Степанов А.Б., Суворов К.А.

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ В РАДИОТЕХНИЧЕСКИХ СИСТЕМАХ

РЕАЛИЗАЦИЯ АЛГОРИТМОВ НА ЭЛЕМЕНТНОЙ БАЗЕ СО СВЕРХНИЗКИМ ЭНЕРГОПОТРЕБЛЕНИЕМ

Лабораторный практикум по дисциплине

СПбГУТ

Лабораторная работа №4
Реализация нерекурсивного цифрового фильтра

Цель работы: реализовать нерекурсивный цифровой фильтр на микроконтроллере MSP430G2452.

4.1. Краткая теоретическая справка

Под цифровым фильтром (ЦФ) в широком смысле [3, 4] понимают линейную дискретную систему (ЛДС), преобразующую входной сигнал (воздействие) в выходной (реакцию) согласно заданному алгоритму (разностному уравнению). По типу ЛДС выделяют рекурсивные и нерекурсивные фильтры. Как правило, рекурсивные ЦФ обладают бесконечной импульсной характеристикой и называются БИХ-фильтры, а нерекурсивные – конченой импульсной характеристикой и называются КИХ-фильтры.

В узком смысле под цифровым фильтром понимается частотно-избирательный фильтр: фильтр нижних частот (ФНЧ), фильтр верхних частот (ФВЧ), режекторный или полосовой.

Микроконтроллеры предназначены в основном для реализации систем управления. Однако производительность современных микроконтроллеров позволяет применять их и при реализации цифровых фильтров.

Рассмотрим основные этапы проектирования нерекурсивного цифрового фильтра с реализацией на микроконтроллере MSP430G2xxx:

1. Задание требований к цифровому фильтру. Выбор: частоты дискретизации, граничных частот полос пропускания и задерживания, максимально допустимого отклонения в полосе пропускания и в полосе задерживания.

2. По типу выбран фильтр, имеющий конечную импульсную характеристику и, как следствие, обладающий следующими достоинствами:

- устойчивость (по определению);
- легкость получения линейной ФЧХ (достаточно, чтобы он обладал симметричной или антисимметричной импульсной характеристикой);
- отсутствие необходимости разбиения структуры фильтра на звенья.

На данном этапе определяется метод синтеза КИХ-фильтра: метод окон или метод наилучшей равномерной (чебышевской) аппроксимации.

3. Моделирование цифрового фильтра в математическом пакете, например, в MATLAB. В результате моделирования производится оценка графиков: АЧХ, ФЧХ, импульсной характеристики, а также рассчитываются коэффициенты фильтра.

4. Выбор типа передаточной функции и структуры фильтра.

5. На основе полученных коэффициентов может быть записано разностное уравнение, являющееся алгоритмом вычисления реакции. Оно используется при написании программного кода для реализации фильтра в Code Composer Studio.

6. Для проверки правильности работы реализованного фильтра на его вход может быть подан цифровой единичный импульс $u_0(n)$. При нулевых начальных условиях на его выходе формируется импульсная характеристика, которая должна совпадать со значениями импульсной характеристики, полученной на этапе моделирования. Для формирования цифрового единичного импульса, а также визуализации графиков на выходе фильтра может быть использован MATLAB.

4.2. Задание на лабораторную работу

Выполнение лабораторной работы осуществляется в следующем порядке:

1. Подключите отладочную плату MSP-EXP430G2 к компьютеру.
2. Запустите Code Composer Studio: «Start → All Programs → Texas Instruments → Code Composer Studio».
3. Создайте новый проект: «File → New → CCS Project».
4. В поле «Project name» введите имя проекта, нажмите «Next», установите тип проекта: «Project Type → MSP430». Нажмите «Next» два раза для перехода к странице настроек проекта – «Project Settings». В поле «Device Variant» выберите необходимое устройство из списка – семейство и модель микроконтроллера, например, MSP430G2452. Модель микроконтроллера можно уточнить, обратившись к маркировке микросхемы. Далее нажмите «Finish».
5. Создайте новый исходный файл: «File → New → Source File». Введите имя файла на английском языке и добавьте расширение *.c.
6. В окне редактирования исходного кода введите текст программы:

```
#include "msp430g2452.h"
// Определение системы ввода-вывода
#define UART_TXD   0x02        // TXD (передатчик) на порт P1.1
(Timer0_A.OUT0)
#define UART_RXD   0x04          // RXD (приемник) на порт P1.2
(Timer0_A.CCI1A)
// Настройка программного UART на 9600 бод, SMCLK = 8MHz
#define UART_TBIT_DIV_2    (8000000 / (9600 * 2))
//Длительность половины бита
#define UART_TBIT          (8000000 / 9600)
                                    //Длительность одного бита
// Определение глобальных переменных для работы полнодуплексно-
го UART
unsigned int txData;     // Перем. для хранения передав. данных
unsigned char rxBuffer;  // Перем. для хранения принятых данных
```

```
float acc = 0;
        // Аккумулятор (хранение данных при работе КИХ-фильтра)
const float filter[11] = {
                    // Массив для хранения коэффициентов фильтра
-0.056603025243006803,
-0.066187865599614315,
 0.02459150075869989,
 0.12733717738513645,
 0.27539738610024989,
 0.31033143717309675,
 0.27539738610024989,
 0.12733717738513645,
 0.02459150075869989,
-0.066187865599614315,
-0.056603025243006803 };
float sample[11] = {0};
                // Массив для хранения принятых отсчетов сигнала
short i,Rbyte = 0;           // Переменные для работы счетчиков
union fl2str{
             // Переменная для хранения входных отсчетов сигнала
float f;                    //имя переменной типа float (4 байта)
char ch[4];     //массив из четырех элементов типа char(1 байт)
} f2s;
union str2fl{
            // Переменная для хранения выходных отсчетов сигнала
     float f;
     char ch[4];
} s2f;
// Определение функций
void TimerA_UART_init(void);
                      //Функция  инициализации программного UART
void TimerA_UART_tx(unsigned char byte);
                         //Функция передачи одного байта по UART
void main(void){
WDTCTL = WDTPW + WDTHOLD;      // Остановка сторожевого таймера
DCOCTL = 0x00;                              // настройка DCOCLK
BCSCTL1 = CALBC1_8MHZ;
DCOCTL = CALDCO_8MHZ;
P1OUT = 0x00;              // Инициализация системы ввода-вывода
P1SEL = UART_TXD + UART_RXD;
//Настройка портов TXD/RXD на работу с внут. периф. (таймером)
P1DIR = 0xFF & ~UART_RXD;
          // Установка всех контактов порта на вывод, кроме RXD
__enable_interrupt();                     //Разрешение прерываний
TimerA_UART_init();                        // Запуск Timer_A UART
P1OUT = 0x01;
for (;;)
{
                                      // Ожидание входящих данных
__bis_SR_register(LPM0_bits);
P1OUT &= ~(BIT0 + BIT6);
```

```
if (0 < Rbyte < 4){                //если счетчик байт меньше 4
s2f.ch[Rbyte - 1] = rxBuffer;
}
if (Rbyte == 4) {                  //если счетчик байт равен 4
s2f.ch[Rbyte - 1] = rxBuffer;
sample[0] = s2f.f;   //запись отсчета в нулевой элемент массива
acc = 0;                                  //Очистка аккумулятора
for (i = 0; i < 11; i++) {
acc += (sample[i] * filter[i]);      // Умножение с накоплением
}
f2s.f = acc;          // Запись результата в выходную переменную
for (i = 10; i > 0; i--)
sample[i] = sample[i - 1];     // Смещение задержанного сигнала
TimerA_UART_tx(f2s.ch[0]);                              //побайтовая
TimerA_UART_tx(f2s.ch[1]);                                //передача
TimerA_UART_tx(f2s.ch[2]);                              //результата
TimerA_UART_tx(f2s.ch[3]);                    //по интерфейсу UART
f2s.f = s2f.f = Rbyte = 0; //очистка переменных
          }
    }
}
// Функция настройки таймера Timer_A для полнодуплексного UART
void TimerA_UART_init(void)
{
TACCTL0 = OUT;
TACCTL1 = SCS + CM1 + CAP + CCIE;
TACTL = TASSEL_2 + MC_2;
}
// Передача одного байта с использованием Timer_A UART
void TimerA_UART_tx(unsigned char byte)
{
while (TACCTL0 & CCIE);  // Проверка передачи предыдущих байтов
TACCR0 = TAR;                    // Текущее состояние счетчика TA
TACCR0 += UART_TBIT;          // Прибавление времени одного бита
TACCTL0 = OUTMOD0 + CCIE;
txData = byte;                  // Загрузка глобальной переменной
txData |= 0x100;   // Добавить стоповый бит к переменной TXData
txData <<= 1;                            // Добавить стартовый бит
P1OUT ^= BIT6;
}
// Timer_A UART. Обработчик прерывания передачи
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A0_ISR(void)
{
static unsigned char txBitCnt = 10;
TACCR0 += UART_TBIT;                // Добавить смещение к CCRx
if (txBitCnt == 0) {                   // Все ли биты переданы?
TACCTL0 &= ~CCIE;                          // Запрет прерываний
txBitCnt = 10;                     // Перезагрузка счетчика бит
}
```

```
else {
if (txData & 0x01) {
TACCTL0 &= ~OUTMOD2;                                    // Вывод '1'
}
else {
TACCTL0 |= OUTMOD2;                                     // Вывод '0'
}
txData >>= 1;                                  //Смещение на один бит
txBitCnt--;                                     //Декремент счетчика
}
}
// Timer_A UART. Обработчик прерывания приема.
#pragma vector = TIMER0_A1_VECTOR
__interrupt void Timer_A1_ISR(void)
{
static unsigned char rxBitCnt = 8;
static unsigned char rxData = 0;
switch (__even_in_range(TA0IV, TA0IV_TAIFG)) {
case TA0IV_TACCR1:
TACCR1 += UART_TBIT;
if (TACCTL1 & CAP) {
TACCTL1 &= ~CAP;
TACCR1 += UART_TBIT_DIV_2;
}
else {
rxData >>= 1;
if (TACCTL1 & SCCI) {
rxData |= 0x80;
}
rxBitCnt--;
if (rxBitCnt == 0) {                        // Все ли биты приняты?
rxBuffer = rxData;              // Запись в глобальную переменную
Rbyte++;
rxBitCnt = 8;                  // Перезагрузка счетчика бит
TACCTL1 |= CAP;
__bic_SR_register_on_exit(LPM0_bits);
}
}
break;
}
}
```

7. Запустите MATLAB. Создайте новый файл с расширением *.m. Наберите текст программы:

```
script
Fs = 1000;                                      %Частота дискретизации
T = 1/Fs;                                        %Время одного отсчета
L = 50;                                  %Количество отсчетов сигнала
t = (0:L-1)*T;                                          %Временной вектор
x = zeros(1, L);                         %Синтез входного сигнала...
```

```
x(1) = 1;                              %...в виде единичного импульса
out = zeros(1, L); %Создание выходного вектора нулей размера L
plotout = stem(out); %визуализация ИХ фильтра
s = serial('COM7','BaudRate',9600,'InputBufferSize', 4);
% Настройка последоват. порта, к кот. подкл. отладочная плата
fopen(s); %открытие порта
i = 0;
while(i < L);                                          %Основной цикл
i = i+1;                                           %Инкремент счетчика
fwrite(s,x(i),'float');
% Передача i-го отсчета в контроллер в виде четырехбайтного
числа
data = fread(s,1,'float');              %Чтение выходного отсчета
k = 1;
while (k < L);                    %Цикл смещения выходных отсчетов
out(k) = out(k+1);
k = k+1;
end
out(L)= data; %Запись выходного отсчета в последнюю ячейку мас-
сива
set(plotout,'YData',out) %вывод графика
drawnow                     %визуализация графика
end
fclose(s);            %закрытие COM-порта
disp ('end')
```

8. Сохраните полученный script-файл.
9. Запустите отладчик: «Target → Debug Active Project» или нажмите кнопку  . При появлении сообщения об ошибках в программном коде их необходимо исправить. При отсутствии ошибок отладчик автоматически произведет форматирование памяти микроконтроллера и запишет в нее новый исполняемый код.
10. Запустите исполняемую программу: «Target → Run» или во вкладке «Debug» нажмите кнопку  . Убедитесь в работе отладочной платы.
11. В MATLAB запустите script-файл. Получите график импульсной характеристики.
12. В Code Composer Studio приостановите выполнение исполняемой программы – нажмите: «Target → Halt» или кнопку  .
13. Для остановки выполнения исполняемой программы нажмите: «Target → Terminate All» или кнопку  .
14. Выполните выход из Code Composer Studio – нажмите: «File → Exit».
15. Выполните выход их MATLAB.

4.3. Требования к отчету

Отчет по лабораторной работе оформляется согласно принятым требованиям и включает:

1. Титульный лист с указанием: названия университета и кафедры, номера и названия лабораторной работы, ФИО обучающихся, номера их учебной группы, ФИО и должности преподавателя. Внизу страницы указывается город и год.

2. Раздел «Выполнение лабораторной работы» содержит: цель работы, основные этапы реализации нерекурсивного цифрового фильтра с рисунками и пояснениями.

3. Рисунок прямой структуры нерукурсивного цифрового фильтра заданного порядка.

4. График импульсной характеристики, полученный при реализации.

5. Выводы и ответы на контрольные вопросы.

6. Список литературы.

4.4. Контрольные вопросы

1. Дайте определение ЦФ (в широком и узком смыслах).
2. Какой фильтр называется КИХ-фильтром?
3. Какой фильтр называется БИХ-фильтром?
4. Приведите достоинства и недостатки нерекурсивного цифрового фильтра.
5. Приведите достоинства и недостатки рекурсивного цифрового фильтра.
6. Назовите основные методы синтеза КИХ-фильтров и приведите их основные этапы.
7. Назовите основные методы синтеза БИХ-фильтров и приведите их основные этапы.
8. Назовите основные типы m-файлов в MATLAB [5]. Для чего они используются и чем отличаются от mat-файлов?
9. Дайте определение импульсной характеристике.
10. Запишите разностные уравнения общего вида рекурсивной и нерекурсивной ЛДС.
11. Дайте определение АЧХ и ФЧХ.
12. Напишите формулы передаточных функций общего вида рекурсивной и нерекурсивной ЛДС.

4.5. Список литературы

3. *Солонина, А. И.* Основы цифровой обработки сигналов / А. И. Солонина, Д. А. Улахович, Е. Б. Соловьева, С. М. Арбузов. – СПб. : БХВ–Петербург, 2005. – 749 с.
4. *Солонина, А. И.* Цифровая обработка сигналов в зеркале MATLAB : учеб. пособие / А. И. Солонина. – СПб. : БХВ–Петербург, 2018. – 560 с.

5. *Солонина, А. И.* Цифровая обработка сигналов и MATLAB: учеб. пособие / А. И. Солонина, Д. М. Клионский, Т. В. Меркучева, С. Н. Перов. — СПб. : БХВ-Петербург, 2013. – 512 с.