF NUM_TO_MUL_025,F; Сдвиг переменной NUM_TO_MUL_025 на 1 разряд
        BSF NUM_TO_MUL_025,7; Установка старшего разряда
        BRA K3_025
K3_025:     RETURN; Возвращение в основную программу
```

Реализация алгоритма подпрограммы – Умножение на 0.75:

```
FUNC_MUL_TO_075:
        MOVFF NUM_TO_MUL_075,NUM_TO_MUL_025; Запись значения NUM_TO_MUL_075 в переменную NUM_TO_MUL_025
        CALL FUNC_MUL_TO_025; Вызов подпрограммы: Умножение на 0.25
        MOVFF NUM_TO_MUL_025,NUM_TO_MUL_075_COPY; Запись значения NUM_TO_MUL_025 в переменную NUM_TO_MUL_075_COPY
        NEGF NUM_TO_MUL_075_COPY; Перевод NUM_TO_MUL_075_COPY в дополнительный код
        MOVF NUM_TO_MUL_075_COPY,W; Запись значения NUM_TO_MUL_075_COPY в WREG
        ADDWF NUM_TO_MUL_075,F; Операция сложения WREG и NUM_TO_MUL_075. Результат записывается в NUM_TO_MUL_075
        RETURN; Возвращение в основную программу
```

Реализация алгоритма прерываний:

```
HighInt:
;       *** high priority interrupt code goes here ***
                BCF INTCON,INT0IF; Сброс флага прерывания
                BSF LATB,RB2; Включение светодиода на RB2
                MOVLW .256; Запись 256 в WREG
                MOVWF Num_HighInt; Запись из WREG в переменную Num_HighInt
                LFSR FSR0,0x100; Задаем базовый адрес для массива A - 0x100

Mint1:                  MOVLW 0xF0; Запись 0xF0 в WREG
                MOVWF TMR1H; Запись из WREG в переменную TMR1H
                MOVLW 0xC8; Запись 0xC8 в WREG
                MOVWF TMR1L; Запись из WREG в переменную TMR1L
                BCF PIR1,TMR1IF; Сброс флага таймера
                BSF T1CON,TMR1ON; Включение таймера
Mint2:                  BTFSS PIR1,TMR1IF; Задержка ~3.9 мс
                  BRA Mint2
                BSF ADCON0,1; Запуск АЦП
Mint3:                  BTFSS ADCON0,1; Проверка на завершение оцифровки
                  BRA Mint3
                MOVFF ADRESH,TMP_V; Запись значений в массив A
                MOVLW 0x81; Запись 0x81 в WREG
                ADDWF TMP_V,F; Операция сложения WREG и TMP_V. Результат записывается в TMP_V
                MOVFF TMP_V,INDF0; Перенос значения из TMP_V в память МК
                INCF FSR0L,F; Переход на следующий адрес массива A
```

```
DECFSZ Num_HighInt,F; Уменьшение на 1 количество операций, и проверка на нулевое значение
  BRA Mint1
BSF LATB,RB3; Включение светодиода на RB3
RETFIE FAST; Возращение из прерывания
```

Список пользовательских констант:

NUM_TO_MUL_05
NUM_TO_MUL_025
NUM_TO_MUL_0125
NUM_TO_MUL_075
NUM_TO_MUL_075_COPY
Num_HighInt
Num
NumA
NumA_prev
NumB
NumB_prev
NumB_prev2
NumA_RES
NumA_prev_RES
NumB_prev_RES
NumB_prev2_RES
TMP_V

Список подпрограмм:

CLRF_ALL
CONF
MAIN_FUNC
FUNC_MUL_TO_0125
FUNC_MUL_TO_025
FUNC_MUL_TO_075

Список источников прерываний:

RB0/INT0 – 21 вывод микросхемы.

Список служебных регистров:

WREG
BSR
STATUS
INDF0
FSR0
FSR0L
INDF1
FSR1
FSR1L
LATB
PORTB
TRISB
TRISA
OSCCON
OSCTUNE
ADCON2
ADCON1
ADCON0
T1CON
INTCON
INTCON2
RCON
TMR1H
TMR1L
PIR1
ADRESH

2. МОДЕЛИРОВАНИЕ И ОТЛАДКА ПРОЕКТА В MPLAB X.

2.1. Ввод подготовленной программы и исходных данных в память микроконтроллера для отладки в MPLAB.

В пакете MPLAB X v4.00 создаем новый проект с названием, к примеру, Main.X. В этом проекте создаем файл формата .ASM с названием KP. Вводим подготовленную программу и проверяем на наличие ошибок. Ошибок не обнаружено, значит собираем проект и запускаем симуляцию. Для получения импульсной характеристики создадим в памяти МК на 100 адресе значение равное 100_{10} или 64_{16}. Для получения переходной характеристики создадим файл тестового сигнала с постоянным значением равным 100_{10} или 64_{16}.

2.2. Отладка в симуляторе MPLAB SIM

Для отображения результата в наглядном виде будем использовать плагин **DMCI Window.** Это позволит нам получать значения в графическом виде, что существенно облегчит анализ цифровой обработки.

Симуляция импульсной характеристики.

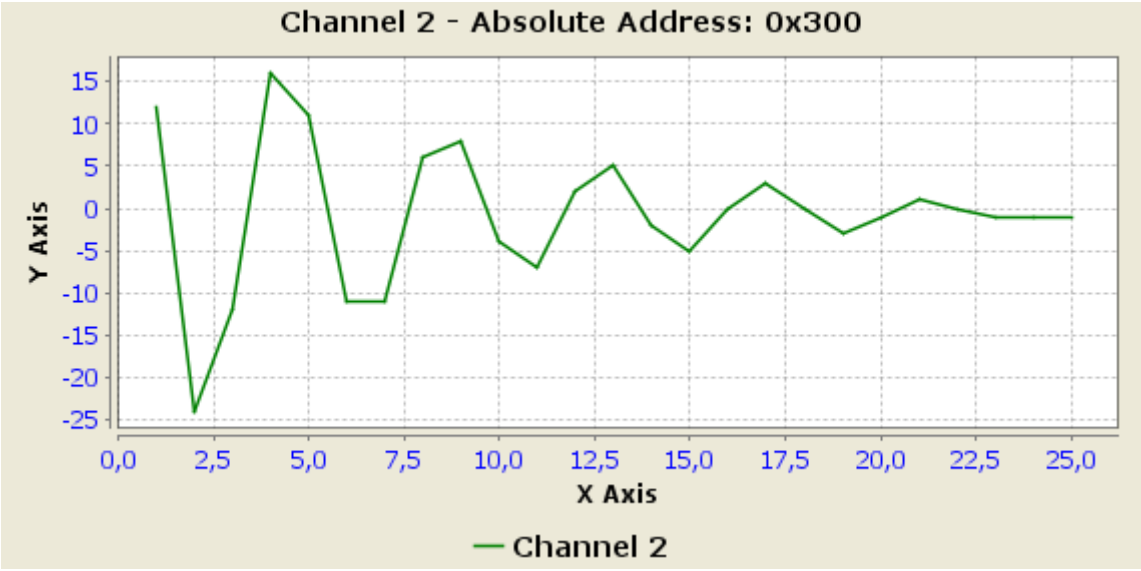


Рисунок 2.1 – Результат симуляции импульсной характеристики.

Общий вид и значения полностью совпадают с рассчитанными в пункте **1.4**.

Симуляция передаточной характеристики.

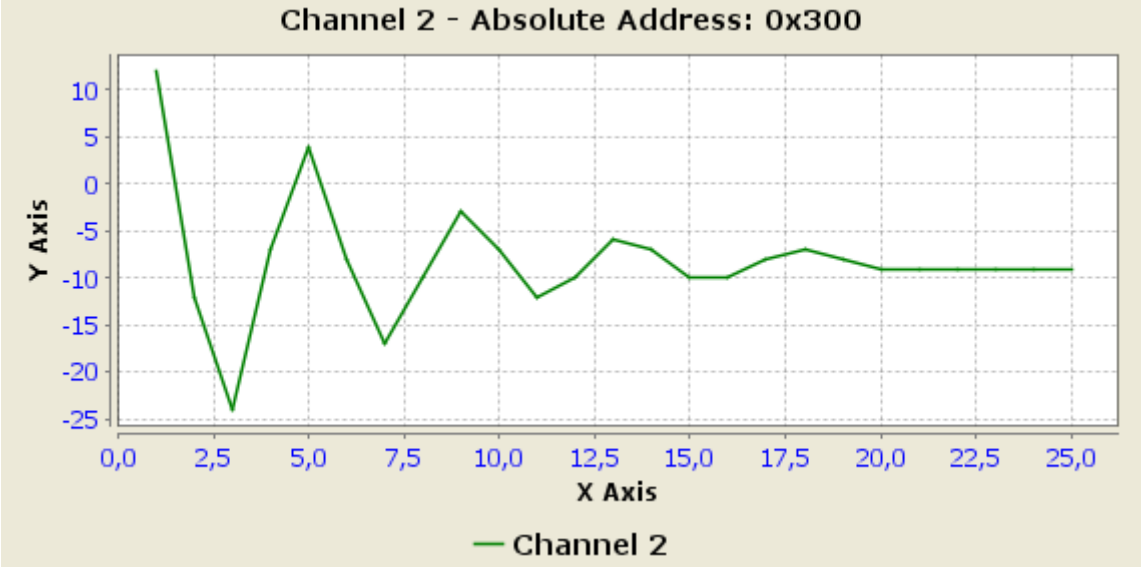


Рисунок 2.2 – Результат симуляции переходной характеристики.

Общий вид и значения так же полностью совпадают с рассчитанными в пункте **1.4**.

Симуляция сигналов разной частоты.

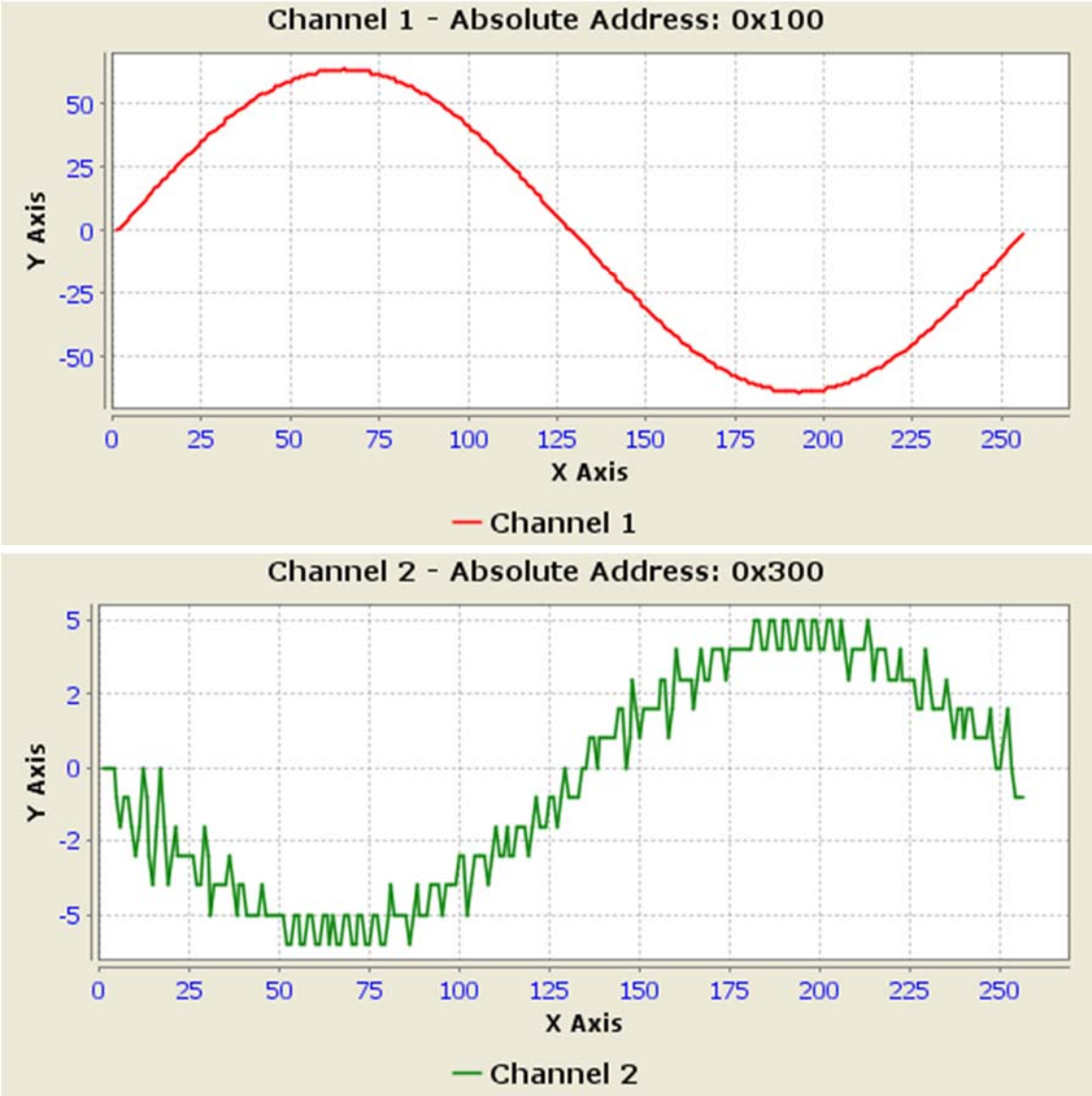


Рисунок 2.3 – Результат симуляции сигнала с частотой 1 Гц.

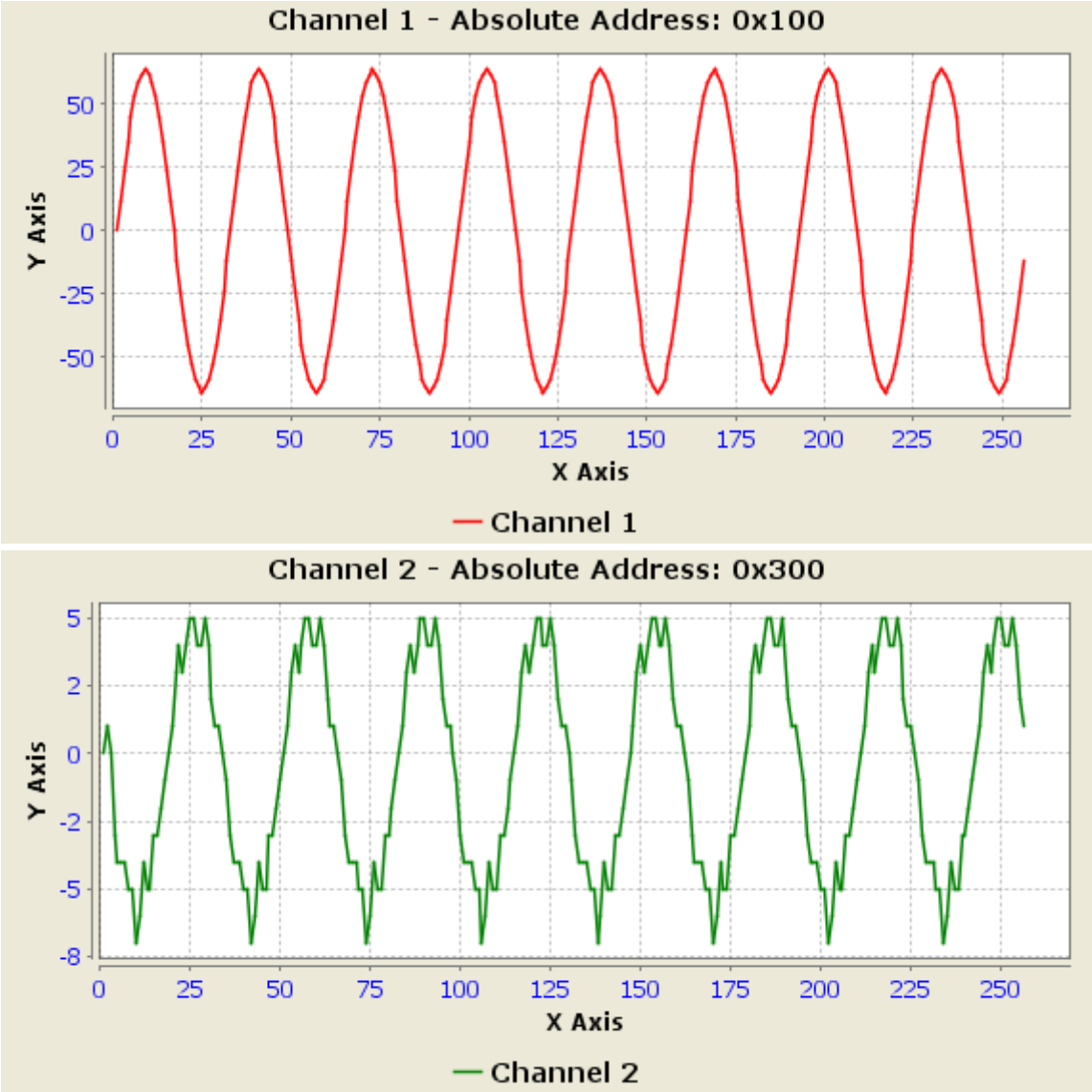


Рисунок 2.4 – Результат симуляции сигнала с частотой 8 Гц.

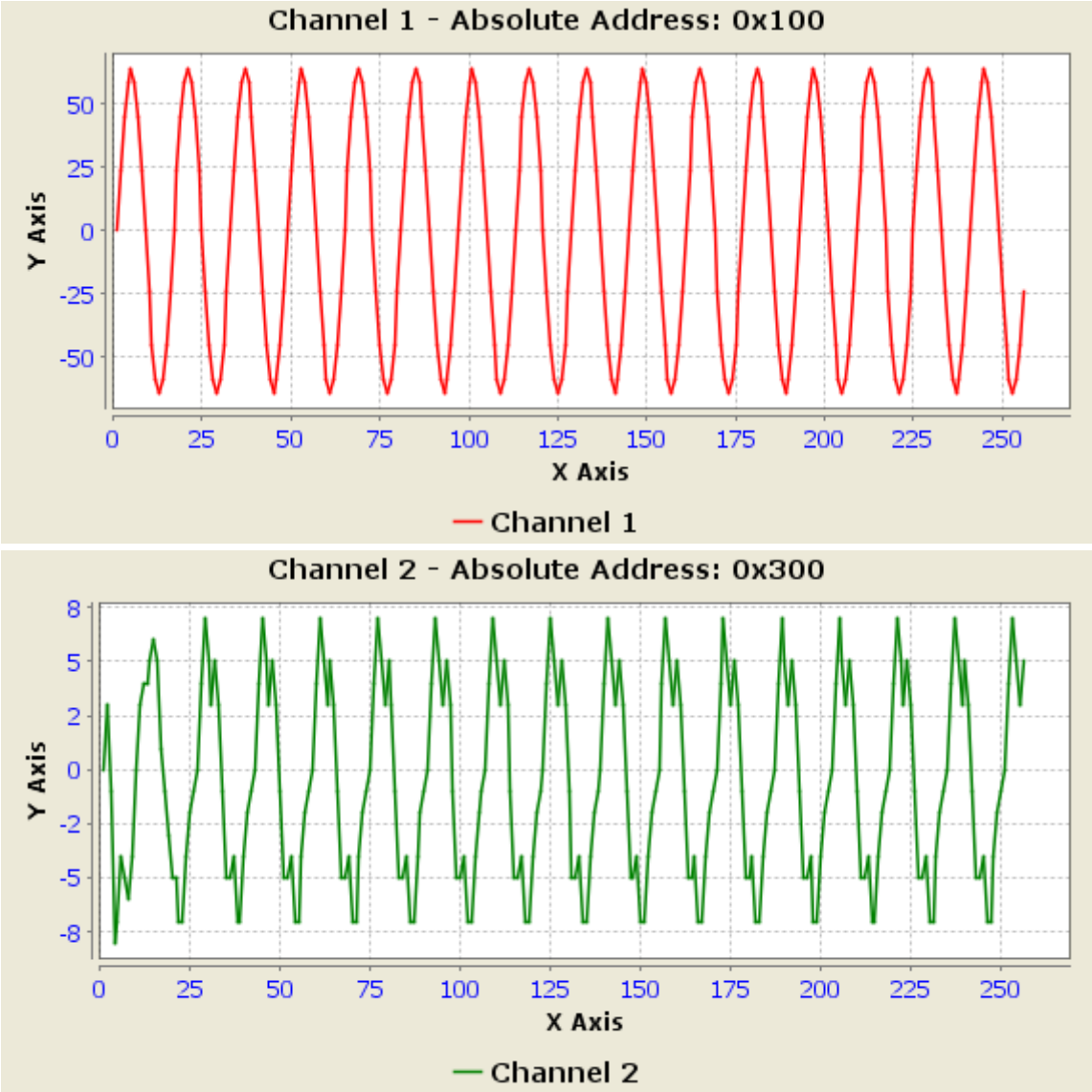


Рисунок 2.5 – Результат симуляции сигнала с частотой 16 Гц.

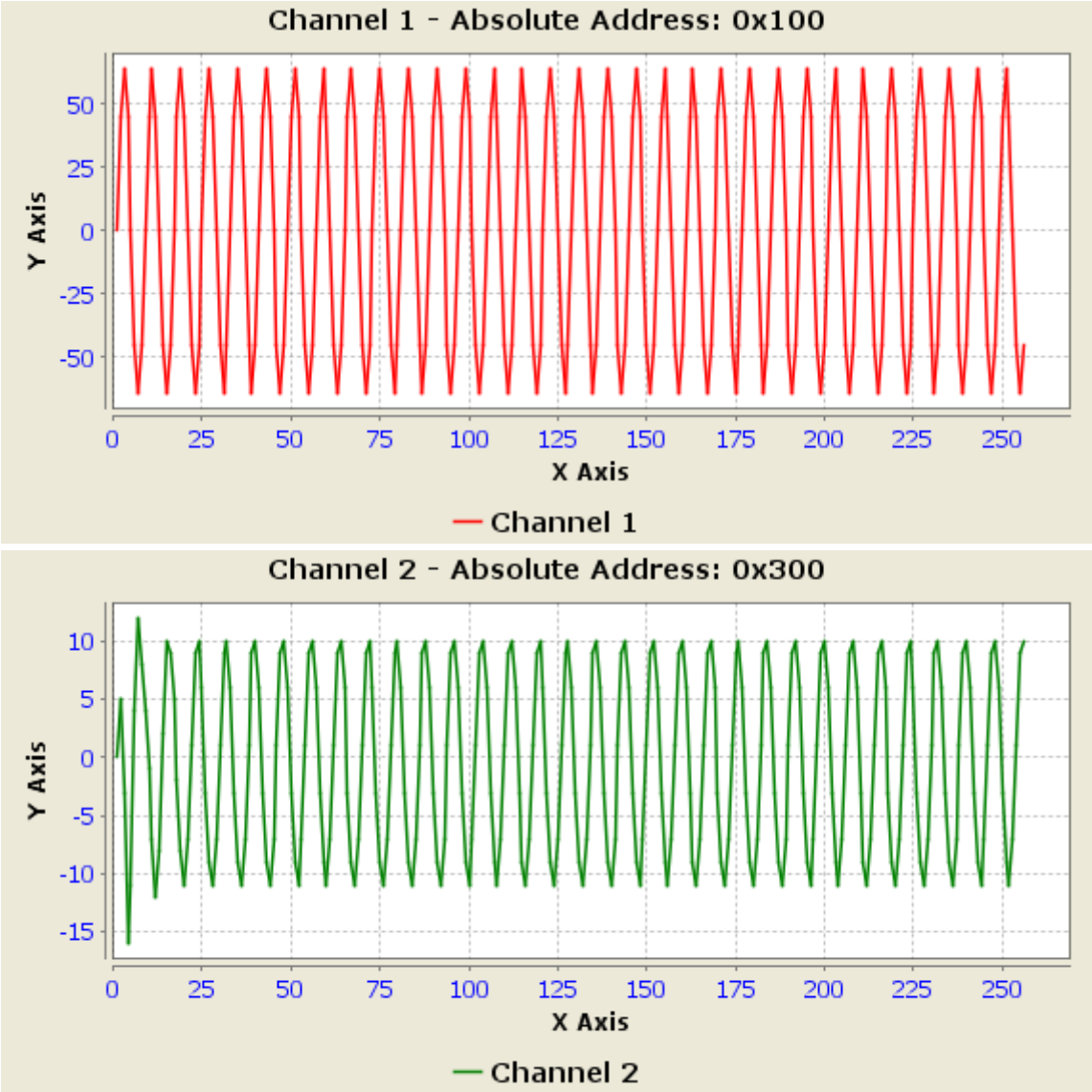


Рисунок 2.6 – Результат симуляции сигнала с частотой 32 Гц.

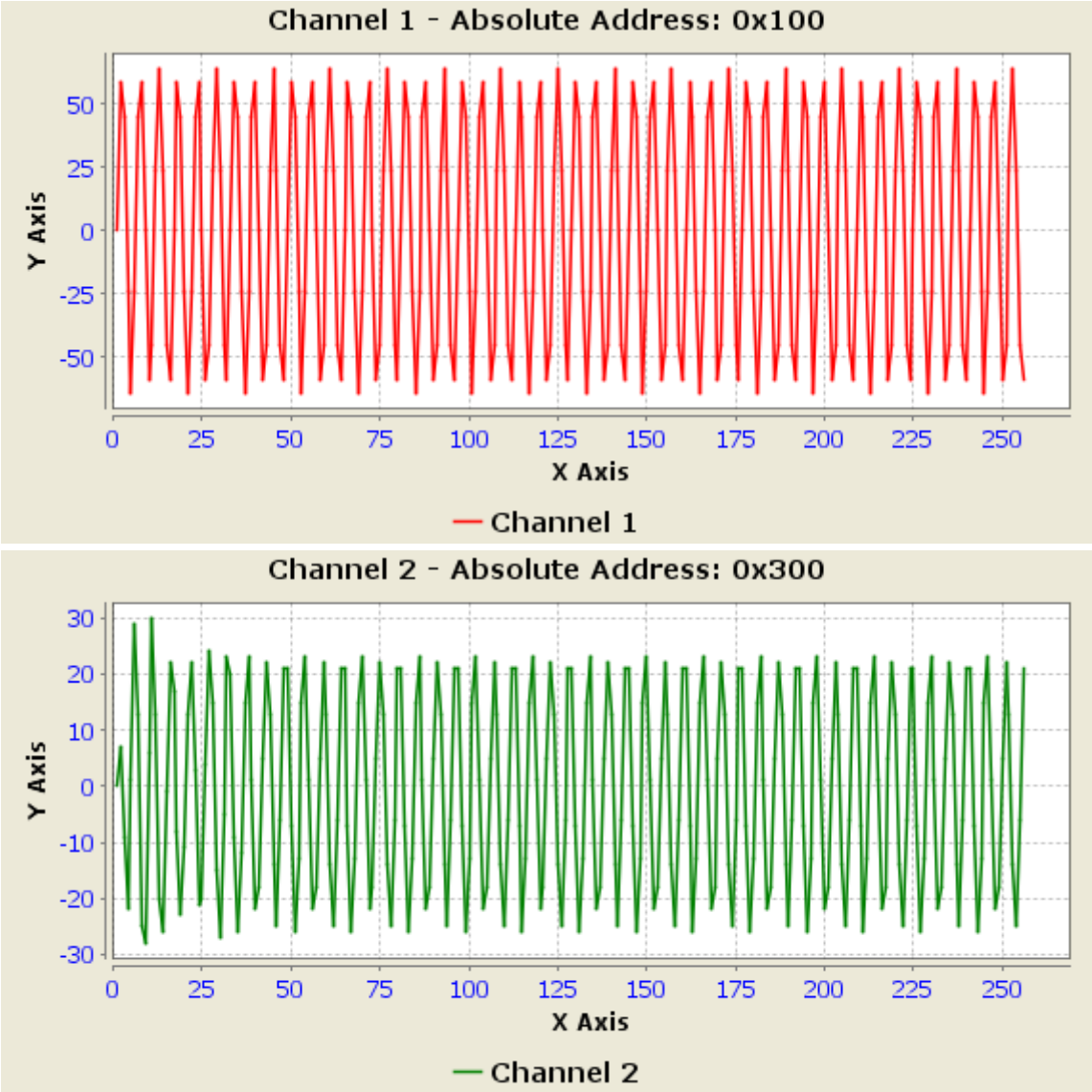


Рисунок 2.7 – Результат симуляции сигнала с частотой 48 Гц.

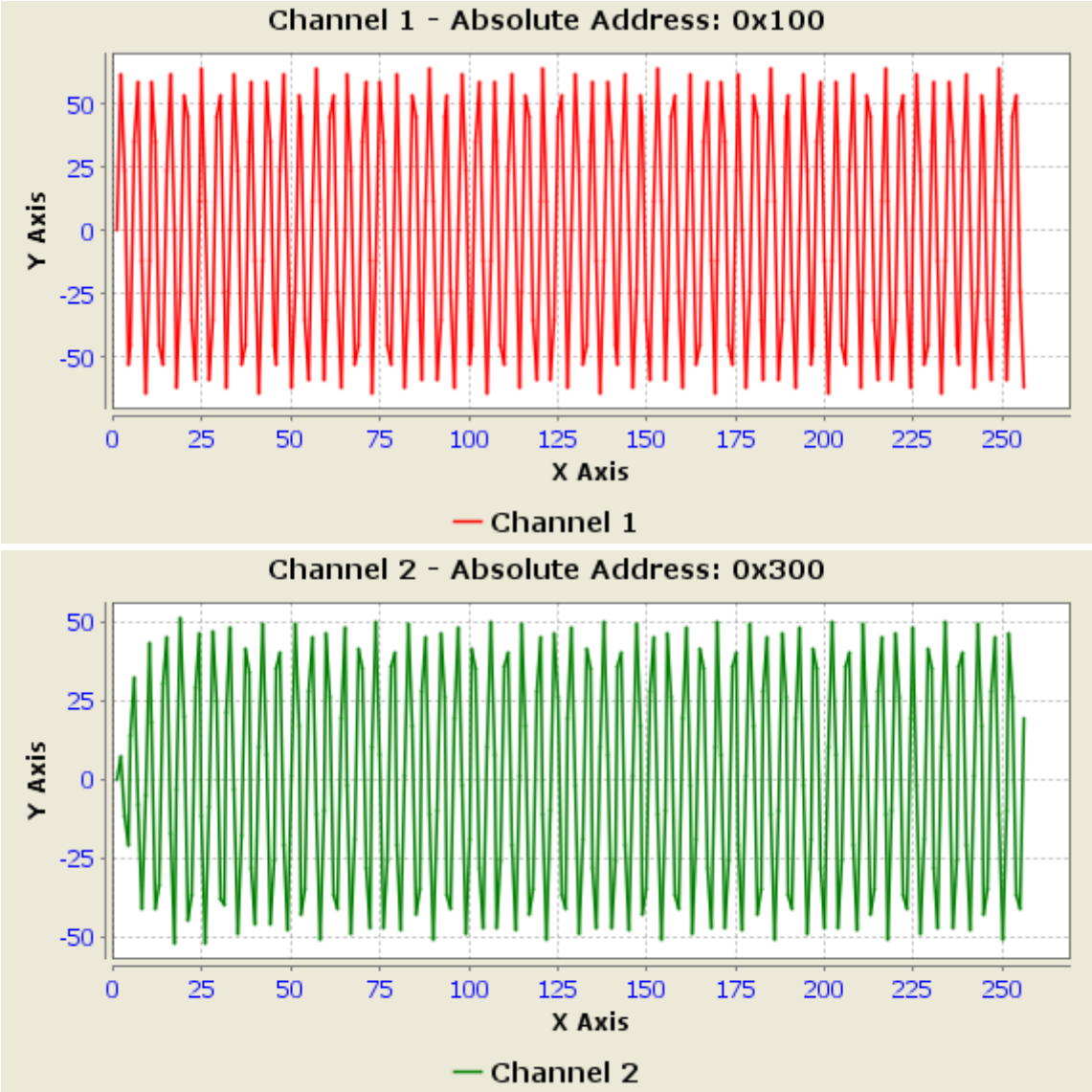


Рисунок 2.8 – Результат симуляции сигнала с частотой 56 Гц.

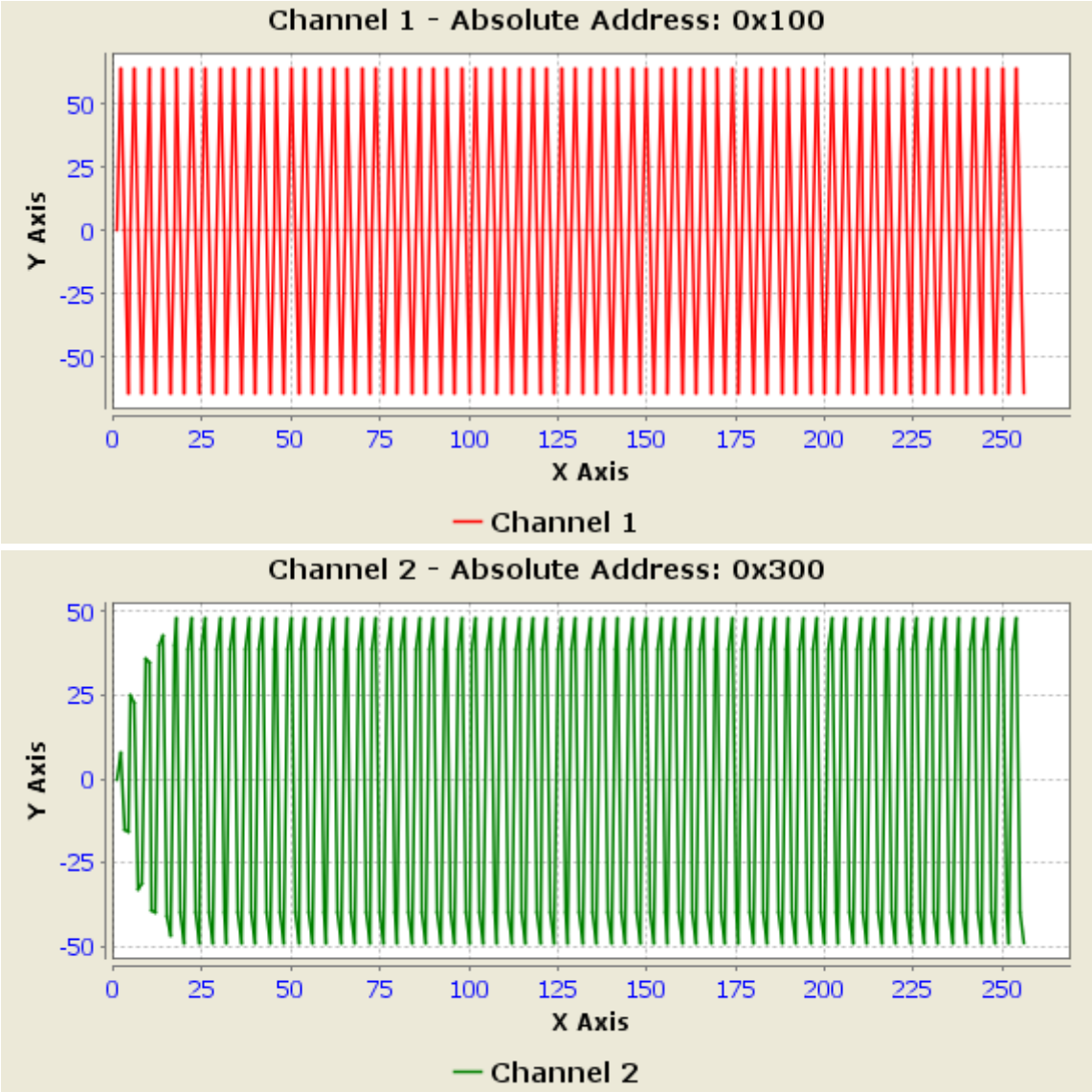


Рисунок 2.8– Результат симуляции сигнала с частотой 64 Гц.

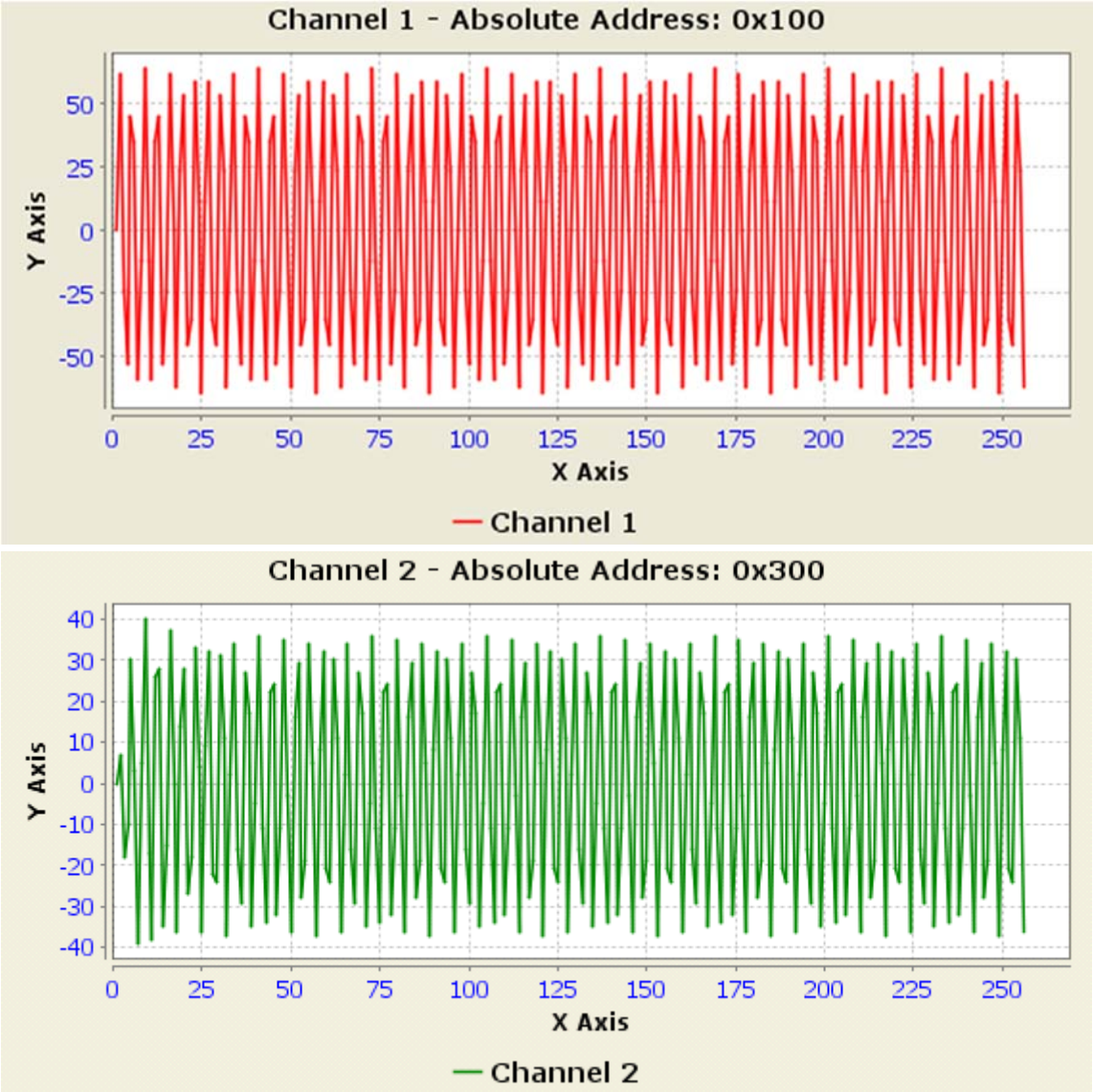


Рисунок 2.9 – Результат симуляции сигнала с частотой 72Гц.

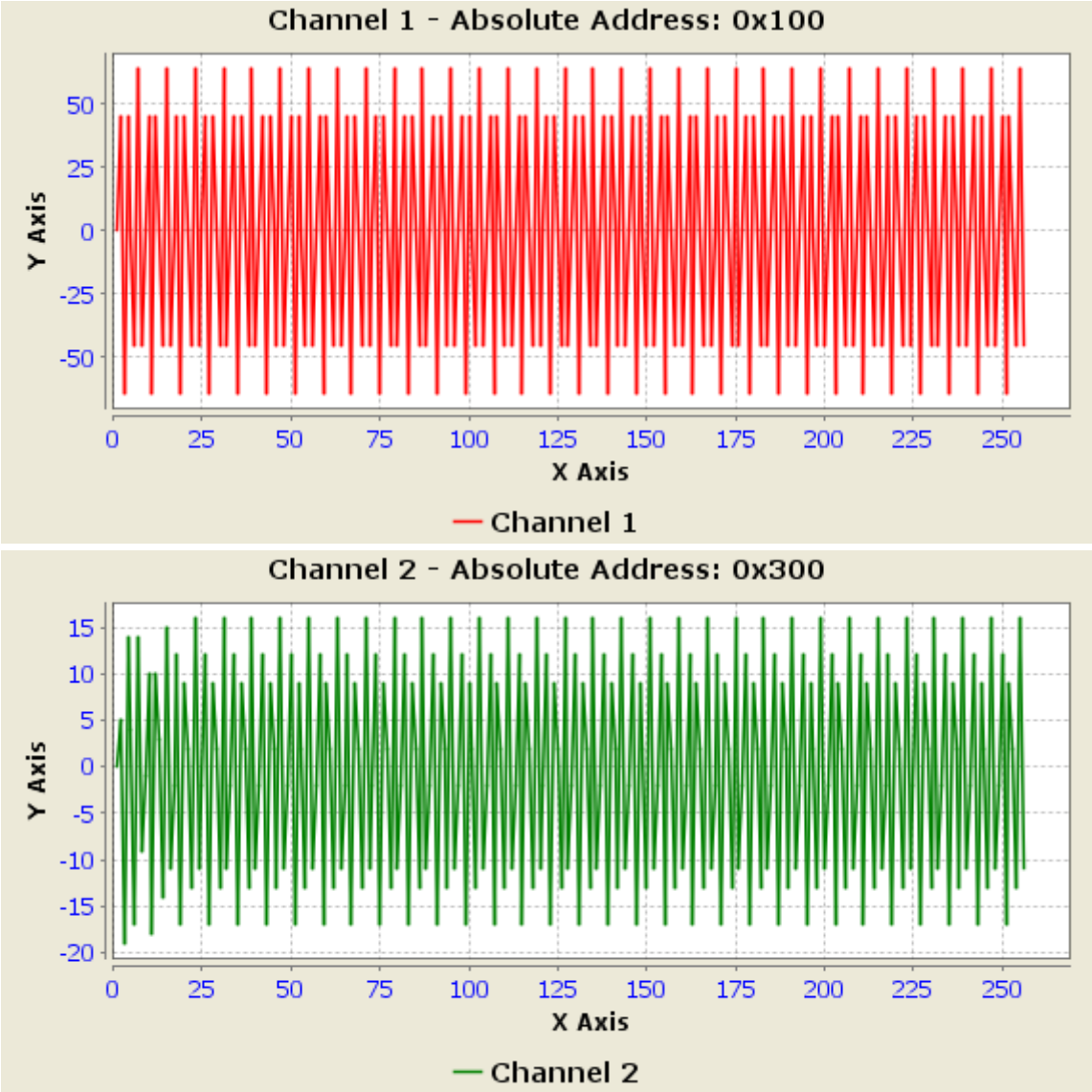


Рисунок 2.10 – Результат симуляции сигнала с частотой 96 Гц.

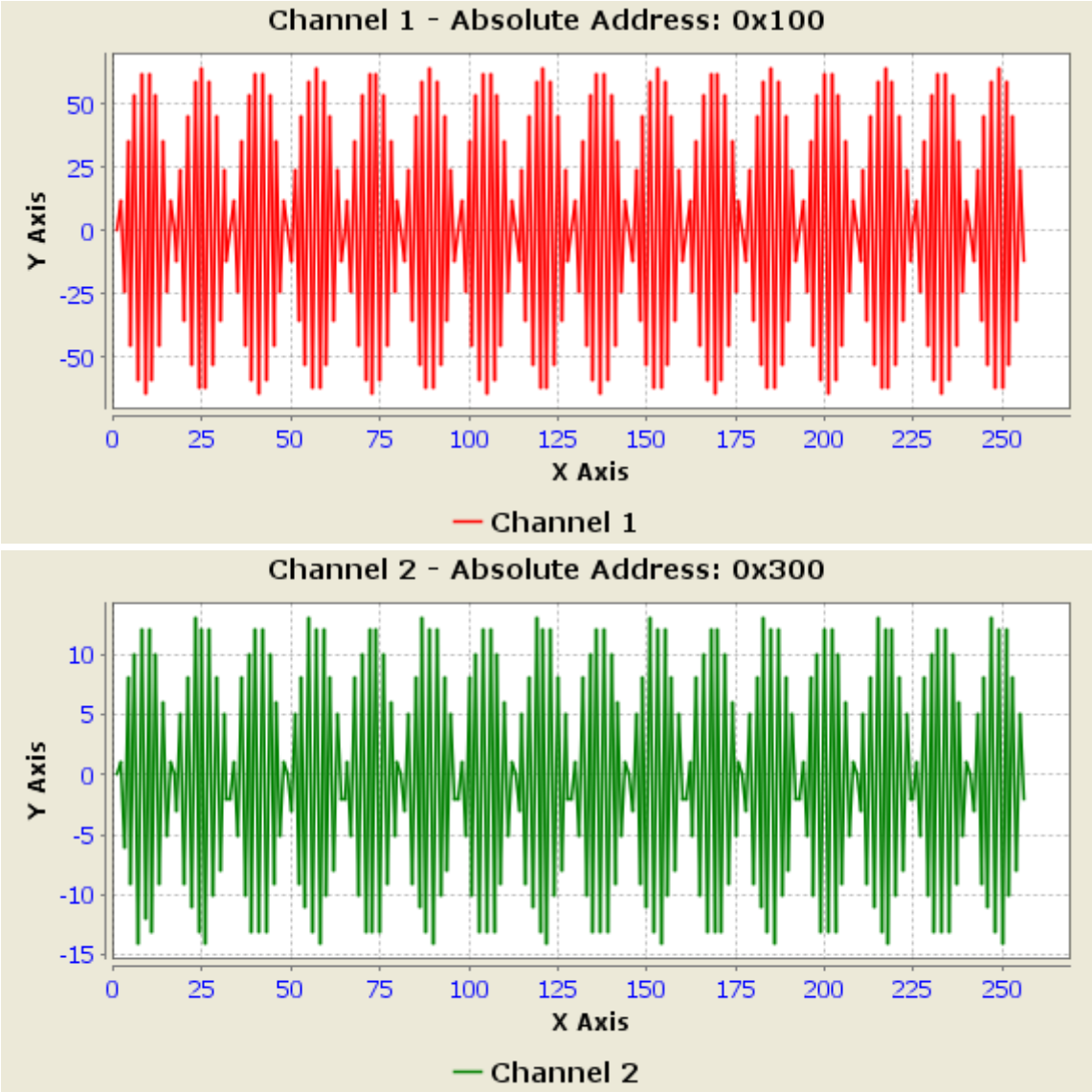


Рисунок 2.11 – Результат симуляции сигнала с частотой 120 Гц.

Симуляция суммы сигналов с разными частотам

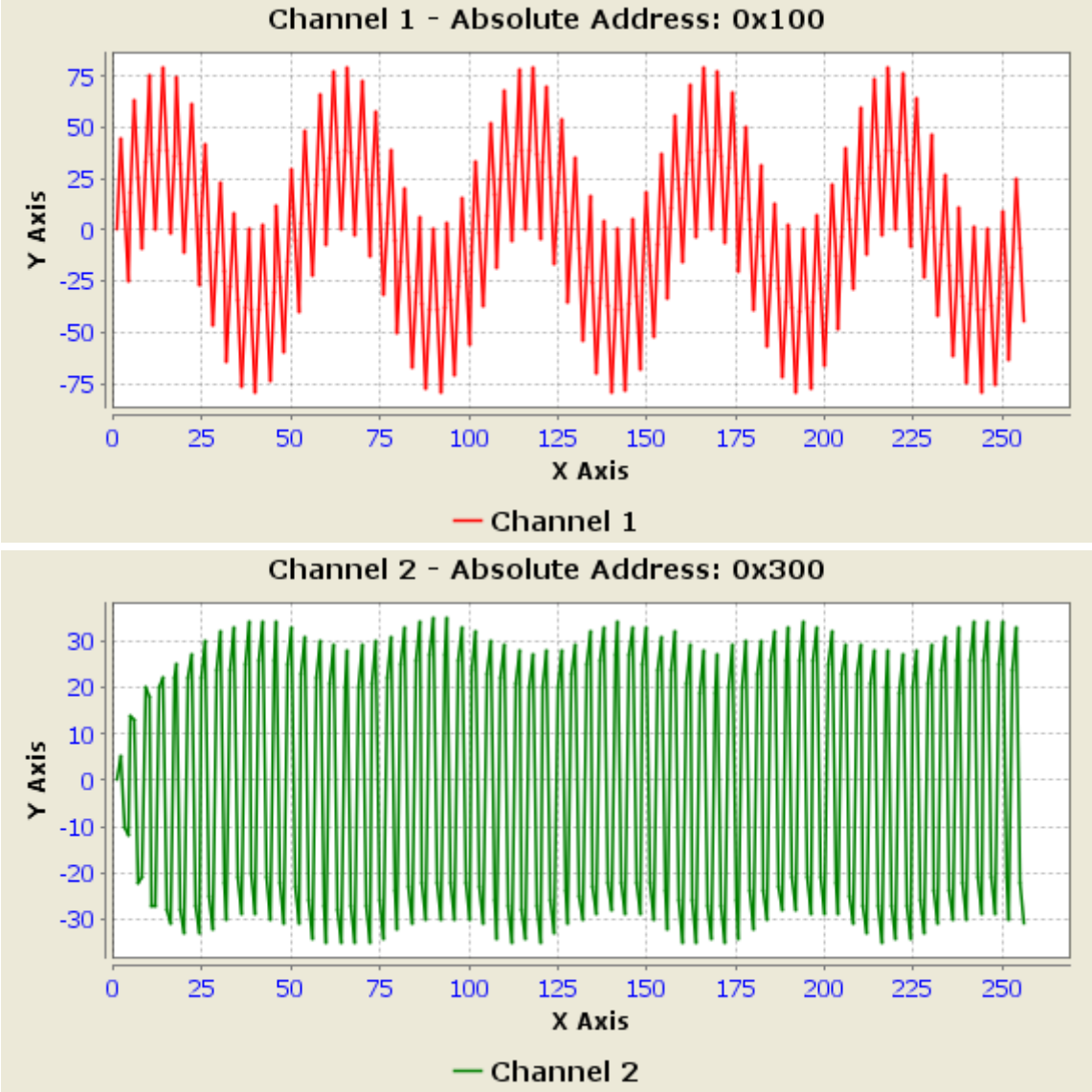


Рисунок 2.12 – Результат симуляции сигналов с частотами 5 Гц и 64 Гц.

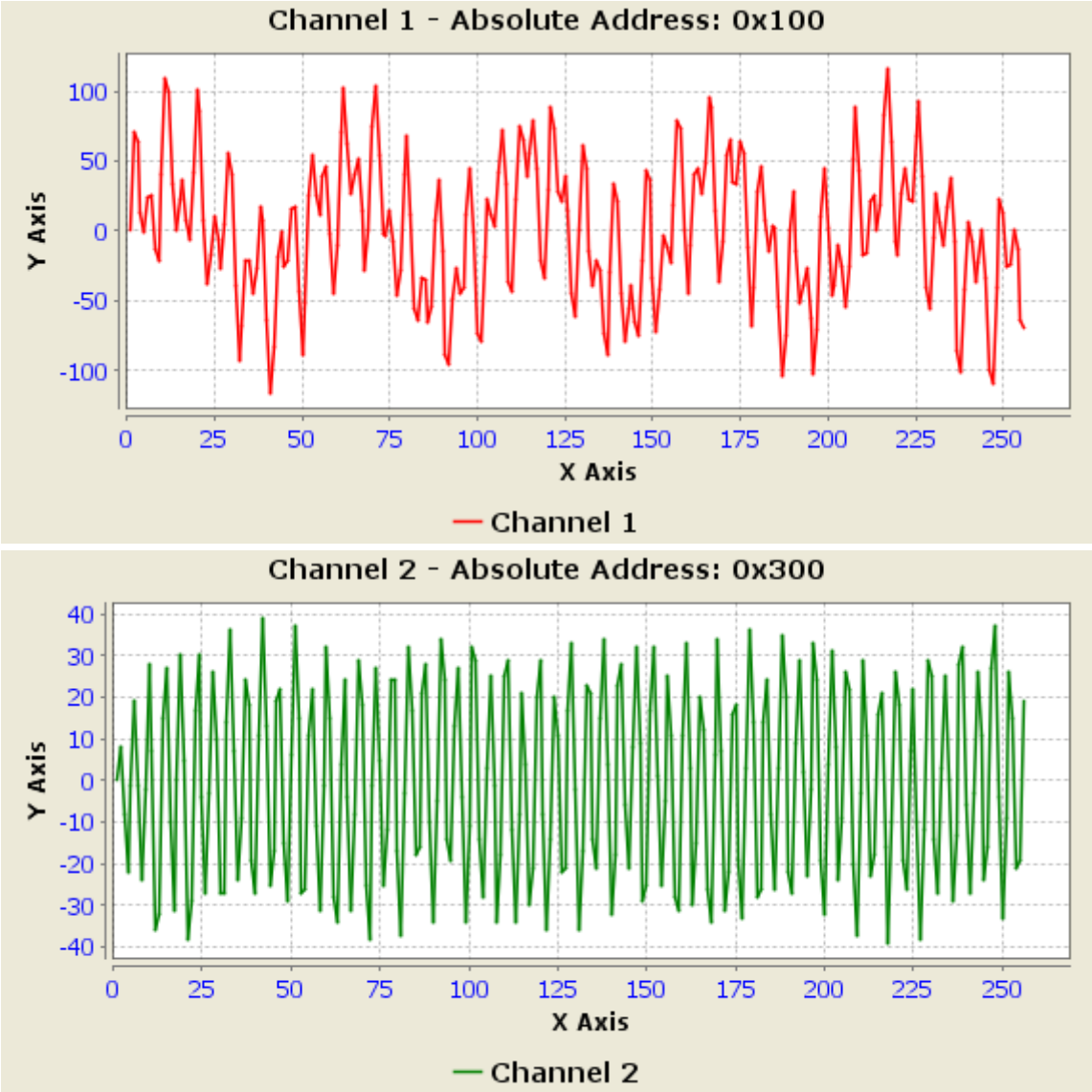


Рисунок 2.13 – Результат симуляции сигналов с частотами 5 Гц, 30 Гц и 56 Гц.

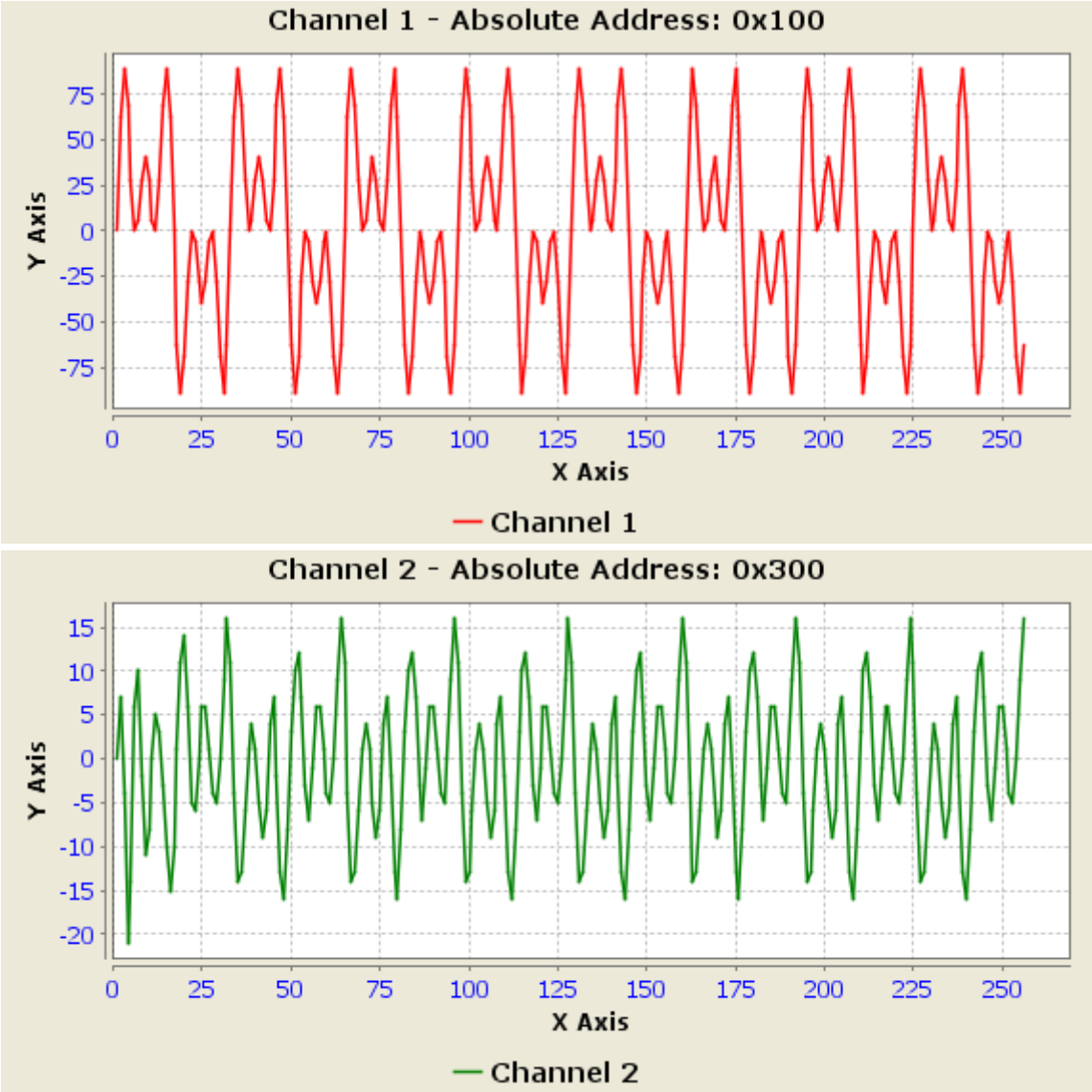


Рисунок 2.14 – Результат симуляции сигналов с частотами 8 Гц, 24 Гц и 40 Гц.

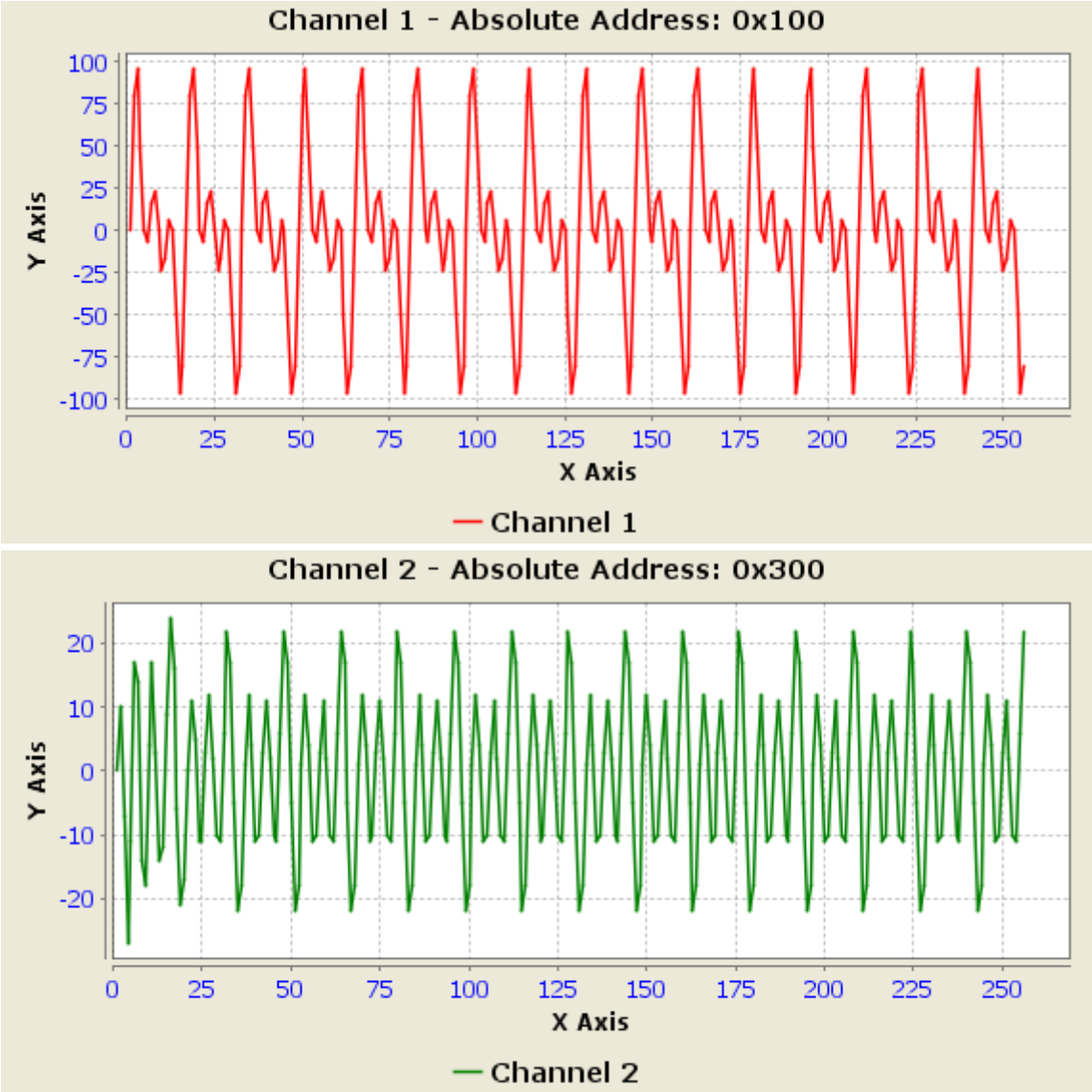


Рисунок 2.15 – Результат симуляции сигналов с частотами 16 Гц, 32 Гц и 48 Гц.

2.3. Проверка правильности полученных результатов.

По рисункам 2.3 – 2.11, на которых присутствует только один сигнал, можно определить отношение амплитуд. Это отношение и будет значением передаточной функции на определённой частоте. Там, где отношение будет максимальным, будет резонансная частота фильтра.

Таблица 2.1. Отношение амплитуд для различных частот.

f, Гц	1	8	16	32	48	56	64	72	90	120
$\frac{U_{вых}}{U_{вх}}$	0,078	0,078	0,109	0,156	0,375	0,703	0,781	0,594	0,25	0,188

Теперь нормируем полученные значение и нанесем их на график АЧХ фильтра

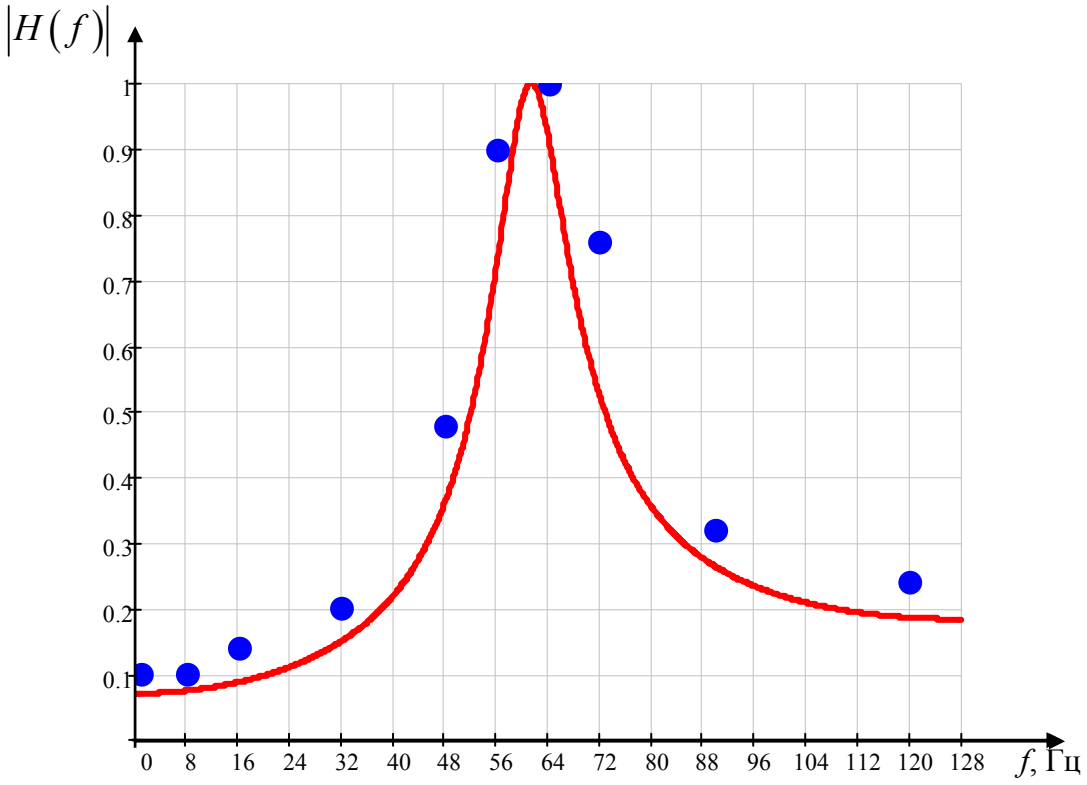


Рисунок 2.16 – Нормированная амплитудно-частотная характеристика ЦФ.

Как видно по рисунку 2.16 результаты симуляции совпадают с теоритическим анализом с небольшой погрешностью. Резонансная частота в обоих случаях составляет 64 Гц.

3. ПРОВЕРКА ПРАВИЛЬНОСТИ ФУНКЦИОНИРОВАНИЯ МПС ПРИ ПОМОЩИ МОДУЛЯ PICKIT.

В пакете MPLAB X переключим Debub Tool с симуляции на модуль PICkit 3.0. Теперь мы можем подключить МПС к данному модулю. После подключения устанавливаем питание +5В. Теперь на МПС подаем постоянное напряжение смещения с помощью делителя напряжения для того, что бы подавать на АПЦ МК только положительные значения напряжения. Подключаем на вход МПС генератор аналогово сигнала. Результат оцифровки АЦП записываем в массив А. После оцифровки обрабатываем сигнала и записываем его в массив В. Результаты обоих массивов выводим с помощью плагина DMCI.

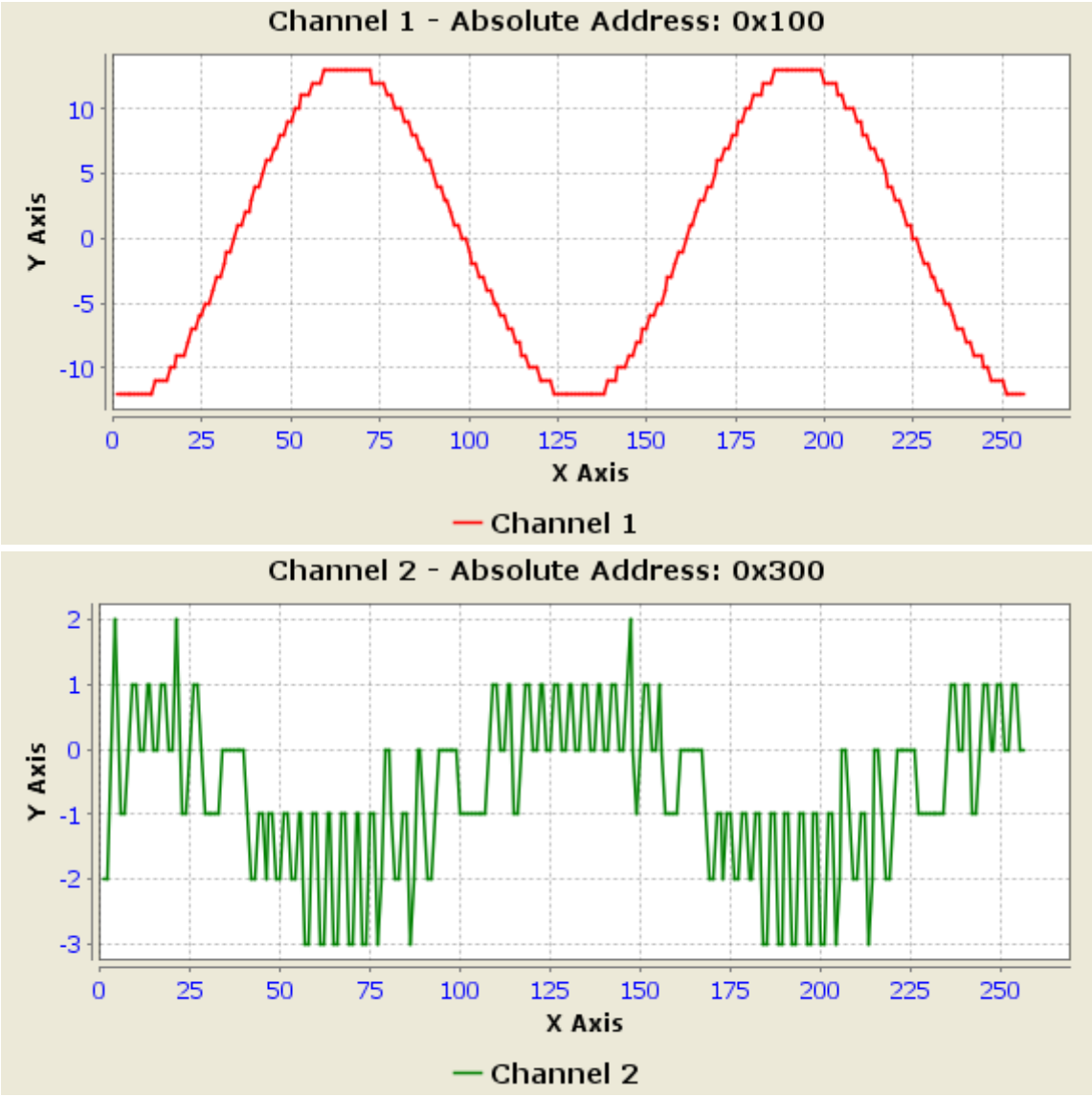


Рисунок 3.1 – Результат оцифровки и обработки сигнала. Частота сигнала 2 Гц.

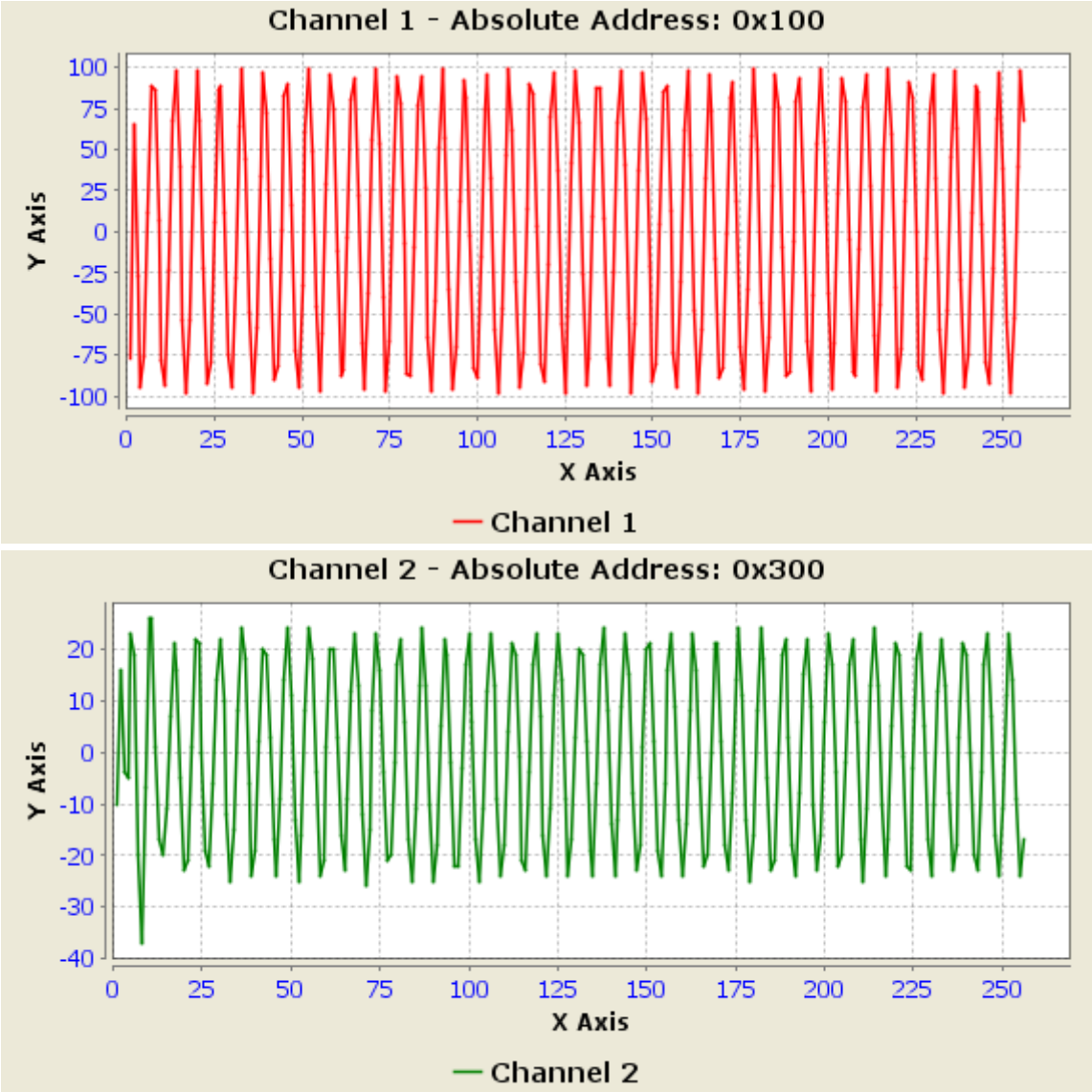


Рисунок 3.2 – Результат оцифровки и обработки сигнала. Частота сигнала 40 Гц.

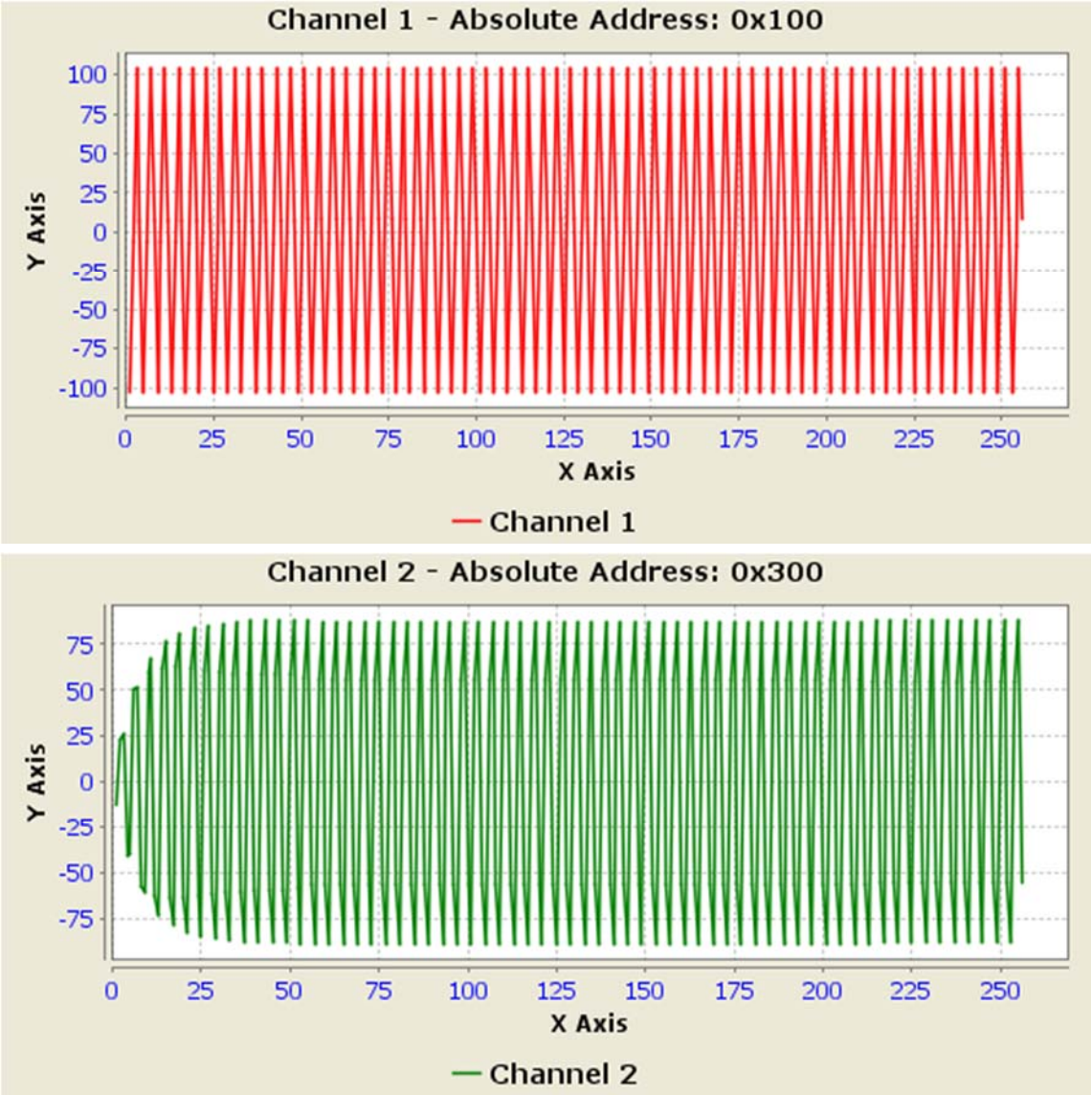


Рисунок 3.3 – Результат оцифровки и обработки сигнала. Частота сигнала 2 Гц.

ЗАКЛЮЧЕНИЕ

Амплитуда входного сигнала во всех 3 случая (Рисунки 3.1 – 3.3) составляет 2 В, но так как на входе МПС стоит конденсатор и резисторы, они образуют некий фильтр, который меняет амплитуду оцифрованного сигнала. Эти данные полностью совпадают и с теоритическими расчетами, и с симуляцией. Следует, что данная программа работает исправно, и работа продела верно.

Время работы АЦП составляет 1 секунда, что и требуется по заданию. Время на обработку массива необходимо в разы меньше, оно составляет 29,387 мс. Если изменить время работы АЦП до уровня обработки сигнала, данная программа будет работать с максимальной скоростью.