Практикум
технические средства автоматизации
научных исследований

Автоматизация контрольно-измерительной аппаратуры с помощью языка команд SCPI. Цифровая фильтрация

Методические указания к лабораторной работе №7
(редакция от 12.2016)

Новосибирск
2016

Данная лабораторная работа состоит из двух взаимосвязанных частей. Первая часть посвящена ознакомлению с подходом к автоматизации контрольно-измерительной аппаратуры при помощи командного языка SCPI. Студентам предлагается реализовать интерфейс удаленного управления генератором сигналов по последовательному каналу связи. Во второй части работы студенты знакомятся с основами цифровой фильтрации сигналов и применяют полученные знания при решении физической задачи – фильтрации аудиосигнала от нежелательных частотных составляющих.

Составитель:
Орешонок В.В.

Рецензент:
Бехтенев Е.А.

Новосибирский государственный
университет, 2016

Введение

Вопрос автоматизации контрольно-измерительного оборудования имеет довольно богатую историю. Приборы с возможностью компьютерного управления были впервые представлены в 1960-х годах: однако, тогда они были редкостью, и при проектировании подобных устройств производители еще не задумывались о реализации единого подхода к их автоматизации.

По мере того как количество автоматизированных устройств росло (а заодно множились разнообразные интерфейсы и протоколы связи), сформировались и потребности в выработке единых принципов автоматизации оборудования. В 1970-х и 1980-х был разработан, а затем дополнен стандарт IEEE 488 (GPIB), определивший принципы конструктивной (разъемы, кабели) и электрической совместимости устройств; а также протокол передачи данных между приборами и компьютером.

Еще одним логичным шагом явилась разработка в начале 1990-х единого языка команд для автоматизированных устройств – SCPI. Цель SCPI состоит в упрощении подхода к автоматизации оборудования за счет создания единообразной системы команд и форматов данных для всех поддерживающих стандарт инструментов независимо от их типа и производителя.

Познакомиться с языком команд SCPI и применить его для управления автоматизированным генератором сигналов вы сможете в первой части данной лабораторной работы. Вторая часть работы посвящена знакомству с основами цифровой фильтрации и их применением в решении физической задачи.

Последовательный интерфейс связи RS-232

RS-232 (сокр. от *Recommended Standard 232*), представляющий собой физический уровень асинхронного интерфейса передачи данных UART, зачастую используется для подключения контрольно-измерительных приборов и других специальных устройств к персональным компьютерам [1]. Хотя используемое в данной работе оборудование не потребует от вас глубоких знаний данного стандарта, приведенная ниже информация будет полезной и позволит составить представление о его основах и особенностях.

Физические основы

Интерфейс RS-232 соединяет два устройства: линия передачи *TxD* одного устройства соединяется с линией приема *RxD* другого и наоборот. Такое соединение называется *полнодуплексным* и позволяет вести прием и передачу данных независимо. Информация передается по линии связи

последовательным двоичным сигналом. Логическому ‘0’ соответствует положительное напряжение от +5 до +15 В (для передатчика), а логической ‘1’ – отрицательное напряжение от -5 до -15 В (для передатчика). Способ кодирования, при котором логическая единица представлена низким уровнем, а логический нуль – высоким, называют *инверсным*. Максимальное расстояние передачи данных согласно стандарту составляет до 15 м.

Формат передачи данных

Данные в RS-232 передаются в последовательном коде побайтно. Формат передачи показан на Рис. 1: данные (5, 6, 7 или 8 бит в зависимости от настроек соединения) обрамляются стартовым битом, битом четности (не всегда) и одним или двумя стоп битами.

Рис. 1. Формат передачи данных RS-232

Пассивным состоянием линии является логическая ‘1’, стартовый бит всегда логический ‘0’. Бит четности используется для контроля качества и обнаружения ошибок при передаче данных: для этого есть несколько различных алгоритмов, рассматривать которые мы не будем. Стоп бит (или биты) всегда равен ‘1’ и обеспечивает минимальную паузу между передачей двух байт.

Обнаружив стартовый бит, приемник начинает фиксировать состояние линии через определенные равные интервалы времени, записывая, тем самым, биты данных. Для успешного обмена очень важно, чтобы тактовые частоты принимающего и передающего устройств были настроены одинаково. Также, одинаковым должен быть и формат посылок для обоих устройств (количество битов данных, наличие бита четности и тип проверки, количество стоп битов).

Физическая скорость передачи по последовательному интерфейсу измеряется в *бодах* (что для двоичного сигнала эквивалентно *бит/с*) и выбирается из ряда: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400 и далее.

Лабораторное оборудование

Схема подключения устройств, используемых в данной работе, изображена на Рис. 2. На ней представлен генератор функций GFG-3015, управляемый от персонального компьютера при помощи SCPI-команд, передаваемых посредством последовательного интерфейса RS-232. Выход

генератора подключен к установленному на шине терминального блока аудиомодулю. Данное устройство позволяет воспроизводить сгенерированные аудиосигналы и осуществляет их регистрацию. Полученный таким образом аналоговый сигнал поступает на вход *ai4* модуля ввода-вывода NI PXI-6251 по шине терминального блока. Модуль ввода-вывода синхронизирован с генератором (для этого синхросигнал генератора подан на вход *APFI0* модуля), а PXI-контроллер отвечает за передачу регистрируемых данных в персональный компьютер.

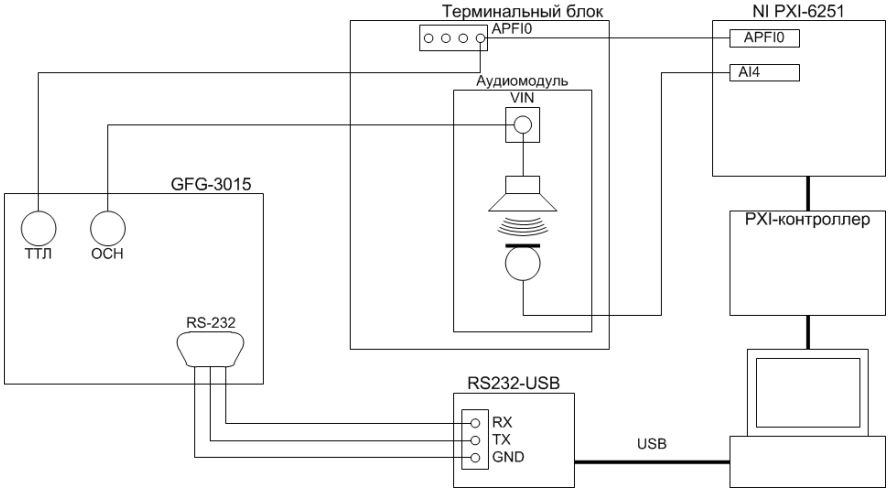

Рис. 2. Блок-схема подключения лабораторного оборудования

Генератор GFG-3015

GFG-3015 (Рис. 3) является современным цифровым генератором сигналов специальной формы. Его выходные характеристики приведены ниже:

- формы сигналов непрерывной генерации: синусоидальный, прямоугольный, треугольный, пилообразный;
- диапазон выходных частот 0.01 Гц – 15 МГц (максимальное разрешение 10 мГц);
- диапазон выходных амплитуд 0.01 В – 10 В с разрешением 10 мВ (на согласованную нагрузку);
- регулируемое смещение постоянного уровня ±5 В с разрешением 10 мВ (на согласованную нагрузку) и коэффициент заполнения (до 80%);
- встроенный частотомер и функция внешнего управления частотой;

- модуляция: амплитудная, частотная;
- выходы синхросигнала и преобразователя «частота-напряжение»;
- возможность удаленного доступа посредством интерфейса RS-232.

За кратким руководством пользователя GFG-3015 обратитесь к *Приложению 1*, воспользуйтесь печатной *Инструкцией по эксплуатации* или [2].

Рис. 3. Генератор GFG-3015 и преобразователь RS232-USB

Преобразователь RS232-USB

Хотя на рубеже 1990-х – 2000-х годов COM-порт (для реализации последовательного асинхронного интерфейса связи) был неотъемлемой частью персонального компьютера, сегодня его уже не так легко встретить в списке периферийных устройств. В то же время, многие современные приборы продолжают использовать для удаленного доступа интерфейс RS-232: после вытеснения COM-порта из состава оборудования персональных компьютеров, в обиход вошли так называемые *преобразователи интерфейса USB-UART*. Эти внешние устройства, подключаемые к порту USB (и питаемые, чаще всего, от него же), после установки соответствующих драйверов являются на программном уровне (а также на физическом уровне по выходу) *виртуальным COM-портом*. Одно из таких устройств – используемый в данной работе преобразователь RS232-USB, разработанный на базе интегральной схемы FT232R (см. Рис 3); он служит для реализации интерфейса связи генератора GFG-3015 с персональным компьютером. Преобразователь имеет клеммный терминал для подключения к последовательному порту генератора, а также USB разъем, подключаемый посредством кабеля к компьютеру. На корпусе устройства предусмотрены светодиодные индикаторы – питания (PWR) и активности

приемопередатчика (ACT). Последний будет мигать, отображая обмен данными в линии связи.

Аудиомодуль

Аудиомодуль (Рис. 4) предназначен для воспроизведения аудиосигналов в диапазоне слышимых частот от 20 Гц до 20 кГц с их последующей регистрацией. Электрические сигналы напряжением до 10 В переменного и постоянного тока подаются на согласованный вход *VIN* модуля и поступают в электродинамический громкоговоритель (динамик), который преобразует электрические колебания в акустические. Регистрация воспроизведенного звука осуществляется посредством электретного микрофона, установленного в непосредственной близости от динамика. Выходной сигнал микрофона после усиления поступает по шине терминального блока на вход *ai4* модуля ввода-вывода NI PXI-6251.

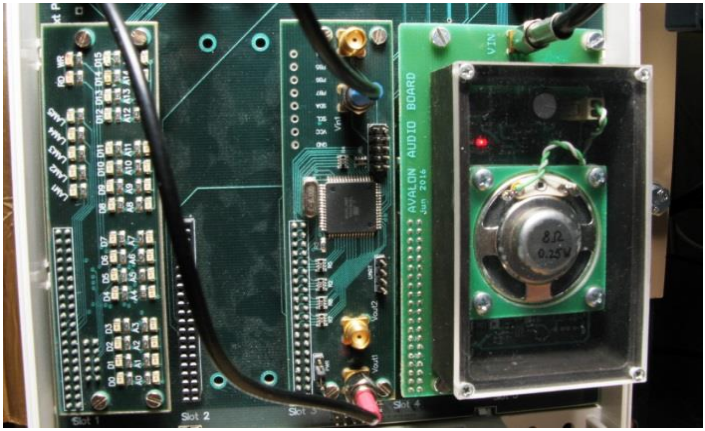

Рис. 4. Аудиомодуль (справа), установленный в терминальном блоке

Модуль ввода-вывода NI PXI-6251

С многофункциональным модулем ввода-вывода NI PXI-6251 и его характеристиками вы уже хорошо знакомы по предыдущим лабораторным работам. Вам потребуется 16-разрядный АЦП модуля – его частота дискретизации в мультиплексированном режиме составляет 1 МГц: это позволит с запасом охватить звуковой диапазон сигналов, воспроизводимых аудиомодулем. Чтобы синхронизировать измерения с генерацией сигнала воспользуемся режимом запуска АЦП по внешнему триггеру: синхросигнал поступает от генератора GFG-3015 на вход *APFI0* модуля NI PXI-6251, расположенный на терминальном блоке.

Язык команд SCPI

SCPI (часто произносится как “скиппи”) – *Standard Commands for Programmable Instruments* (Стандартные команды для программируемых приборов) – командный язык, предназначенный для автоматизации различных приборов и устройств (осциллографов, генераторов, источников питания и др.).

Стандарт включает описание синтаксиса, структуры команд, а также форматов пересылаемых данных [3]. Используемые команды разделены по классам: кроме общепринятых (понимаемых любыми поддерживающими стандарт приборами) есть и специфические – присущие только конкретному классу устройств (например, источникам питания). Физический уровень связи стандарт SCPI не описывает: первоначально созданный на основе IEEE 488 (GPIB), он может быть использован с RS-232, Ethernet, USB и другими протоколами. SCPI-команды представляют собой последовательность ASCII-кодированных символов, отправляемых прибору по линии связи. Символы составляют набор ключевых слов с параметрами или без. Ответом на такую команду является также ASCII-кодированная строка.

Структура языка

Стандартная SCPI-команда (Рис. 5) состоит из следующих элементов: *заголовка команды*, *параметра* (если таковой требуется) и *разделителя команд* или *признака конца команды (терминатора)*.

Рис. 5. Структура SCPI-команды

Заголовки – это ключевые слова, определяющие назначение команды. Они подразделяются на две категории: общепринятые (их понимают приборы независимо от типа) начинаются символом «звездочка» '*', например,

```
*IDN?
```

а также специфические для данного класса приборов – они начинаются с двоеточия ':' –

```
:FREQuency ?
```

Заголовки, предназначенные для управления приборами, имеют иерархическую (древовидную) структуру, уровни разделяются двоеточиями:

```
:FUNCtion:WAVeform 1
```

Если команда предполагает параметр, то он включается после заголовка через символ пробела:

```
:AMPLitude:VOLTage 3.3
```

Для параметров выделяют четыре типа данных: *логический* (0, 1) *целый* (0, 3, 32), *вещественный с десятичной точкой* (1.5, 3.14, 28.2), *вещественный в экспоненциальной форме* (2.5E-1, 8.25E+1, 1E+3). В специальной литературе их обозначают <*Boolean*>, <*NR1*>, <*NR2*> и <*NR3*> соответственно; встречается, также, обозначение <*NRf*> – этот тип является объединением вышеперечисленных.

Кроме команд на изменение значений настроек, можно выполнить запрос текущего значения большинства параметров, добавив к записи команды вопросительный знак '?':

```
:OFFSet ?
```

Признаком конца команды для различных устройств могут служить разные символы: обычно, однако, это *Carriage Return* (т.н. «возврат каретки», <CR> = '\r' = 0x0D) или *Line Feed* («перевод строки», <LF> = '\n' = 0x0A) и их комбинации. В качестве же разделителя команд (в том случае, когда требуется отправить несколько команд одной посылкой) используется символ ';'.

Программные средства

Библиотека RS-232

Работа с последовательным портом передачи данных осуществляется в среде LabWindows посредством библиотеки *RS-232 Library*. С детальным описанием функций данной библиотеки можно ознакомиться с помощью справочной системы среды. Краткое описание функций представлено в *Приложении 2*.

Прежде чем можно будет начать передавать и принимать данные по последовательному интерфейсу (при условии, конечно, что запрашиваемое устройство подключено к COM-порту вашего компьютера и готово к работе), необходимо открыть последовательный порт и настроить его параметры связи. Для этих целей служит функция **OpenComConfig** – она принимает в качестве аргументов номер и название открываемого порта, скорость связи, настройки формата передаваемых данных и контроля четности, а также настройки размеров служебных буферов. В качестве статусной информации функция возвращает неотрицательное число, если открытие последовательного порта прошло успешно, или отрицательный код ошибки.

Имя, присвоенное преобразователю интерфейса USB-UART операционной системой, поможет узнать *Диспетчер устройств*: зайдите в меню **Пуск → Панель управления → Система → Оборудование →**

→ **Диспетчер устройств → Порты (COM и LPT).**
В выпадающем списке ищите устройство с названием *«USB Serial Port»* его имя и номер будут указаны в скобках (например, COM8).

Следующим важным шагом послужит установка временных ограничений на операции чтения и записи последовательного порта – за это отвечает функция **SetComTime**. Ее аргументами служат номер порта и величина устанавливаемого таймаута в секундах. Для операций чтения этот таймаут ограничивает ожидание ответа от запрашиваемого устройства, а для записи – обеспечивает выход из ситуации, когда выходной буфер последовательного порта заполнен, но данные не могут быть отправлены получателю в течение установленного интервала времени. Функция возвращает отрицательное число, если порт, указанный в аргументе, не открыт или значение таймаута задано некорректно.

Чтобы получить код текущей ошибки последовательного соединения, а также узнать ее расшифровку, воспользуйтесь функциями **ReturnRS232Err** и **GetRS232ErrorString**. За информацию о статусе COM порта отвечает функция **GetComStat**.

В целом, функцию инициализации последовательного порта можно оформить так:

```
#include <rs232.h>
/* Функция инициализации последовательного соединения по
номеру последовательного порта portNumber */
int InitCOM (int portNumber) {
  char PortName[5] = {(0)}; // для хранения имени порта
  sprintf (PortName, "COM%i", portNumber);
  // 1. Открываем последовательное соединение: настройки
  // (скорость связи 19200 бод; без контроля четности;
  // 8 бит в "слове" данных; 1 стоп-бит) соответствуют
  // параметрам соединения генератора по умолчанию.
  if (OpenComConfig(portNumber, PortName, 19200, 0, 8, 1,
                    0, 0) < 0)  goto Error;
  // 2. Настраиваем таймаут соединения.
  if (!SetComTime (portNumber, 0.4))
    return GetComStat (portNumber);
Error:
  // в случае ошибки - выводим всплывающее окно со
  // статусной информацией
  MessagePopup("Serial Configuration Error",
               GetRS232ErrorString (ReturnRS232Err ()));
    return -1;
}
```

Для обмена данными по последовательному интерфейсу связи в библиотеке *RS-232 Library* предусмотрены различные функции: мы воспользуемся двумя из них – **ComWrt** и **ComRdTerm.**

ComWrt предназначена для записи в заданный последовательный порт сообщения фиксированной длины. Функция возвращает количество байт сообщения, помещенных в выходной буфер порта, либо отрицательный код ошибки в случае неудачи.

Чтение входного буфера последовательного порта до обнаружения признака конца сообщения осуществляется при помощи функции **ComRdTerm.** В качестве аргументов она принимает номер порта, указатель на массив, куда будут считаны входящие данные, а также максимальную длину сообщения либо признак его конца (терминатор). В случае успеха функция возвращает количество принятых байт, в случае неудачи – отрицательный код ошибки.

Чтобы закрыть последовательный порт (например, при выходе из программы), воспользуйтесь функцией **CloseCom.**

Рассмотрим использование функций записи и чтения на примере запроса строки-идентификатора генератора GFG-3015. Данный запрос осуществляется командой *IDN?, а ответ на него содержит уникальный идентификатор устройства. Описание других команд, используемых для работы с генератором, ищите в *Приложении 1*.

```
#include <rs232.h>
// Constants
#define PORTNUM       8      // номер используемого порта
#define TERMCHAR      "\n"   // символ-терминатор
#define BUFSIZE       64     // размер буфера с данными
// Static global variables
static char Buf [BUFSIZE] = {(0)};
// The main entry-point function
int main (int argc, char *argv[])
{ int error = 0;
  /* initialize and load resources */
  nullChk (InitCVIRTE (0, argv, 0));
  errChk (panelHandle = LoadPanel (0,"test.uir",PANEL));
  // 1. Инициализация последовательного соединения.
  if (InitCOM (PORTNUM)) goto Error;
  // 2. Отправка команды *IDN?
  sprintf (Buf, "%s%s", "*IDN?", TERMCHAR);
  if (ComWrt (PORTNUM, Buf, strlen(Buf)) <= 0)
    MessagePopup ("Serial Write Error",
                  GetRS232ErrorString (ReturnRS232Err()));
```

```
  // 3. Прием ответного сообщения.
  else if (ComRdTerm(PORTNUM,Buf,BUFSIZE,TERMCHAR[0])<=0)
    MessagePopup ("Serial Read Error",
                  GetRS232ErrorString (ReturnRS232Err()));
  else
    // 4. Вывод ответа на индикатор Text Message.
    SetCtrlVal (panelHandle, PANEL_TEXTMSG, Buf);
  /* display the panel and run the user interface */
  errChk (DisplayPanel (panelHandle));
  errChk (RunUserInterface ());
    // 5. Задание предустановленных настроек генератора.
  sprintf (Buf, "%s%s", "*RCL 0", TERMCHAR);
  if (ComWrt (PORTNUM, Buf, strlen(Buf)) <= 0)
    MessagePopup ("Failed to Recall Settings",
                  GetRS232ErrorString (ReturnRS232Err()));
Error:
  /* clean up */
  if (GetComStat (PORTNUM) >= 0) CloseCom (PORTNUM);
  DiscardPanel (panelHandle);
  return 0;
}
```

Средства библиотеки tsanilib

Библиотека *tsanilib* вам уже знакома из опыта эксплуатации модуля ввода-вывода NI PXI-6251. Кроме этого, в ней реализованы функции, используемые для работы с оборудованием, применяемым в экспериментальных лабораторных работах.

Для синхронизации измерений с выходным сигналом генератора GFG-3015, необходимо запускать АЦП модуля NI PXI-6251 по внешнему синхроимпульсу, поступающему от генератора, а затем вычитывать получаемые выборки данных.

Выбор источника тактового сигнала АЦП и его частоты, а также выбор и настройка триггера запуска аналогового канала осуществляется с помощью функций **analogInClk** и **analogInTrigger**. В качестве первого аргумента обе функции принимают строку с названием источника синхронизации: для **analogInClk** необходимо указать *"OnboardClock"* (или, что равносильно, ввести пустую строку *""*), чтобы выбрать внутренний источник тактового сигнала; для **analogInTrigger** укажите порт *"APFI0"*, на который поступает сигнал внешней синхронизации.

Вторым аргументом у **analogInClk** служит задаваемое пользователем значение частоты дискретизации АЦП. Выбирая это значение, помните, что

его величина должна удовлетворять критерию Найквиста для регистрируемого сигнала, но не превышать 1 МГц – максимального значения для используемого АЦП.

У функции **analogITrigger** вторым аргументом задается уровень срабатывания триггера (он измеряется в вольтах, типичные значения 1 – 2 В).

Для регистрации выборки аналогового входного сигнала заданной длины используется функция **analogRead** – она принимает номер аналогового канала, указатель на массив для хранения регистрируемых данных, а также величину выборки, а возвращает (через указатель) количество измеренных значений.

Описание вышеперечисленных функций вы найдете в *Приложении 2*.

Задания

При выполнении данных заданий подключите генератор GFG-3015 напрямую к АЦП модуля NI PXI-6251 (используйте входы AI0 или AI1 терминального блока).

1. Для начала вам потребуется реализовать функции, позволяющие записывать и считывать параметры различных настроек генератора. Ниже приведен пример одной из таких функций – она позволяет задавать целочисленные параметры:

```
// Функция записи команды с целочисленным параметром
int SerWriteInt (const char command[], int param) {
  // создаем буфер сообщений
  char buf [1024] = {(0)}
  // запишем в буфер команду для отправки
  sprintf (buf, “%s %d%s”, command, param, TERMCHAR);
  // отправляем команду
  if (ComWrt (PORTNUM, buf, strlen(buf)) != strlen(buf))
     return(ReturnRS232Err());
  return 0;
}
```

   Функцию `SerReadInt`, а также `SerWriteFp`, `SerReadFp` (чтение и запись вещественных настроек) реализуйте самостоятельно. При работе с генератором руководствуйтесь *Приложением 1* и инструкцией по эксплуатации прибора – там вы найдете порядок действий для изменения настроек генератора с передней панели и список команд, позволяющих осуществить эти действия с помощью последовательного интерфейса.
2. Реализуйте интерфейс, позволяющий регулировать основные настройки генератора GFG-3015 (форму сигнала, частоту (Гц), амплитуду и смещение (В)), а также регистрирующий полученный сигнал с помощью

АЦП модуля NI PXI-6251 с использованием внешней синхронизации. Интерфейс должен выводить регистрируемый сигнал и его спектр на графики, обновляемые с частотой 1 Гц. За основу возьмите результаты практического задания 2 из лабораторной работы №1.

3*.Добавьте в интерфейс работы с генератором возможность настройки режима модуляции (амплитудной либо частотной – на ваш выбор).

Цифровые фильтры

На сегодняшний день задачи фильтрации сигналов получили широкое распространение в различных приложениях: научных, медицинских, индустриальных а также бытовых. Фильтры обычно используют для восстановления зашумленных сигналов, либо разделения сложных сигналов на составляющие. Решение данных задач можно поручить как аналоговым, так и цифровым фильтрам: аналоговые выделяются относительной простотой, дешевизной, а также широким динамическим диапазоном; в то же время цифровые фильтры позволяют достичь на их фоне существенно лучших характеристик [4].

Термины

Рассмотрим основные понятия, используемые в теории цифровой обработки сигналов. Зачастую сигналы, подвергаемые цифровой обработке, представляют собой результаты измерений различных физических величин (напряжений, токов, отклонений координат и т.д.), выполненные через равные временные интервалы (есть, конечно, и другие способы получения сигналов, но мы их обсуждать не будем). Говорят, что такие сигналы определены во *временной области*. Информация, которую сигналы несут во временной области, описывает *время* возникновения событий, а также их *амплитуду*. Кроме временного представления сигналов, полезным также бывает *частотное представление*, где отражены частотные составляющие исследуемых сигналов.

Линейный фильтр может быть охарактеризован *импульсной характеристикой*, *частотной (амплитудно-частотной) характеристикой (АЧХ)*, а также *реакцией на ступеньку*. Каждая из них полностью описывает фильтр (однако разными способами), что позволяет получить данные о поведении фильтра в различных условиях. Если известна одна из вышеперечисленных характеристик, две другие всегда могут быть вычислены. Что же они из себя представляют? Под импульсной характеристикой системы понимают её выходной сигнал при наличии на входе импульса (дельта функции). Аналогично, реакцией на ступеньку будет выходной сигнал системы, когда на вход подан сигнал с единичным положительным фронтом. Т.к. ступенчатый сигнал является результатом

интегрирования дельта функции, значит, проинтегрировав импульсную характеристику, мы получим реакцию на ступеньку (для дискретных сигналов интегрирование заменяется суммированием). Частотную же характеристику получим, применив к импульсной характеристике дискретное преобразование Фурье.

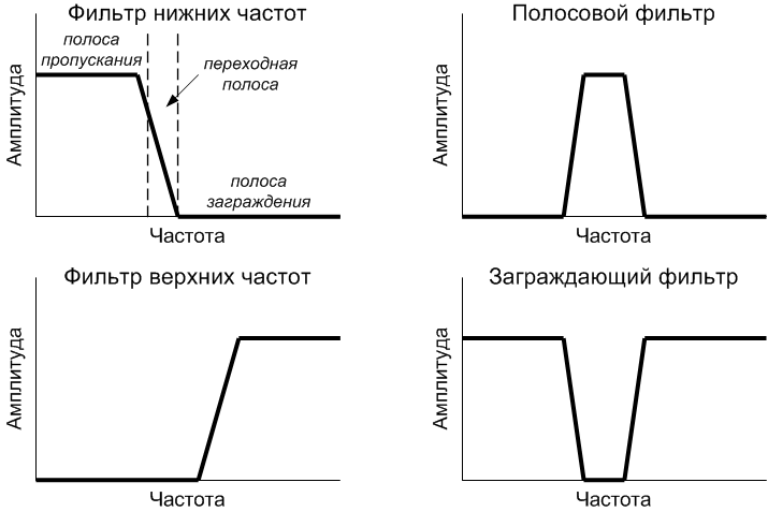

Рис. 6. Основные виды частотных фильтров

Рассмотрим четыре основные характеристики частотных фильтров, представленные на Рис. 6. Данные фильтры предназначены для подавления одних частотных составляющих сигналов при полном пропускании других. Термин *полоса пропускания* описывает диапазон пропускаемых частот, в то время как *полоса заграждения* относится к диапазону частот, которые фильтр блокирует; переход между ними составляет *переходную полосу* фильтра. Чтобы подавлять одни частотные составляющие, а другие пропускать без искажений, фильтр должен иметь плоскую полосу пропускания, а также достаточное затухание в полосе заграждения. Переход от полосы пропускания к полосе заграждения характеризуется *частотой среза*: в аналоговой электронике её значение определяется по уровню -3 дБ или падению амплитуды до 0.707 от исходного значения. Для цифровых фильтров это понятие не столь стандартизовано, но мы будем придерживаться его «классической» интерпретации.

Основы цифровой фильтрации

Наиболее общий алгоритм цифровой фильтрации заключается в свертке

обрабатываемого сигнала с импульсной характеристикой фильтра. При этом каждая точка выходного сигнала вычисляется масштабированием входных точек с последующим их суммированием:

```
/* Свертка входного сигнала X[1000] с ядром фильтра
H[100]. Результат - сигнал Y[1000].*/
for (int i = 100; i < 1000; i++) {
    Y[i] = 0;
    for (int j = 0; j <= 100; j++)
        Y[i] += X[i - j]*H[j];
}
```

Импульсную характеристику в этом случае называют *ядром* цифрового фильтра. Фильтры, в основу алгоритма которых положена свертка сигналов, называют фильтрами с *конечной импульсной характеристикой (КИХ)*.

Проектирование цифрового фильтра начинается с вычисления ядра ФНЧ: любой другой фильтр (ФВЧ, полосовой, полосно-заграждающий) пересчитывается на его основе при помощи различных методик, из которых мы рассмотрим *спектральную инверсию*.

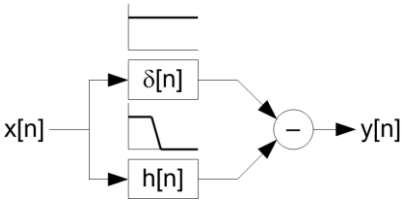

Рис. 7. Блок-схема алгоритма спектральной инверсии

Пусть сигнал *x[n]* подается на входы двух параллельных систем: одна из них пропускает сигнал без изменений, следовательно, её импульсной характеристикой будет дельта функция *δ[n]*, а другая представляет собой ФНЧ с импульсной характеристикой *h[n]* (Рис. 7). Выходной сигнал *y[n]* сформируем из разности выходов двух вышеприведенных систем: очевидно, что из исходного сигнала мы вычли все низкочастотные составляющие, выделенные ФНЧ, получив, таким образом, фильтр верхних частот. Если же взглянуть на частотную характеристику, то мы увидим, что спектральная инверсия приводит к горизонтальному отражению АЧХ системы, превращая ФНЧ в ФВЧ, а полосовой фильтр в заграждающий и наоборот. В алгоритмической интерпретации спектральная инверсия вычисляется как разность единичного импульса и исходного ядра фильтра:

```
/* Алгоритм спектральной инверсии для ядра H[100]. Меняем
знаки всех точек ядра, кроме центральной, на противополо-
жные. Значение центральной точки отнимаем от 1. */
for (int i = 0; i <= 100; i++)
    H[i] = (i != 50) ? (-H[i]) : (1 - H[i]);
```

У данного метода, однако, есть ограничения: во-первых, ядро исходного фильтра должно быть симметричным, а, во-вторых, единичный импульс должен располагаться по центру симметрии (это следует из необходимости равенства фаз вычитаемых сигналов).

Фильтр с оконной (весовой) функцией

Перейдем теперь к реализации одной из разновидностей частотных фильтров – фильтру с оконной (весовой) функцией. Такие фильтры обычно используют для разделения АЧХ сигнала на составляющие.

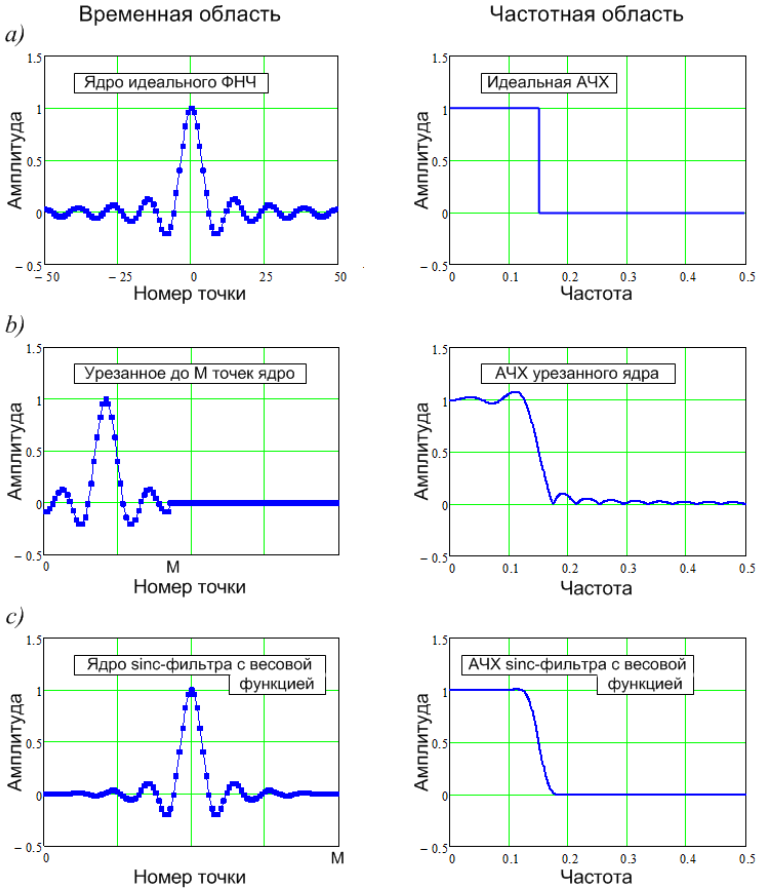

Рис. 8. Вычисление импульсной характеристики sinc-фильтра с весовой функцией:
a – идеальный ФНЧ и его характеристики; *b* – урезанное ядро;
c – характеристики sinc-фильтра с весовой функцией

Рассмотрим Рис. 8(a): частотной характеристике идеального ФНЧ свойственна плоская полоса пропускания с единичным коэффициентом усиления, бесконечное ослабление в полосе заграждения, а также переходная полоса, шириной которой можно пренебречь. Обратное преобразование Фурье, примененное к данной АЧХ, позволит нам получить импульсную характеристику (ядро) идеального ФНЧ. Эта кривая описывается так называемой sinc-функцией

$$h[i] = \frac{\sin(2\pi f_c i)}{i\pi}$$

Свертка сигнала с таким ядром позволит реализовать идеальный ФНЧ, однако, проблема заключается в том, что для этого sinc-функция $h[i]$ должна быть задана на отрезке (-∞; +∞), и здесь мы сталкиваемся с серьезным ограничением – ведь алгоритмически это ядро задать невозможно. Попробуем обойти ограничение: возьмем для $h[i]$ только $(M+1)$ точек, где M – четное, а все значения вне этой области проигнорируем; кроме того, сдвинем ядро на $M/2$ вправо просто для того чтобы индексация начиналась с нуля (Рис. 8(b)). Что же случилось с частотной характеристикой? В полосе пропускания обнаружились пульсации, а ослабление в полосе заграждения стало конечным – всё это результат «обрывающейся» импульсной характеристики фильтра. Для того чтобы ослабить данный эффект необходимо использовать *оконные* (их еще называют *весовыми*) *функции*: они позволяют сгладить разрывы на концах импульсной характеристики и, как следствие, улучшить АЧХ фильтра. Умножая нашу импульсную характеристику на оконную функцию, получим так называемый *sinc-фильтр с оконной (весовой) функцией* (Рис. 8(c)). Из нескольких доступных видов оконных функций наиболее распространенными являются *окно Хэмминга* и *окно Блэкмана*, приведенные ниже (в порядке следования):

$$w[i] = 0.54 - 0.46\cos\left(\frac{2\pi i}{M}\right)$$

$$w[i] = 0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right)$$

Окно Хэмминга имеет более узкую переходную полосу, чем окно Блэкмана; последнее, однако, обеспечивает лучшее затухание в полосе заграждения и меньшие пульсации в полосе пропускания, что делает его выбор предпочтительнее. В случае, когда весовые функции в явном виде не используются (или можно сказать, что мы умножили импульсную характеристику на единицу), говорят о *прямоугольном окне*.

Наконец, для проектирования оконного sinc-фильтра важно правильно выбрать два аргумента: частоту среза, f_c, и размер ядра, M. Здесь, в вышеприведенных формулах и далее под f_c понимают относительную частоту, т.е. отношение частоты среза к частоте дискретизации исходного

сигнала (её значение, следовательно, согласно теореме Котельникова, должно лежать в пределах от 0 до 0.5). Величина *M* влияет на быстродействие фильтра – ширину его переходной полосы: чем больше ядро фильтра, тем уже его переходная полоса. Учитывая всё вышесказанное, мы можем записать формулу для вычисления ядра sinc-фильтра нижних частот:

$$h[i] = K\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{\left(i-\frac{M}{2}\right)\pi}(0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right))$$

Несмотря на кажущуюся сложность, это выражение не содержит почти ничего, что мы бы не обсуждали выше: в нем можно разглядеть sinc-функцию, сдвиг индексации на *M/2*, а также окно Блэкмана.

Единственная новая добавка – множитель *K*, используемый для *нормировки* ядра: он нужен, чтобы фильтр давал единичный коэффициент усиления в полосе пропускания. На практике, сначала необходимо вычислить ненормированное ядро фильтра, а также сумму всех его элементов, а уже после этого ввести коэффициент нормировки:

```
/*  Нормировка ядра H[100].    */
//  1. Вычисляем ядро и его сумму.
for (int i = 0; i <= 100; i++) {
    H[i] = ((i != 50) ? (sin(2*Pi()*fc*(i-50))
                         / ((i-50)*Pi()))
                      : (2*fc)) * (0.54 - 0.46
                         * cos((2*Pi()*i)/100)));
    Hsum += H[i];
}
//  2. Нормируем ядро.
for (int i = 0; i <= 100; i++) H[i] /= Hsum;
```

Задания

При выполнении данных заданий подключите генератор GFG-3015 ко входу *VIN* аудиомодуля.

1. Напишите программу, вычисляющую нормированное ядро sinc-фильтра нижних частот с оконной функцией на основе введенных параметров. Программа должна отображать импульсную, а также амплитудно-частотную характеристики фильтра. Для вычисления ядра реализуйте функцию:

```
int CalcLPFKernel (unsigned char window,
                   unsigned int length, double cutoff,
                    double kernel[]);
```

где `window` задает тип используемого окна (прямоугольное, Блэкмана, Хэмминга), `length` – размер ядра (он всегда должен быть нечетным, чтобы ядро было симметричным относительно центра), `cutoff` – частоту среза в относительных единицах, а `kernel[]` – возвращает результат вычислений.

2. Добавьте интерфейс вычисления характеристик ФНЧ в программу работы с генератором. С помощью фильтра выделите постоянную составляющую из регистрируемого аудиосигнала. Также можно использовать ФНЧ для избавления сигнала от нежелательных высокочастотных составляющих (постоянное смещение при этом останется – чтобы убрать его вам потребуется выполнить п.5).
3. Реализуйте функцию, позволяющую по выбору вычислять ядро ФНЧ или ФВЧ. Она должна вызывать функцию `CalcLPFKernel` (а значит, унаследовать её набор аргументов), а также использовать, если необходимо, алгоритм спектральной инверсии.
4. С помощью ФВЧ выделите переменную составляющую из регистрируемого аудиосигнала.

5*.Измените функцию из п.3 так, чтобы она могла вычислять также характеристики полосовых и полосно-заграждающих фильтров.

6*.С помощью полученной программы выделите основную гармонику из регистрируемого аудиосигнала произвольной частоты и формы.

Список литературы

1. Описание RS-232: http://www.gaw.ru/html.cgi/txt/interface/rs232/

2. Описание генератора GFG-3015: http://www.gwinstek.com/en-global/products/Signal_Sources/Analog_Function_Generators/GFG-3015

3. Стандарт SCPI: http://www.ivifoundation.org/docs/scpi-99.pdf

4. The Scientist and Engineer’s Guide to Digital Signal Processing. Steven W. Smith, California Technical Publishing, 1999.

Приложение 1
Краткое руководство по эксплуатации генератора GFG-3015

Многофункциональный генератор сигналов GFG-3015 представляет собой источник, вырабатывающий сигналы синусоидальной, треугольной, прямоугольной формы, имеющий режимы амплитудной и частотной модуляции, а также формирования сигнальных пакетов и свипирования (сканирования по частоте). Для связи с персональным компьютером предусмотрен интерфейс RS-232.

Последовательность действий для изменения настроек генератора с передней панели приведена в таблице ниже.

Таблица 1

Изменение настроек генератора

Операция	Порядок действий	Примечания
1. Включение прибора и установка начальных параметров	Нажатием на кнопку **СЕТЬ** включите питание генератора. Осуществите установку параметров генератора «по умолчанию»: для этого последовательно нажмите кнопки **ПРЕФ** и **RS232**.	
2. Установка формы выходного сигнала	Форма выходного сигнала задается нажатием кнопки **ФОРМА** на лицевой панели. Выбранная форма сигнала отображается значком в верхней части индикационного табло генератора.	
3. Установка частоты выходного сигнала	Нажмите кнопку **ЧАСТОТА** – на табло замигает индикатор **ЧАСТ.** Текущее значение частоты сигнала отображается на индикаторе с указанием размерности. Изменить значение частоты можно прямым набором требуемой величины и вводом соответствующей размерности (кнопки **МГц/dBm**, **кГц/Vrms**, **Гц/Vpp**), а также кнопками **<** и **>** и вращаемым регулятором.	В случае превышения допустимых пределов установки частоты выходного сигнала, в поле частоты на индикаторе загорится надпись **E01**.
4. Установка	Нажмите кнопку **АМПЛ** – на табло	В случае превы-

уровня выходного сигнала	замигает индикатор **АМПЛ**. Текущее значение амплитуды сигнала отображается на индикаторе с указанием размерности. Изменить значение можно способами, описанными в п.3.	шения допустимых пределов установки уровня выходного сигнала, в поле амплитуды на индикаторе загорится надпись **E03**.
5. Установка смещения постоянной составляющей выходного сигнала	Нажмите кнопку **СМЕЩ** – на табло замигает индикатор **СМЕЩ**. Текущее значение смещения отображается на индикаторе с указанием размерности. Изменить значение смещения можно прямым набором требуемой величины и нажатием кнопки **Гц/Vpp**, а также кнопками **<** и **>** и вращаемым регулятором.	В случае превышения допустимых пределов установки величины смещения выходного сигнала, на индикаторе появится надпись **E05**.
6. Включение и настройка управления прибором по последовательному интерфейсу	Нажмите кнопку **RS232** – на индикационном табло отобразится надпись **RS232 ON.** Изменение состояния (ON/OFF) производится с помощью вращаемого регулятора. Повторное нажатие кнопки **RS232** приведет к выводу меню настроек скорости связи (Baud rate). С помощью вращаемого регулятора установите значение **19200**.	
7. Установка режима амплитудной модуляции от внутреннего источника	Установите частоту несущего сигнала, его форму и уровень (пп. 2, 3, 4). Нажмите кнопку **ЧМ/АМ** – на табло в поле **МОД/ГКЧ** загорится индикатор **АМ.** Последовательным нажатием кнопок **ПРЕФ** и **МОД ВНУТР/ВНЕШ** выберите внутренний источник модуляции (индикатор **ВНЕШ** в поле **МОД/ГКЧ** не горит). Нажмите кнопку **МОД ГЕН** и установите частоту модулирую-	В случае превышения допустимых пределов установки глубины модуляции, на индикаторе появится надпись **E15**.

	щего сигнала. Установите форму модулирующего сигнала последовательным нажатием кнопок **ПРЕФ** и **ИСТОЧНИК** (форма модулирующего сигнала отображается значками в поле **МОД/ГКЧ**). Нажмите кнопку **ПАРАМ** и установите глубину модуляции в процентах (прямым набором величины и вводом размерности **Град/%**). Чтобы включить режим модуляции, нажмите кнопку **МОД ВКЛ** – в поле **МОД/ГКЧ** загорится индикатор **ВКЛ**.	
8. Установка режима частотной модуляции от внутреннего источника	Установите частоту несущего сигнала, его форму и уровень (пп. 2, 3, 4). Последовательно нажмите кнопки **ПРЕФ** и **ЧМ/АМ** – на табло в поле **МОД/ГКЧ** загорится индикатор **ЧМ**. Выберите внутренний источник модуляции, установите частоту модулирующего сигнала и его форму, руководствуясь инструкциями из п.6. Нажмите кнопку **ПАРАМ** и установите величину девиации частоты в процентах (прямым набором величины и вводом размерности **Град/%**). Чтобы включить режим модуляции, нажмите кнопку **МОД ВКЛ** – в поле **МОД/ГКЧ** загорится индикатор **ВКЛ**.	В случае превышения допустимых пределов установки величины девиации частоты, на индикаторе появится надпись **E17**.

Дистанционное управление – интерфейс RS-232

Для удаленного управления генератором GFG-3015 используется последовательный интерфейс связи RS-232 и система команд SCPI. Подключение персонального компьютера или другого управляющего устройства осуществляется через разъем DB-9, расположенный на задней панели прибора.

Подключение и отключение портов RS-232 всегда должно осуществляться при выключенном питании устройств.

Настройки скорости связи «по умолчанию» для модели генератора, используемой в данной лабораторной работе, следующие:

Скорость связи – 19200 бод;
количество бит данных – 8;
количество стоп-битов – 1;
проверка четности – нет.

Чтобы установить настройки «по умолчанию», следуйте инструкциям, приведенным в п.1 Таблицы 1 данного приложения.

Написание SCPI-команд допускается как в верхнем (прописными буквами), так и в нижнем регистре. Большинству команд соответствует укороченный вариант написания (он приведен в Таблице 2 прописными буквами). В качестве терминатора сообщения используется символ перевода строки <LF> = '\n' = 0x0A.

Таблица 2

SCPI-команды генератора

Команда	Описание	Тип пар-ра	Аргументы
`*IDN?`	Запрос сроки-идентификатора прибора		Нет
`*RST`	Установка начальных параметров прибора		
`:FUNCtion:WAVeform`	Установка формы выходного сигнала	<NR1>	<1>Синус <2>Треуг. <3>Прямоуг.
`:FUNCtion:WAVeform?`	Запрос настроек формы выходного сигнала		Нет
`:FREQuency`	Установка частоты выходного сигнала	<NRf>	Числовое значение
`:FREQuency?`	Запрос значения частоты выходного сигнала		Нет
`:AMPLitude:VOLTage`	Установка уровня выходного сигнала	<NRf>	Числовое значение
`:AMPLitude:VOLTage?`	Запрос значения уровня выходного сигнала		Нет
`:OFFSet`	Установка смещения постоянной составляющей выходного сигнала	<NRf>	Числовое значение

:OFFSet?	Запрос значения смещения постоянной составляющей выходного сигнала		Нет
:SOURce:STATe	Выбор режима модуляции	<NR1>	<0>Выкл <1>АМ <2>ЧМ <3>Свипир.
:SOURce:STATe?	Запрос настроек режима модуляции		Нет
:SOURce:SOURce	Выбор источника модуляции	<NR1>	<0>Внутр <1>Внеш
:SOURce:SOURce?	Запрос настроек источника модуляции		Нет
:SOURce:MODAM:RATe	Установка частоты модулирующего сигнала в режиме АМ	<NRf>	Числовое значение
:SOURce:MODAM:RATe?	Запрос значения частоты модулирующего сигнала в режиме АМ		Нет
:SOURce:WAVeform	Установка формы модулирующего сигнала	<NR1>	<1>Синус <2>Треуг. <3>Прямоуг.
:SOURce:WAVeform?	Запрос настроек формы модулирующего сигнала		Нет
:SOURce:MODAM:SPAN	Установка глубины модуляции (%) в режиме АМ	<NR1>	Числовое значение
:SOURce:MODAM:SPAN?	Запрос значения глубины модуляции (%) в режиме АМ		Нет
:SOURce:MODFM:RATe	Установка частоты модулирующего сигнала в режиме ЧМ	<NRf>	Числовое значение
:SOURce:MODFM:RATe?	Запрос значения частоты модулирующего сигнала в режиме ЧМ		Нет
:SOURce:MODFM:SPAN	Установка величины девиации частоты (%) в режиме ЧМ	<NR1>	Числовое значение
:SOURce:MODFM:SPAN?	Запрос значения величины девиации частоты (%) в режиме ЧМ		Нет

Приложение 2

Функции библиотеки RS-232 Library для работы с последовательным портом

Прототип	Описание	Аргументы
int **OpenComConfig** (int **portNumber**, char **deviceName[]**, long **baudRate**, int **parity**, int **dataBits**, int **stopBits**, int **inputQueueSize**, int **outputQueueSize**);	Открывает последовательный порт и устанавливает его настройки. Если порт с таким именем уже открыт, OpenComConfig закрывает его и открывает снова с заданными настройками.	**portNumber** – номер открываемого порта; относится к порту, имя которого задается в **deviceName**. **deviceName[]** – имя открываемого COM-порта. **baudRate** – скорость связи; значение по умолчанию – 9600 бод. **parity** – настройка проверки четности; значение по умолчанию 0 = без проверки. Другие значения: 1, 2, 3, 4 относятся к различным алгоритмам проверки четности. **dataBits** – количество бит данных в сообщении; значение по умолчанию 7. Допустимые значения 5, 6, 7, 8. **stopBits** – количество стопбитов в сообщении; значение по умолчанию 1. Допустимые значения 1, 2. **inputQueueSize**, **outputQueueSize** – размер входной и выходной очереди сообщений для данного порта. Если задать эти аргументы равными 0, величина очереди составит 512 байт. **Возвращаемое значение** – неотрицательная величина в случае успешного выполнения, отрицательный код ошибки в противном случае.

int **SetComTime** (int **portNumber**, double **timeout_seconds**);	Устанавливает таймаут на операции последовательного ввода/вывода для данного порта. Для операций чтения таймаут возникает при отсутствии данных во входном буфере порта в течение установленного времени. Для операций записи – когда выходной буфер заполнен, и отправка не производится.	**portNumber** – номер порта; для которого устанавливается ограничение. **timeout_seconds** – величина таймаута в секундах; значение по умолчанию 5 секунд. Если установить значение 0, то ограничение по длительности операций будет снято. **Возвращаемое значение** – неотрицательная величина в случае успешного выполнения, отрицательный код ошибки в противном случае. Ошибка возникает при обращении к неоткрытому порту, а также при указании неверных аргументов.
int **GetComStat** (int **portNumber**);	Возвращает статусную информацию для данного порта.	**portNumber** – номер порта; для которого запрашивается информация о состоянии. **Возвращаемое значение** – состоит из набора бит-флагов, каждый из которых отвечает за обозначение отдельных свойств состояния; в случае ошибки выполнения возвращается отрицательный код. Ошибка возникает при обращении к неоткрытому порту, а также при указании неверных аргументов.
int **ReturnRS232Err** (void);	Возвращает отрицательный код ошибки последней вызванной функции библиотеки или 0, если вызов последней функции прошел успешно.	

char* **GetRS232ErrorString** (int **errorNumber**);	Возвращает указатель на строку – сообщение об ошибке, код которой задан в аргументе.	**errorNumber** – отрицательный код ошибки, возвращаемый функциями библиотеки в случае неудачного выполнения.
int **ComWrt** (int **portNumber**, char **buffer[]**, size_t **count**);	Записывает заданное число байт в выходной буфер последовательного порта для отправки.	**portNumber** – номер порта; для которого производится операция. **buffer[]** – массив, содержащий данные для отправки. **count** – количество отправляемых байт. **Возвращаемое значение** – количество байт, записанных в выходной буфер в случае успешного выполнения, отрицательный код ошибки в противном случае. Ошибка возникает при обращении к неоткрытому порту, при указании неверных аргументов, а также при срабатывании таймаута отправки.
int **ComRdTerm** (int **portNumber**, char **buffer[]**, size_t **count**, int **terminationByte**);	Побайтно читает данные из входного буфера последовательного порта пока не встретит признак конца сообщения, не вычитает заданное количество байт, либо не истечет отведенный на операцию таймаут.	**portNumber** – номер порта; для которого производится операция. **buffer[]** – массив для хранения прочитанных данных. **count** – задаваемое количество байт, после считывания которых вызов функции должен быть завершен. **terminationByte** – код символа-терминатора, после обнаружения которого вызов функции должен быть завершен. **Возвращаемое значение** – количество байт, прочитанных из входного буфера (не считая терминатора) в случае

		успешного выполнения, отрицательный код ошибки в противном случае. Ошибка возникает при обращении к неоткрытому порту, при указании неверных аргументов, а также при срабатывании таймаута приема.
int **CloseCom** (int **portNumber**);	Закрывает заданный последовательный порт. Если порт не был открыт, функция не производит никаких операций.	**portNumber** – номер закрываемого порта **Возвращаемое значение** – неотрицательное число в случае успеха; в случае ошибки выполнения возвращается отрицательный код.

Некоторые функции библиотеки tsanilib

Прототип	Описание	Аргументы
int **analogInClk** (const char **source[]**, double **frequency**);	Выбор источника тактового сигнала для аналоговых входов АЦП NI PXI-6251, а также его частоты дискретизации.	**source[]** – имя источника тактового сигнала. Для того чтобы тактировать аналоговые измерения от внутреннего источника модуля NI PXI-6251, введите NULL или “OnboardClock”. **frequency** – величина частоты дискретизации (в Гц).
int **analogInTrigger** (const char **source[]**, double **level**);	Выбор источника синхроимпульсов для аналоговых входов АЦП NI PXI-6251, а также уровня срабатывания триггера. Позволяет запускать измерения, когда сигнал синхроимпульса достигнет заданного уровня.	**source[]** – имя источника синхросигнала. Введите “APFI0” чтобы запускать аналоговые измерения с приходом внешнего синхроимпульса на порт APFI0. **level** – величина уровня срабатывания триггера (в В).
int **analogRead** (int **idx**, double **warray[]**, int **size**, int* **read**);	Измерение определенного количества точек входного ана-	**idx** – номер канала АЦП. **warray[]** – массив для сохранения результатов измерений.

	логового сигнала заданным каналом АЦП NI PXI-6251.	**size** – задаваемое количество точек измерений. **read** – реально записанное количество точек после завершения измерений.