ЛАБОРАТОРНАЯ РАБОТА №5

МОДЕЛИРОВАНИЕ ЭКСПЕРИМЕНТАЛЬНЫХ СИГНАЛОВ И ОБРАБОТКА ДАННЫХ В РЕАЛЬНОМ МАСШТАБЕ ВРЕМЕНИ

А.М.АНОХИНА, А.Г.МЯСНИКОВ, С.И.СВЕРТИЛОВ

ВВЕДЕНИЕ

Экпериментальная установка должна надежно регистрировать и эффективно разделять близкие события. Следовательно, обработка и анализ данных начинается еще в процессе регистрации события с использованием последних технических достижений в области электроники и вычислительной техники. Среди компонентов детектора присутствуют электронные схемы с высокой степенью интеграции, из которых строятся системы с распределенным интеллектом, а именно процессоры данных и памяти в составе детекторной электроники; полупроводниковые кристаллы памяти; быстродействующие спецпроцессоры, выполняющие алгоритм отбора за микросекундные времена; микропроцессоры, применение которых позволяет принимать даже довольно сложные решения за миллисекунды; системы считывания и стандартизованного доступа к магистралям; распределенные параллельные процессоры, позволяющие организовать многоуровневый триггер и фильтрацию и, следовательно, значительное снижение потоков данных на возможно ранних этапах, до записи на носители массовой памяти. В последние годы существенную модификацию претерпели процессоры цифровых сигналов, в основном, как отклик на требования из областей распознавания образов и речи, быстрого преобразования Фурье и цифровой фильтрации. С помощью предлагаемых упражнений студенты должны научиться общению с одним из таких процессоров (ADSP2181) на языке программирования высокого уровня СИ.

ЗАДАЧА 1.

"Анализатор амплитудного спектра импульсов".

1. Постановка задачи.

Каждому факту регистрации частицы соответствует электрический

импульс на выходе измерительного преобразователя. Амплитуда этого импульса соответствует энергии зарегистрированной частицы.

Задача анализатора энергетического спектра сводится к определению амплитуды каждого импульса и построению спектра амплитуд импульсов. По оси X обычно откладывается энергия пришедшей частицы, а по оси Y - количество пришедших частиц за время накопления, то есть плотность вероятности прихода частиц с данной энергией. Регистрацию импульсов производят, разбив область вероятных амплитуд на ряд полос, называемых каналами.

Определение амплитуды импульса производится при помощи пикового детектора, а определение номера канала - при помощи набора дискриминаторов, пороги которых настроены на соответствующие энергии. Число импульсов, попавших в канал, подсчитывается соответствующим счетчиком. Количество счетчиков соответствует числу дискриминаторов. Результатом работы анализатора является массив значений вероятности прихода импульсов с амплитудой, попадающей всоответствующие амплитудные каналы - т.е. массив вероятностей амплитуд.

2. Задание

На базе системы цифровой обработки сигналов EZKIT-Lite разработать анализатор амплитудного спектра импульсов.

Изучить результат работы анализатора при помощи Имитатора импульсов и осциллографа.

Частота оцифровки входного сигнала - 48 кгц.

Количество амплитудных каналов - 32.

Распределение каналов по амплитудному диапазону - равномерное.

Диапазон амплитуд импульсов - 0.05В .. 1 В.

Длительность импульса - 100 мкс.

Максимальная частота следования импульсов - 5 кГц.

3. Состав установки.

Персональная ЭВМ AT486.

Система ЦОС EZKIT-Lite.

Имитатор импульсов, поступающих с детектора частиц.

Осциллограф.

4. Методические указания.

4.1. Составьте блок-схему алгоритма работы анализатора.

В качестве первого этапа моделирования выведите на экран осциллографа шкалу анализатора со ступеньками заданых вручную числом событий по каждому из каналов. Например по первому каналу 1000 отсчетов, по второму 1500, по третьему 2000 и т.д.
Фиксированные числа по первому и последнему каналу анализатора можно использовать в качестве точек отсчета - границ шкалы анализатора.

4.2. Реализуйте алгоритм анализатора в виде программы на языке СИ для процессора ADSP2181. В качестве заготовки программы на языке СИ используйте файл **MAINTST.C** в подкаталоге LAB4. В ПРИЛОЖЕНИИ 1 приводится распечатка файла MAINTST.C.
Командный файл для компиляции программы - T.BAT.

4.3. Проверьте функционирование программы на системе EZKIT-Lite при помощи генератора импульсов и осциллографа.

5. Алгоритм анализа.

5.1. Входными данными для анализатора является оцифрованный входной сигнал, поступающий по каналу АЦП2.

5.2. Пиковое детектирование.

Так как в данном случае период дискретизации АЦП существенно меньше длительности импульсов, поступающих с детектора, примените цифровой метод пикового детектирования.
Алгоритм пикового детектирования состоит в следующем:
- происходит анализ цифровых отсчетов входного сигнала с целью нахождения минимума;
- по превышению входным сигналом заданного порога P1 над минимумом принимается решение о наличии нарастающей части импульса;
- происходит анализ цифровых отсчетов с целью нахождения максимума;
- при поступлении отсчета, меньшего, чем порог P2, относительно максимума, принимается решение о спаде сигнала, и , следовательно, о регистрации найденного максимума;
- регистрация производится в массиве счетчиков, в элементе, соответствующем данной амплитуде;
- анализ продолжается циклически.

5.3. Вывод результатов.

Результаты анализа выводите на ЦАП.
По каналу ЦАП2 периодически выводите накопленные значения амплитудного спектра. Каждому каналу должна соответствовать на экране осциллографа ступенька длительностью 100 мкс.

По каналу ЦАП1 необходимо выводить импульс синхронизации, начало которого соответствует началу вывода первого канала на ЦАП2.

5.4. Накопление данных.

Используйте для накопления массив 16-битных счетчиков. Предусмотрите останов процесса накопления при возникновении переполнения одного из счетчиков.

ЗАДАЧА 2.

"Генерация сигналов заданной формы с помощью сигнального процессора ADSP2181, моделирование линии задержки и формирователя со следящим порогом".

1**. Постановка задачи.**

Одним из наиболее распространенных детекторов частиц является сцинтилляционный счетчик, в котором часть теряемой заряженной частицей энергии идет на возбуждение атомов сцинтиллирующей среды. При переходе атомов в основное состояние, часть энергии может пойти на образование фотонов. Передний фронт электрического сигнала на выходе ФЭУ определяется длительностью световой вспышки в сцинтилляторе, которая, в свою очередь, определяется временем высвечивания. Время спада импульса определяется временем разряда емкости, на которой интегрируется токовый импульс.

Для пластических сцинтилляторов время высвечивания, в зависимости от типа вещества, лежит в интервале от 1.3 до 4 нс. "Медленными" детекторами считаются сцинтилляционные счетчики с кристаллами NaI(Tl) или CsI(Tl). Время высвечивания этих сцинтилляторов, а следовательно, и длительность переднего фронта составляют 0.2–1,0 мкс. Сцинтилляционное излучение

превращается в электрический сигнал и усиливается фотоумножителем.

Еще одним быстродействующим детектором является черенковский счетчик. Частицы, движущиеся быстрее скорости света в радиаторе черенковского счетчика, испускают свет под определенным углом. Свет фокусируется при помощи зеркал или с помощью специальных световодов на фотокатод фотоумножителя.

Моделирование сигналов, подобных тем, что получаются в реальном эксперименте (по форме импульса, флуктуациям формы и амплитуды импульса) часто необходимо для того, чтобы определить оптимальный алгоритм преобразования и обработки сигнала. **В Задании 1** предлагается с помощью системы цифровой обработки сигналов EZKIT-Lite смоделировать непрерывную последовательность импульсов, подобных наблюдаемым в эксперименте.

Проблемой многих экспериментов является сбор данных. Из-за высоких скоростей поступления и больших объемов информации на входе регистрирующей аппаратуры в определенные промежутки времени система сбора данных оказывается неспособной к приему новых событий. Это время называется "мертвым временем". Уменьшение мертвого времени позволит увеличить число регистрируемых в единицу времени полезных событий. "Мертвое время" регистрирующей системы складывается из мертвого времени детекторов и мертвого времени электроники. Мертвое время детекторов колеблется от нескольких наносекунд для сцинтилляторов до микро- и миллисекунд для приборов, в которых необходимо восстанавливать высокое напряжение, таких как искровые камеры. Мертвое время электроники и системы сбора данных перекрывает широкий диапазон. У быстродействующих электронных систем мертвое время составляет наносекунды, а мертвое время ЭВМ может достигать нескольких минут, если, например, необходимо поставить новую магнитную ленту.

Мертвое время формирователей или дискриминаторов связано с разрядом конденсаторов или с восстановлением начальных условий в цепях с обратной связью. Для уменьшения мертвого времени необходимо, чтобы выходные сигналы детекторов были как можно короче.

Для ограничения сигнала по длительности применяют короткий коаксиальный кабель или дискретную линию задержки, короткозамкнутые на конце.

Для ограничения сигнала по длительности применяют короткий коаксиальный кабель или дискретную линию задержки, короткозамкнутые на конце.

Линии задержки - линейные системы, задерживающие сигнал без существенного изменения их формы. Линии задержки состоят из индуктивностей и емкостей, которые могут быть распределены равномерно по всей длине линии (в кабелях) или сосредоточены на определенных ее участках (в искусственных линиях задержки). Скорость распространения сигналов в кабелях $v=c/\varepsilon^{1/2}$, где $c = 3.10^{10}$ см/с – скорость света в вакууме, ε - диэлектрическая проницаемость вещества, заполняющего кабель (типичное значение $\varepsilon = 1.5$). Единицу длины линии сигнал проходит за время $\tau_3 =(LC)^{1/2}$, где L и C - индуктивность и емкость линии на единице длины. Время задержки импульсов в кабеле составляет ~5 нс. на каждый метр его длины.

На Рис.1 приведена схема использования ограничивающего кабеля для укорачивания импульса. Сигнал в точке А представляет собой суперпозицию выходного сигнала детектора и сигнала, отраженного в противофазе от короткозамкнутого конца кабеля и задержанного на 2τ, где τ - задержка в один конец.

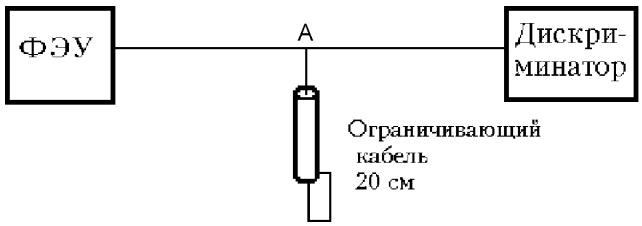

Рис.1. Ограничивающий кабель для укорачивания импульса.

В **Задании 2** предлагается с помощью системы цифровой обработки сигналов EZKIT-Lite смоделировать линию задержки необходимую для формирования сигнала.

Необходимость линии задержки можно проиллюстрировать также и на следующем примере. В эксперименте желательно передать всю информацию (амплитуду импульса, временные характеристики и т.д.) в ЭВМ для дальнейшего анализа. С другой стороны, выходной сигнал детектора требуется также и для выработки триггера - решения типа да/нет - принимать или не принимать событие. Поэтому выходной сигнал детектора расщепляется на две части (Рис.2.).

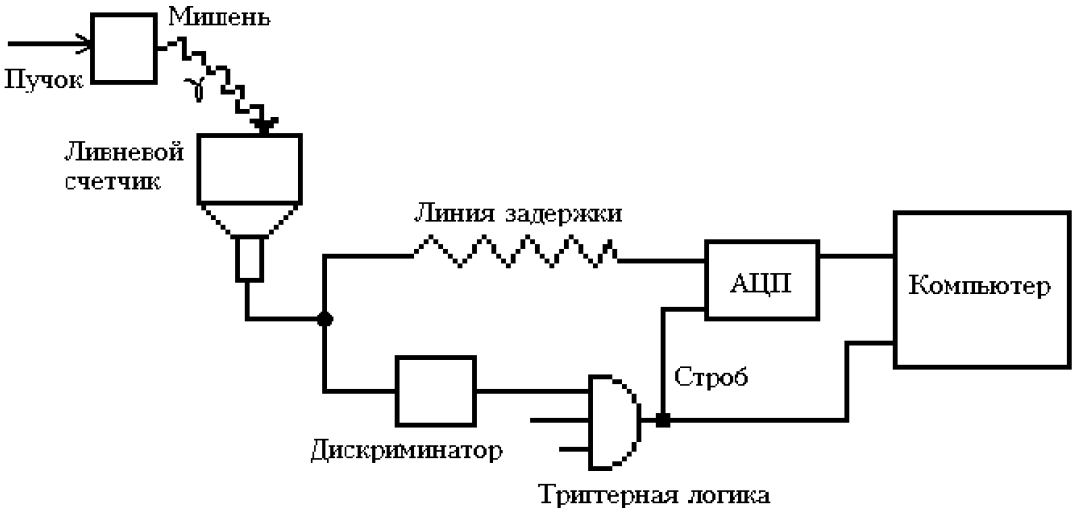

Рис.2. Схема, иллюстрирующая необходимость линии задержки

Одна часть доставляет известную долю выходного сигнала в АЦП, другая направляется в триггер. При выработке выполняются несколько логических операций, что требует определенного времени в диапазоне от 50 нс до нескольких микросекунд. Когда триггер появляется на входе вентиля АЦП, первоначальный сигнал уже потерян. Исходную информацию необходимо задержать и сохранить с помощью линии задержки.

Линии задержки могут использоваться в схемах формирования сигналов для временной привязки. Очень часто импульсы детекторов имеют большие амплитудные разбросы, а погрешность временной привязки должна быть значительно меньше времени нарастания сигналов. Особенно это относится к "медленным" детекторам с длительностью переднего фронта порядка 0.5мкс. При работе с такими детекторами вести привязку по переднему фронту, как правило, нецелесообразно из-за амплитудного разброса. Схемы, в которых корректируется амплитудный разброс с помощью метода **следящего порога**, достаточно универсальны, т.к. они свободны от погрешностей статистических флуктуаций амплитуды. Идея метода поясняется схемой и временными диаграммами, приведенными на Рис.3 и Рис.3а. Поступающие на вход схемы привязки сигналы разветвляются по двум направлениям. В одной ветви сигнал проходит через линию задержки, которая задерживает сигнал на время **tз**, равное или несколько больше переднего фронта импульса **tн (tз ≥ tн)**. Во второй ветви сигнал инвертируется и ослабляется в **f** раз. Задержанный и инвертированный сигналы поступают на схему суммирования. Из временных диаграмм видно, что задержанные

сигналы с амплитудами $\mathbf{A_1}$ и $\mathbf{A_2}$ "насаживаются" на вершины инвертированных импульсов соответственно с амплитудами $\mathbf{fA_1}$ и $\mathbf{fA_2}$. Сформированные таким образом сигналы пересекают нулевую линию в одной точке. С выхода суммирующей схемы сформированыый сигнал поступает на дискриминатор, срабатывающий в момент пересечения нуля.

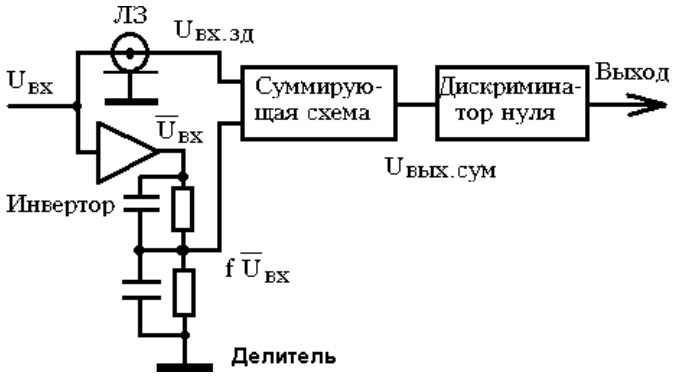

Рис.3. Схема временной привязки со следящим порогом.

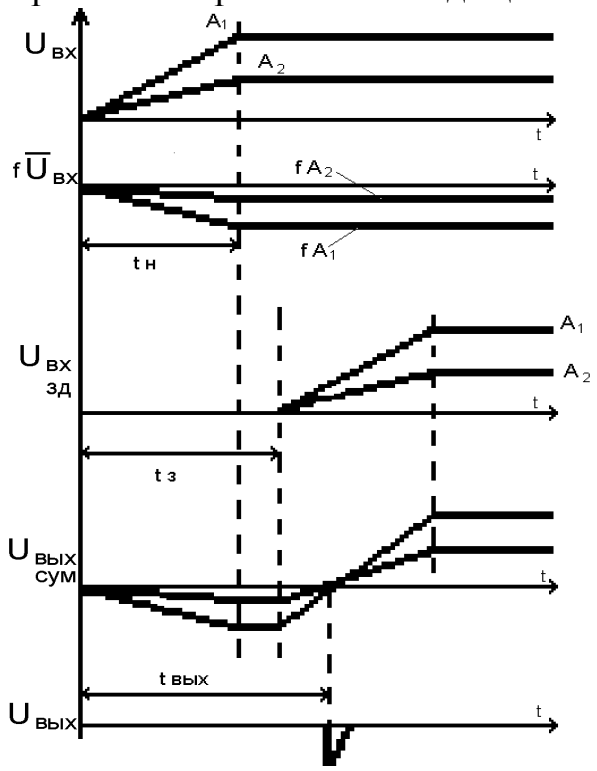

Рис.3а. Временные диаграммы. Выходной сигнал формирователя не зависит от амплитуды входных импульсов и задержан относительно начала линейно нарастающего сигнала на время $\mathbf{t_{вых} = (t_з + f \cdot t_н)}$.

В **Задании 3** предлагается с помощью системы цифровой обработки сигналов EZKIT-Lite смоделировать формирователь со следящим порогом.

2. Задание 1.

1.1. На базе системы цифровой обработки сигналов EZKIT-Lite с помощью программы на языке СИ для процессора ADSP2181 получить последовательность импульсов составленных из двух экспоненциальных зависимостей и имеющих существенно отличающиеся передний и задний фронты:

$$S = A \cdot e^{-t/\tau_1}(1 - e^{-t/\tau_2})$$ где $\tau_1 = 25$, $\tau_2 = 250$.

(Масштаб величин τ_1 и τ_2 связан с возможностями учебного варианта процессоров ADSP2181)

1.2. Состав установки.

Персональная ЭВМ AT486.
Система ЦОС EZKIT-Lite.
Осциллограф.

1.3. Методические указания.

Реализуйте алгоритм моделирования псевдоэкспериментального сигнала в виде программы на языке СИ для процессора ADSP2181.
В качестве заготовки программы на языке СИ используйте файл MAINEXP.C в подкаталоге LAB4. В ПРИЛОЖЕНИИ 2 приводится распечатка файла MAINEXP.C.
Командный файл для компиляции программы - T.BAT.

1.4 Алгоритм моделирования псевдоэкспериментального сигнала.

Рассчитать значения функции S в 200 точках, заполнить массив точек, описывающих импульс. Рассчитать полученную частоту следования импульсов.
Подобрать значение амплитуды A так, чтобы амплитуда моделируемых импульсов была ~ 1В. (Проверить по осциллографу).

Рассчитанный массив значений, описывающий необходимый вариант псевдоэкспериментального сигнала, выводить циклически по каналу ЦАП2.

По каналу ЦАП1 необходимо выводить импульс синхронизации, начало которого соответствует началу вывода первого элемента массива.

3. Задание 2.

Смоделировать формирователь со следящим порогом.

Для этого один сигнал задержать на несколько отсчетов - X, другой - умножить на -1· K и сложить с задержанным сигналом. Подобрать значения X и K.

Предусмотреть возможность переключения значений амплитуды A сигнала с помощью внешнего прерывания (кнопки INTERRUPT). Пронаблюдать на экране осциллографа неизменность момента пересечения нулевого уровня для сигналов с разной амплитудой – исследовать результат реакции формирователя на внешнее прерывание (кнопка INTERRUPT) (изменение амплитуды сигнала).

4. Задание 3.

Обработать полученные импульсы при помощи моделируемой линии задержки.

Для этого задержать сигнал на несколько отсчетов - X, умножить задержанный сигнал на -1· K и сложить с исходным сигналом. Предусмотреть возможность переключения значений X (10 - 50) и K(0.1 - 0.9) с помощью внешнего прерывания (кнопки INTERRUPT). Подобрать значения X и K так, чтобы в результате обработки сигнала не исказились бы его передний фронт и амплитуда, а задний фронт существенно обрезался.

При выводе результатов работы программы на осциллограф, исследовать результат реакции линии задержки на внешнее прерывание (кнопка INTERRUPT) - переключение значений X и K.

ЗАДАЧА 3.

"Генерация сигналов заданной формы со случайным распределением амплитуды с помощью генератора случайных чисел."

1. Постановка задачи.

Моделирование сигналов со случайными характеристиками, (например, амплитудой) подобных тем, которые реализуются в эксперименте, часто необходимо для того, чтобы оттестировать алгоритм преобразования и обработки сигнала. В настоящей работе предлагается с помощью генератора случайных чисел RAND смоделировать последовательность импульсов заданной формы (см. ЗАДАЧА 2), имеющих случайную амплитуду в диапазоне 0.5-1.В. В качестве первого этапа моделируется равномерно случайное распределение. Далее моделируются экспоненциальное и нормальное распределения с помощью **метода обратных функций**. Результат моделирования импульсов со случайными амплитудами проверяется с помощью анализатора амплитудного спектра импульсов (см. ЗАДАЧА 1).

Метод обратных функций.

Пусть нужно смоделировать или получить реализацию непрерывной случайной величины X с плотностью распределения f(x) путем преобразования последовательности случайных чисел, равномерно распределенных на интервале [0,1). Преобразование осуществляется с помощью некоторой функции. Запишем ее в виде: $X=\psi(R)$, связывающей случайные числа с равномерным распределением со случайными числами с заданным законом распределения.
Согласно методу обратных функций значение x случайной величины X является решением уравнения

$$F(x) = r, \qquad (*)$$

где r - значение случайной величины R, т.е. $x=F^{-1}(r)$.
Для большей наглядности можно обратиться к Рис.4, где изображена интегральная функция случайной величины X. Если по оси ординат брать случайные числа от 0 до 1 и находить для них, как показано стрелкой соответствующие числа на оси абсцисс, то полученные

значения будут являться значениями случайной величины X с функцией распределения F(x). Действительно, случайная точка с координатами (X; R=F(x)) перемещается только по кривой F(x). Абсцисса этой точки определяется ее ординатой. Проведем прямую AB, параллельную оси абсцисс на расстоянии F(x) от нее. Чтобы выполнялось неравенство X<x, случайная точка (X;R) должна попадать ниже линии AB, т.е. R(X<x) = P(R<F(x)).

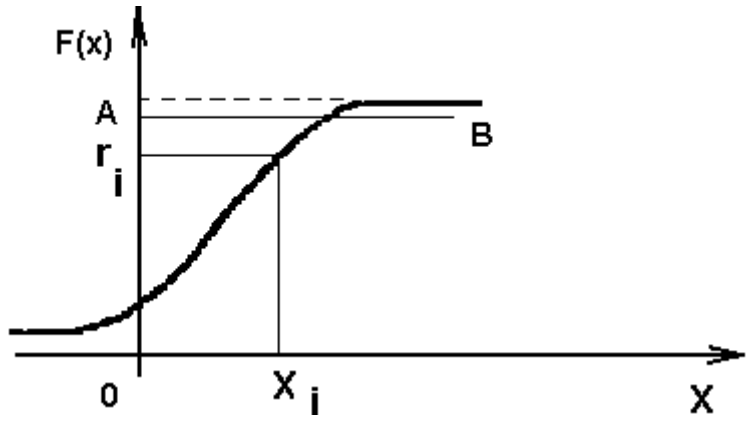

Рис.4. Иллюстрация к методу обратных функций.

Так как случайная величина R имеет плотность распределения, равную 1 на интервале [0,1), получим

$$P(X < x) = P(R < F(x)) = \int_0^{F(x)} f(R)dR = F(x)$$

Таким образом, процедура получения значений случайной величины с заданной функцией распределения **F(x)** заключается в следующем:
1) реализация случайной величины R, равномерно распределенной на интервале [0,1);
2) вычисление значений случайной величины X по формуле
$X=F^{-1}(R)$.

Пример 1.

Случайная величина X имеет экспоненциальный (показательный) закон распределения

$$f(x) = \lambda e^{-\lambda x}; x \geq 0, \lambda > 0;$$

$$F(x) = \int_0^x f(x)dx = \int_0^x \lambda e^{-\lambda x}dx = 1 - e^{-\lambda x},$$

где λ - параметр распределения, математическое ожидание $M(x)=1/\lambda$.

Требуется найти формулу для моделирования случайной величины X с помощью равномерно распределенной случайной величины $R = F(x) = 1 - e^{-\lambda x}$.

Находим обратную по отношению к F функцию. Имеем:

$$X = -\frac{1}{\lambda} ln(1 - R).$$

Так как 1 - R имеет то же самое распределение, что и R, то удобнее при нахождении случайной величины X пользоваться формулой

$$X = -\frac{1}{\lambda} lnR.$$

Случайные числа с экспоненциальным распределением вычисляются по формуле

$$x_n = -\frac{1}{\lambda} lnr_n.$$

Пример 2.

Случайная величина X имеет равномерное распределение на интервале [a,b). найти формулу для вычисления значений случайной величины X на этом интервале с помощью R.

Функция распределения случайной величины X на интервале [a,b) равна $F(x)=(x-a)/(b-a)$.

Составим уравнение согласно (*): $(x-a)/(b-a) = r$. Откуда

$x = a + r\,(b-a)$.

Пример 3.

Обращение функции распределения во многих случаях представляет собой сложную численную задачу. Поэтому этот метод не всегда применим. Однако для обратной функции F^{-1} можно найти достаточно хорошую аппроксимацию. Например, обратную

функцию плотности нормального распределения нельзя представить в виде простых, легко вычисляемых формул. Поэтому для получения случайных чисел с нормальным распределением иногда пользуются аппроксимацией:

$$\sqrt{\frac{2}{\pi}e^{-\frac{-x^2}{2}}} \approx \frac{2ke^{-kx}}{(1+e^{-kx})^2}, x > 0, \qquad (**)$$

где $k = (8/\pi)^{1/2}$.

если воспользоваться (**) для случайной величины с плотностью

$$f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}, -\infty < x < \infty \qquad (***)$$

то получим для функции (***) на положительной полуоси следующую аппроксимацию

$$r = \frac{2}{1+e^{-kx}} - 1,$$

из которой получаем приближенное выражение обратной функции

$$x \approx \frac{1}{k} ln \frac{1+r}{1-r}, 0 \leq r < 1.$$

Таким образом, если R - случайная величина, равномерно распределенная на интервале [0,1), то в соответствии с выше изложенным методом обратных функций величина

$$X = \frac{1}{k} ln \frac{1+R}{1-R},$$

имеет усеченное нормальное распределение

$$f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}, x > 0.$$

2. Задание.
2.1. На базе системы цифровой обработки сигналов EZKIT-Lite с помощью программы на языке СИ для процессора ADSP2181 получить непрерывную последовательность импульсов составленных из двух экспоненциальных зависимостей (см.ЗАДАЧА 2.) и имеющих случайным образом распределенную амплитуду А: а) равномерно (Пример 2), б) экспоненциально (Пример 1), в) нормально (Пример 3)

$$S = A \cdot e^{-t/\tau_1}(1 - e^{-t/\tau_2})$$

где τ_1= 25 , τ_2=250

2.2. Состав установки.

Персональная ЭВМ AT386.
Система ЦОС EZKIT-Lite N1.
Система ЦОС EZKIT-Lite N2.
Осциллограф.
2.3. Методические указания.

Используйте алгоритм моделирования псевдоэкспериментального сигнала в виде программы на языке СИ для процессора ADSP2181 отлаженный в ЗАДАЧЕ 2.
Рассчитайте значения функции S в 200 точках с фиксированной амплитудой А, определите частоту следования импульсов (частота оцифровки 48кгц), заполните массив точек, описывающих импульс.
В качестве заготовки программы на языке СИ используйте файл MAINEXP.C в подкаталоге LAB4.

а) Моделирование равномерного распределения:

проверьте подключение библиотеки (#include stdlib.h);

используйте генератор случайных чисел srand(), рассчитывающий последовательность случайных чисел от 0 до 32767 (RAND_MAX - зарезервированное имя);

смоделируйте равномерно случайную амплитуду А в пределах от А1 до А2 (Пример 1);

подберите значения A1 и A2 так, чтобы диапазон случайных амплитуд A составлял, приблизительно 0.5 - 1.0 В

перед выводом точек каждого импульса на ЦАП2, вызывайте генератор srand(), определяйте амплитуду и умножайте на значения элементов массива, описывающего сигнал; каждый последующий импульс должен выводится с новой, случайной амплитудой;

по каналу ЦАП1 выводите импульс синхронизации, начало которого соответствует началу вывода первого элемента массива;

результаты моделирования импульсов со случайными амплитудами проверяются с помощью анализатора амплитудного спектра импульсов (см. ЗАДАЧА 1); для этого с помощью системы ЦОС EZKIT-Lite N1 загрузить вариант программы, моделирующей последовательность импульсов со случайной амплитудой и подать сигнал на вход системы ЦОС EZKIT-Lite N2; в системе ЦОС EZKIT-Lite N2 загрузить программу, моделирующую амплитудный анализатор; результат работы анализатора амплитуд импульсов наблюдать на осциллографе.

б) Моделирование экспоненциального распределения:

аналогично способу, указанному в (а), за исключением вида распределения (см. Пример 1), параметр λ подобрать самостоятельно.

в) Моделирование нормального распределения (дополнительное задание):

аналогично способу, указанному в (а), за исключением вида распределения (см. Пример 3); параметр **k** и математическое ожидание амплитуды (в пределах 0.5-1В) подобрать самостоятельно.

Л И Т Е Р А Т У Р А

1.Н.Н.Дмитриева, А.С.Ковтюх, Б.Х.Кривицкий. Ядерная электроника. М.: Изд.МГУ, 1982.

2. В.А.Григорьев, А.А.Колюбин, В.А.Логинов. Электронные методы ядерно-физического эксперимента. М.: Энергоатомиздат. 1988.

3. А.П.Цитович. Ядерная электроника. М.: Энергоатомиздат. 1984.

4. Р.Бок, Х.Грот, Д.Ноц, М.Реглер. Методы анализа данных в физическом эксперименте. М.: Мир. 1993.

5. В.М.Иванова. Случайные числа и их применение. М.: Финансы и статистика. 1984.

ПРИЛОЖЕНИЕ 1.

ЗАГОТОВКА ПРОГРАММЫ, МОДЕЛИРУЮЩЕЙ РАБОТУ АМПЛИТУДНОГО АНАЛИЗАТОРА

```
/* EZ-LAB example of Impulse Amplitude Spectrum Analyzer */

#include <math.h>
#include <stdlib.h>
#include "cdef2181.h"
#define        MAX_INT  (32767)
#define bufset( buf, val, size) \
  ( { int _i; for( _i=0; _i < size; _i++)    buf[ _i] = val; } )

/* adc/dac flags & buffers */
int volatile tx_flg = 0;     /* adc input int. flag */
int volatile rx_flg = 0;     /* dac output int. flag */
int volatile ie_flg = 0;     /* button IRQE pressed flag */
extern volatile int rx_buf[3];     /* adc input data buffer */
extern volatile int tx_buf[3];     /* dac output data buffer */

/* channels definition and count buffer */
#define  MAXCHN  32            /* number of channels */
#define  MAXLEN  32            /* length of one channel in
                                  output gistorgam */
int COUNTS[ MAXCHN];                /* the amplitude spectrum counters
*/

void main()
{
const int LIM1 = 100,              /* the beg. of impulse treshold */
          LIM2 = 100;              /* the maximum (end of imp.) treshold */
const int SYNC_HIGH = 30000,       /* output sync high value */
          SYNC_LOW = 0;            /* output sync low value */
int MIN = 30000,           /* current min. value */
   MAX = 0,                        /* current max. value */
   IMP_FLG = 0,            /* 1 - beg. of impulse found */
   STOP_FLG = 0;                   /* 1 - stop count */
int CHAN = 0,                      /* current channel to output */
   LENGTH = 0;                     /* current channel outputted length */
int CHNNUM;                        /* channel to increment counter */

extern void ez_init();
```

```
  ez_init();                    /* init ezkit sport and codec */

  bufset(COUNTS, 0, MAXCHN);     /* clear COUNTS */

/* infinite loop */
  while (1)
  {
   if(ie_flg)                   /* button pressed? */
   {
    ie_flg = 0;

    asm("toggle fl1;");

    STOP_FLG = (STOP_FLG)?  0 : 1;
    if( ! STOP_FLG)
        {
       bufset(COUNTS, 0, MAXCHN);
        }
   };

   if(rx_flg)                   /* data received? */
   {
    rx_flg = 0;

    if( ! STOP_FLG)
    {
       register int D;
       D = rx_buf[2];               /* data from adc2 */
/* WORK AROUND WITH  D  VALUE  AND  COUNTS  ARRAY */

  АЛГОРИТМ АМПЛИТУДНОГО АНАЛИЗА С ПРИМЕНЕНИЕМ
  ЦИФРОВОГО МЕТОДА ПИКОВОГО ДЕТЕКТИРОВАНИЯ

    }
   };

   if( tx_flg)                  /* previous data transmitted? */
   {
    tx_flg = 0;
    tx_buf[1] = (CHAN == 0)?  SYNC_HIGH : SYNC_LOW; /* sync to
dac1 */
    tx_buf[2] = COUNTS[ CHAN];                  /* data to dac2 */
```

```
    if( ++LENGTH > MAXLEN)
    {
       LENGTH = 0;
       if( ++CHAN > MAXCHN) CHAN = 0;
    }
  };

 }     /* while(1) */
}
```

ПРИЛОЖЕНИЕ 2.

ЗАГОТОВКА ПРОГРАММЫ, МОДЕЛИРУЮЩЕЙ ПОСЛЕДОВАТЕЛЬНОСТЬ ИМПУЛЬСОВ

```
#include "cdef2181.h"
#include "math.h"
#define         MAX_INT  (32767)
const int SYNC_HIGH = 30000,        /* output sync high value */
          SYNC_LOW = 0;             /* output sync low value */
#define bufset( buf, val, size) \
  ( { int _i; for( _i=0; _i < size; _i++)    buf[ _i] = val; } )
const int amp=10000;
const float pi=3.141592654;
int n=0,
    fr_num=0,
    sig=0,
    i=0,
    STOP_FLG = 0,               /* 1 - stop count */
    CHAN = 1;                   /* current channel to output */

int fr[10] = {100,200,300,400,500,600,700,800,900,1000};
int num[1000];

/* adc/dac flags & buffers */
int volatile tx_flg = 0;     /* adc input int. flag */
int volatile rx_flg = 0;     /* dac output int. flag */
int volatile ie_flg = 0;     /* button IRQE pressed flag */
extern volatile int rx_buf[3];     /* adc input data buffer */
extern volatile int tx_buf[3];     /* dac output data buffer */

void init_num()
   {int i;
    for(i=0;i<1000;i++) num[i]=0;
   }

void count(int f)
{}

void main()
{
extern void ez_init();
  ez_init();                    /* init ezkit     sport and codec */
```

```
  init_num();
  count(fr[0]);
/* infinite loop */
  while (1)
  {
   if(ie_flg)                    /* button pressed? */
   {
    ie_flg = 0;
    asm("toggle fl1;");
    STOP_FLG = (STOP_FLG)?  0 : 1;
    /* if( ! STOP_FLG)*/
    init_num();
    if(++fr_num>9) fr_num=0;
    count(fr[fr_num]);
   };
   if(rx_flg)                    /* data received? */
   {
    rx_flg = 0;
    if( ! STOP_FLG)
    {
       register int D;
       D = rx_buf[2];              /* data from adc2 */
       sig=D;
       }
       };
   if (tx_flg)                   /* previous data transmitted? */
   {
    tx_flg = 0;
    tx_buf[1] = (CHAN == 0)?  SYNC_HIGH : SYNC_LOW;/* sync to
dac1 */
    CHAN=1;
    tx_buf[2] = num[i];                  /* data to dac2 */
    if(++i>fr[fr_num]) i=0;
   };
  }      /* while(1) */
}
```