Федеральное агентство по образованию

САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

В.С.Тутыгин

ОСНОВЫ АВТОМАТИЗАЦИИ ФИЗИЧЕСКОГО ЭКСПЕРИМЕНТА

Лабораторный практикум

Санкт-Петербург
Издательство Политехнического университета
2008

УДК 001.891:681.3 (075.8)

Тутыгин В.С.Основы автоматизации физического эксперимента: Лаборат. практикум. СПб.: СПбГПУ, 2008. 96с.

Лабораторный практикум соответствует основным разделам курса дисциплины "Основы автоматизации физического эксперимента" направления бакалаврской подготовки 220200.

Изложены основные сведения по техническим и программным средствам автоматизации физического эксперимента на базе ПЭВМ IBM PC с использованием стандартных интерфейсов: PCI, ISA, USB, RS-485, CAMAC. Рассматриваются вопросы разработки программных средств систем сбора и обработки результатов экспериментов. Приводятся примеры созданных в среде САПР LabWindows/CVI 8.0 и Visual C++ программ автоматизации эксперимента.

Содержатся задания по проектированию на языке C/C++ библиотек функций для работы с аппаратурой, подключенной к компьютеру с помощью стандартных интерфейсов, программы реального времени автоматизации эксперимента, сервисной части программы, включающей средства графического пользовательского интерфейса, и математической обработки

Предназначено для студентов третьего курса специальностей "Автоматизированные системы обработки информации и управления" и "Программное обеспечение вычислительной техники и автоматизированных систем" факультета технической кибернетики.

Табл. 19. Ил. 22.

Печатается по решению редакционно-издательского совета Санкт-Петербургского государственного политехнического университета.

Содержание

Введение

Система автоматизации физического эксперимента включает в себя компьютеры и технические средства измерения и управления. Состав и исполнение технических средств измерения и управления могут существенно отличаться в зависимости от степени сложности, особенностей, пространственного расположения объекта автоматизации и условий его эксплуатации.

Компьютер подсистемы автоматизации эксперимента нижнего уровня решает задачи:

- ♦ управления аппаратурой сбора данных в реальном времени;
- ♦ формирования и выдачи управляющих воздействий на объект автоматизации;
- ♦ управления контрольно-измерительной аппаратурой и источниками эталонных сигналов;
- ♦ отображения информации о состоянии объекта автоматизации;
- ♦ формирования знаков внимания оператору при возможных критических или аварийных ситуациях.

Управление аппаратурой может заключаться в инициализации функций (например, запуска АЦП), передаче данных от компьютера функциональному элементу аппаратуры (ФЭ) или от ФЭ компьютеру. Во всех случаях взаимодействие компьютера с ФЭ (усилителем, АЦП, ЦАП и др.) происходит в форме передачи информационных посылок в цифровой форме. Обмен данными между компьютером (или, в общем случае, источником программ) и ФЭ реализуется с помощью интерфейса.

Технические характеристики автоматизированной системы измерения и управления существенно зависят от технических характеристик интерфейса таких как:

- ♦ Пропускная способность (Определяет максимально возможную скорость передачи информации с учетом затрат времени на адресацию, установление связи).
- ♦ Вместимость. (Определяет максимальное количество адресуемых устройств, которые могут быть подключены к линиям интерфейса).
- ♦ Максимальная протяженность линий интерфейса.

Для построения систем автоматизации физического эксперимента используется разнообразная аппаратура:

- Аппаратура PXI. Комплекс представляет собой блочный каркас с источником питания и шиной PCI. В блочный каркас устанавливается компьютер Pentium и функциональные модули для измерения и управления.
- Plug-in Data Acquisition Boards (DAQ boards) - встраиваемые в компьютер платы сбора данных с интерфейсом PCI, ISA или выносные модули с интерфейсом USB. DAQ boards содержат, как правило, аналоговый

мультиплексор на 4 - 64 канала, усилитель с программно устанавливаемым коэффициентом усиления, быстродействующий 12-разрядный АЦП с памятью на 16 - 512 слов, двухканальный ЦАП с памятью до 2 Кслов, 32 - 48-разрядный таймер, регистр цифрового ввода/вывода. В компьютер может быть установлено до 5 модулей DAQ boards.

- Приборы со встроенными интерфейсными узлами GPIB (General Purpose Interface Bus) Комплекс аппаратуры на базе интерфейса GPIB может иметь линейную протяженность до 25 м и содержать до 15 многофункциональных приборов.
- Аппаратура магистрально-модульных систем на основе интерфейсов VXI, CAMAC. Комплекс на базе аппаратуры CAMAC может иметь линейную протяженность до 32 км и содержать более 1200 адресуемых приборов.
- Data Acquisition Modules (Field Points, ICP CON, ADAM) - выносные модули измерения и управления (АЦП, ЦАП, модули цифрового ввода/вывода, счетчики, модули управления реле и т. д.), с интерфейсом RS-485. Характерными особенности модулей являются автономное питание от батареи или аккумулятора, гальваническая развязка с компьютером, конструктивное исполнение в герметичном корпусе, позволяющее использовать модули в системах промышленной автоматизации. Комплекс, построенный на базе модулей, может иметь линейную протяженность до 1200 м и содержать до 256 модулей в каждой ветви.
- Аппаратура Compact RIO (Reconfigurable Embedded Control and Acquisition System). Комплекс представляет собой миниатюрный блочный каркас с источником питания и шиной PCI. В блочный каркас устанавливается контроллер Pentium 200 МГц, DRAM 64Мб, Compact Flash 512 Мб и функциональные модули ввода/вывода аналоговых и цифровых сигналов. Контроллер может сопрягаться с компьютером для загрузки программы и обмена данными с помощью интерфейсов Ethernet и RS232.

Лабораторный практикум знакомит с приемами решения практических задач автоматизации эксперимента с помощью аппаратуры DAQ boards с интерфейсами PCI, ISA, USB, RS-485, аппаратуры ICP CON, CAMAC и техникой разработки программ на языках C/C++ и в среде современной системы автоматизации программирования LabWindows/CVI 8.0. Приложением к приведенному ниже описанию является базовое программное обеспечение для работы с аппаратурой и демонстрационные программы.

Работа 1. Мобильные системы сбора данных с интерфейсом USB

Целью лабораторной работы является изучение методики использования средств LabWindows/CVI 8.0 API[1] и проектирования на языке Visual C++ библиотек функций для работы с аппаратурой, подключенной к компьютеру с помощью интерфейса USB, сервисной части программ, включающей средства графического пользовательского интерфейса, и математической обработки, разработки виртуальных приборов.

Теоретические основы

Мобильные системы сбора и обработки потоков данных реализуются с помощью интерфейса USB. Необходимость в таких системах возникает, например, при использовании ноутбуков, имеющих крайне ограниченные возможности подключения внешней аппаратуры.

Программное обеспечение для работы с модулями сбора данных и управления может быть создано средствами LabWindows/CVI 8.0 API с тем, чтобы программное обеспечение затем функционировало в среде LabWindows/CVI 8.0. Это не исключает последующего переноса созданной программы в другие программные системы.

Концепция LabWindows/CVI 8.0 API (NI DAQmx API) включает универсальный подход к программированию устройств сбора данных и управления, выпускаемых фирмой National Instruments, в частности, модуля USB-6008. Главные элементы этой концепции – **каналы и задачи**. Физический канал – это терминал или контакт, на котором генерируется или принимается аналоговый или цифровой сигнал. Один физический канал может включать более, чем один, терминал, например, дифференциальный аналоговый канал или цифровой порт, содержащий 8 линий. Каждый физический канал устройства (модуля сбора данных) имеет уникальное имя, например, Dev1/port 0/line0:7, которое соответствует соглашению NI DAQmx о именах.

Виртуальные каналы это программная категория, которая инкапсулирует физические каналы со специфической информацией канала: диапазоном, терминальной конфигурацией, шкалированием, форматирующим данные. Чтобы создать виртуальный канал, нужно воспользоваться функцией Create Channel function/VI. Можно создать виртуальный канал также, используя DAQ Assistant.

Если Вы создаете виртуальные каналы с помощью DAQ Assistant, вы можете использовать их в других задачах и ссылаться на них вне контекста **задачи**. Поскольку эти каналы могут использоваться в нескольких задачах, они

[1] API – (Application Program Interface) – интерфейс прикладных программ.

называются ***<u>глобальными каналами</u>***. Можно выбрать глобальные каналы с помощью NI DAQmx API или DAQ Assistant и добавить их к **задаче**. Если вы добавляете глобальный канал к нескольким задачами модифицируете глобальный канал с помощью DAQ Assistant, изменения будут действительны во всех задачах, использующих этот глобальный канал.

ПРИМЕР.

Содержание задачи. Требуется создать средствами LabView или LabWindows/CVI виртуальный канал NI DAQmx для измерения температуры в диапазоне от 50 до 200 градусов, используя термопару J-типа, соединенную с каналом 0 модуля M-серии, конфигурируемого как Device 1.

Решение:

1. Используем функцию Create AI Thermocouple Channel function/VI.
2. Используем Dev1/ai0 как физический канал устройства, на вход которого подается сигнал с термопары.
3. Определим myThermocoupleChannel как имя, назначенное виртуальному каналу.
4. Выберем соответствующие значения для типа термопары и диапазон входного сигнала. NI DAQmx свяжет эти атрибуты с виртуальным каналом.

Программы, созданные средствами LabWindows/CVI 8.0 API могут быть использованы и в другой программной среде, например, Visual C++.

Для этого можно создать исполняемую программу и DLL, которые вызывают библиотеки LabWindows/CVI. LabWindows/CVI содержит DLL реального времени, которые включают все библиотеки. Исполняемые программы в среде LabWindows/CVI создаются также с использованием этих DLL. Директория EXLIB содержит импортируемые DLL-библиотеки и Startup-библиотеку; все они совместимы с внешними компиляторами.

Примечание. Никогда не используйте .lib файлы в \bin директории внешнего компилятора.

Всегда нужно включать в проект, создаваемый внешним компилятором, две библиотеки:

```
cvisupp.lib /* startup библиотеку */
cvirt.lib /* библиотеку импорта DLL содержащую:*/
/* User Interface Library */
/* Formatting and I/O Library */
/* RS-232 Library */
/* DDE Support Library */
/* TCP Support Library */
/* Utility Library */
```

Можно добавить статический библиотечный файл из \extlib в проект во внешнем компиляторе:

analysis.lib /* Analysis or Advanced Analysis Library */

Можно добавить файл DLL-библиотеки импорта из \extlib в проект во внешнем компиляторе:

gpib.lib/* GPIB/GPIB 488.2 Library*/
dataacq.lib/* Traditional NI-DAQ Library*/
visa.lib/* VISA Library*/
nivxi.lib/* VXI Library*/
ivi.lib/* IVI Library*/
nidaqmx.lib/* NI-DAQmx Library*/
cviauto.lib/* ActiveX Library*/
cvintwrk.lib/* Internet Library*/
cviddc.lib/* DIAdem Connectivity Library*/

Если Вы используете инструментальный драйвер, который относится к библиотекам GPIB/GPIB 488.2 and VXI, можно использовать две библиотеки: gpib.lib and nivxi.lib, чтобы разрешить ссылки на символы этих библиотек. Если Вы не имеете доступа к одному из этих файлов, вы можете заменить его одним из следующих файлов:

gpibstub.obj/* stub GPIB functions*/
vxistub.obj/* stub VXI functions*/

Если вы используете внешний компилятор, который требует точку входа WinMain, следующая опциональная библиотека позволяет определить только одну main в Вашей программе.

cviwmain.lib /* contains a WinMain() function that*/
/* calls main() */

Схема лабораторной установки

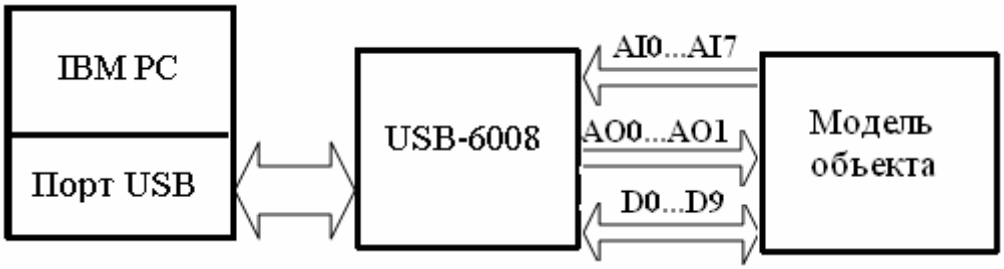


Рис. 1. AI0…AI7 – аналоговые входы, AO0…AO1 – аналоговые выходы, DO0…DO9 – цифровые входы/выходы.

Модули с интерфейсом USB (USB-6008/6009) обеспечивают многоканальный ввод и вывод аналоговых и цифровых сигналов: 10 цифровых программируемых линий ввода/вывода, 16 линий аналогового ввода, 2 линии аналогового вывода и один вход счетчика импульсов. Структура модуля приведена на рис. 2.

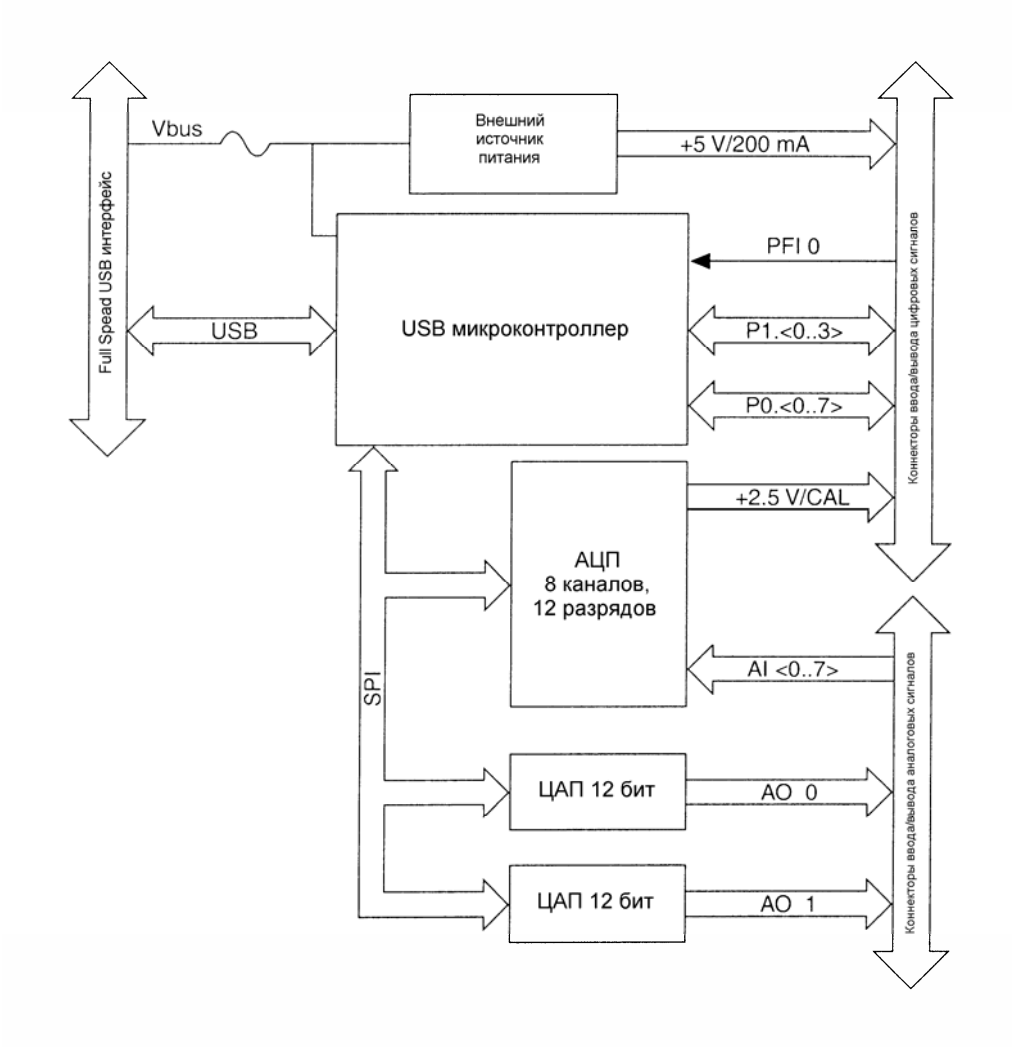


Рис. 2

Программное управление модулем USB-6008 должно включать, как минимум, следующие функции:

а) многоканального ввода аналоговых сигналов;

б) вывода аналоговых сигналов;

в) ввода/вывода цифровых сигналов;

г) управления таймером/счетчиком импульсов;

с различными способами синхронизации.

Электрические соединения модуля USB-6008 с терминальным модулем выполнены следующим образом: аналоговые входы AI0…AI7 соединены с гнездами AI0…AI7 терминального модуля, аналоговые выходы AO1...AO2 модуля USB-6008 соединены с коаксиальными разъемами AO1...AO2 терминального модуля, цифровые выходы DO0…DO7 модуля USB-6008 соединены с входами светодиодных индикаторов DO0…DO7 терминального модуля и модели объекта.

Программное обеспечение модуля USB-6008 включает библиотеку программных модулей LabWindows/CVI 8.0.

Программа работы

1. Изучить структуру и состав элементов лабораторной установки, состав и функции библиотеки программ для работы с модулем USB-6008.
2. Разработать программу многоканального ввода, отображения и обработки аналоговых сигналов.
3. Оценить экспериментально максимально достижимые характеристики скорости ввода с анализом факторов, определяющих быстродействие (характеристики компьютера, операционной системы, системы программирования, интерфейса, функциональной части модуля).
4. Разработать программу ввода и вывода цифровых данных.
5. Оценить экспериментально максимально достижимые характеристики скорости ввода и вывода цифровых данных с анализом факторов, определяющих быстродействие (характеристики компьютера, операционной системы, системы программирования, интерфейса, функциональной части модуля).

Указания к выполнению лабораторной работы

При выполнении п. 1 Программы изучить состав и функции библиотеки программ для работы с модулем USB-6008, приведенную в приложении, произвести тестирование работы модуля USB-6008 с использованием программы Test Panels. Для этого:

а) запустить MAX (Measurement and Automation Explorer) с рабочего стола Windows, затем выбрать **Devices and Interfaces>>NI DAQmx Devices**;

б) вызвать контестное меню щелчком правой кнопки мыши;

в) выбрать пункт **Test Panels** и открыть нужную панель тестирования.

Проверить с помощью панелей тестирования функционирование цифровых и аналоговых каналов ввода/вывода.

При выполнении п.1 Программы необходимо также изучить принципиальную схему и схему соединений внешних приборов (генератора стандартных сигналов и осциллогорафа), входящих в состав лабораторной установки рис. 1, обратив внимание на порядок подготовки установки к работе, порядок включения и отключения. ***Внимание! Несоблюдение правил включения и отключения может привести к выходу из строя компьютера!***

При выполнении п.2 Программы разработать программу ввода аналоговых сигналов с выхода генератора стандартных сигналов и программу вывода цифровых данных на светодиодные индикаторы, программу снятия статической и динамической модели объекта.

Программа должна создаваться в виде проекта в среде LabWindows/CVI 8.0. При выполнении п.2 Программы разработать в среде LabWindows/CVI программу просмотра данных в графической форме.

Примеры программ в среде LabWindows/CVI 8.0 можно найти, вызвав в меню **Help>>FindExamples**

Содержание отчета

1. Схема лабораторной установки.
2. Задание к работе.
3. Текст прикладной программы сбора данных выполненной в среде. LabWindows/CVI 8.0.
4. Текст программы, выполненной в среде Visual C++.
5. Графический экран разработанного пользовательского интерфейса.
6. Результаты измерения достигнутой максимальной скорости ввода/вывода цифровых и аналоговых сигналов.
7. Графический экран разработанного виртуальных приборов ввода/вывода цифровых и аналоговых сигналов.
8. Выводы.

Примечание. Тексты программ должны содержать необходимые комментарии.

Приложение 1. Примеры функций библиотеки

1. DAQmxCreateTask

Описание

Создание задачи.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxCreateTask(“ ”, &taskHandle)

Аргументы

taskHandle:

2. DAQmxClearTask

Описание

Выгрузка задачи.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxClearTask(taskHandle)

Аргументы

taskHandle:

Возвращаемый код

3. DAQmxStartTask

Описание

Запуск задачи на исполнение.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxStartTask(taskHandle)

Аргументы

taskHandle:

Возвращаемый код

4. DAQmxStopTask

Описание

Останов задачи.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxStopTask(taskHandle)

Аргументы

taskHandle:

Возвращаемый код

5. DAQmxCreateDOChan

Описание

Создание канала цифрового вывода.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxCreateDOChan(taskHandle, chan,””,DAQmx_Val_ChanForAllLines)

Аргументы

taskHandle:

chan: номер канала

DAQmx_Val_ChanForAllLines:

Возвращаемый код

Пример:

DAQmxCreateDOChan(taskHandle, chan,””,DAQmx_Val_ChanForAllLines)

DAQmxCreateDOChan(taskHandle, “Dev1/port0/line0:7” , ”” ,
DAQmx_Val_ChanForAllLines)

6. DAQmxWriteDigitalLines

Описание

Запись данных цифрового канала.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxCreateDOChan(taskHandle, 1, 1, 10.0, DAQmx_Val_GroupByChannel, data, NULL, NULL)

Аргументы

taskHandle:
chan:
DAQmx_Val_ChanForAllLines:

Возвращаемый код

Пример

DAQmxCreateDOChan(taskHandle, 1, 1, 10.0, DAQmx_Val_GroupByChannel, data, NULL, NULL)

7. DAQmxReadDigitalLines

Описание.

Чтение данных цифрового канала.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxReadDigitalLines (taskHandle, 1, 10.0, DAQmx_Val_GroupByChannel, data, 100, &read)

Аргументы

taskHandle:
chan:
DAQmx_Val_ChanForAllLines:

Возвращаемый код

Пример

DAQmxReadDigitalLines (taskHandle, 1, 10.0, DAQmx_Val_GroupByChannel, data, 100, &read)

2. DAQmxCreateAIVoltageChan

Описание

Создание задачи ввода аналогового канала.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxCreateAIVoltageChan (taskHandle, “Dev1/ai0 ” , ”” ,
DAQmx_Val_Cfg_Default, -10.0, 10.0, DAQmx_Val_Volts, NULL)

Аргументы

taskHandle:
chan:
DAQmx_Val_Cfg_Default:
DAQmx_Val_Cfg_Default

Возвращаемый код

Пример:

DAQmxCreateAIVoltageChan (taskHandle, “Dev1/ai0 ” , ”” ,
DAQmx_Val_Cfg_Default, -10.0, 10.0, DAQmx_Val_Volts, NULL)

3. DAQmxReadAnalogF64

Описание

Чтение данных аналогового канала.

Синтаксис.

ANSI C(LabWindows/CVI 8.0)

DAQmxReadAnalogF64 (taskHandle, 1000 , 10.0 ,
DAQmx_Val_GroupByChannel,data,1000,&read, NULL)

Аргументы

taskHandle:

chan:

DAQmx_Val_GroupByChannel:

Возвращаемый код

Пример:

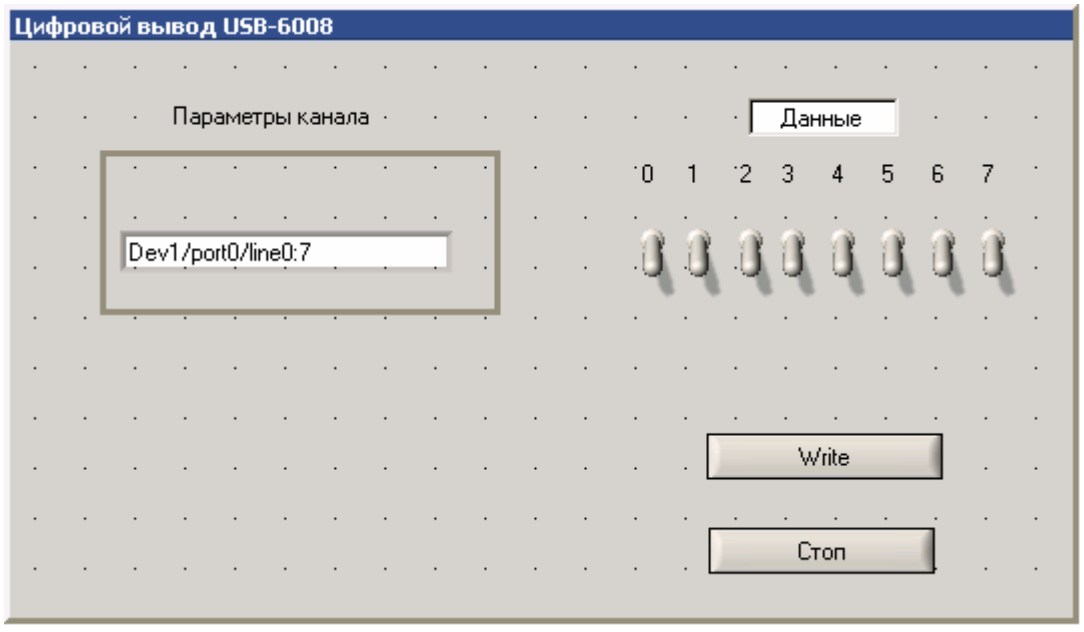
DAQmxReadAnalogF64 (taskHandle, 1000 , 10.0 ,
DAQmx_Val_GroupByChannel,data,1000,&read, NULL)

Приложение 2. Пример программы

Цифровой вывод

```
#include <NIDAQmx.h>
#include <cvirte.h>
#include <userint.h>
#include "second.h"

static int panelHandle;

int main (int argc, char *argv[])
{
	if (InitCVIRTE (0, argv, 0) == 0)
		return -1;	/* out of memory */
	if ((panelHandle = LoadPanel (0, "second.uir", PANEL)) < 0)
		return -1;
	DisplayPanel (panelHandle);
	RunUserInterface ();
	DiscardPanel (panelHandle);
	return 0;
}

int CVICALLBACK rQuit (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:
		 QuitUserInterface (0);

			break;
		}
	return 0;
}

int CVICALLBACK rWrite (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{

TaskHandle taskHandle=0;
char  chan[256];
uInt8 data[8];
/*char  errBuff[2048]={'\0'};*/
	switch (event)
		{
		case EVENT_COMMIT:
		GetCtrlVal (panel, PANEL_CHANNEL, chan);
		GetCtrlVal (panel, PANEL_SW_0, &data[0]);
```

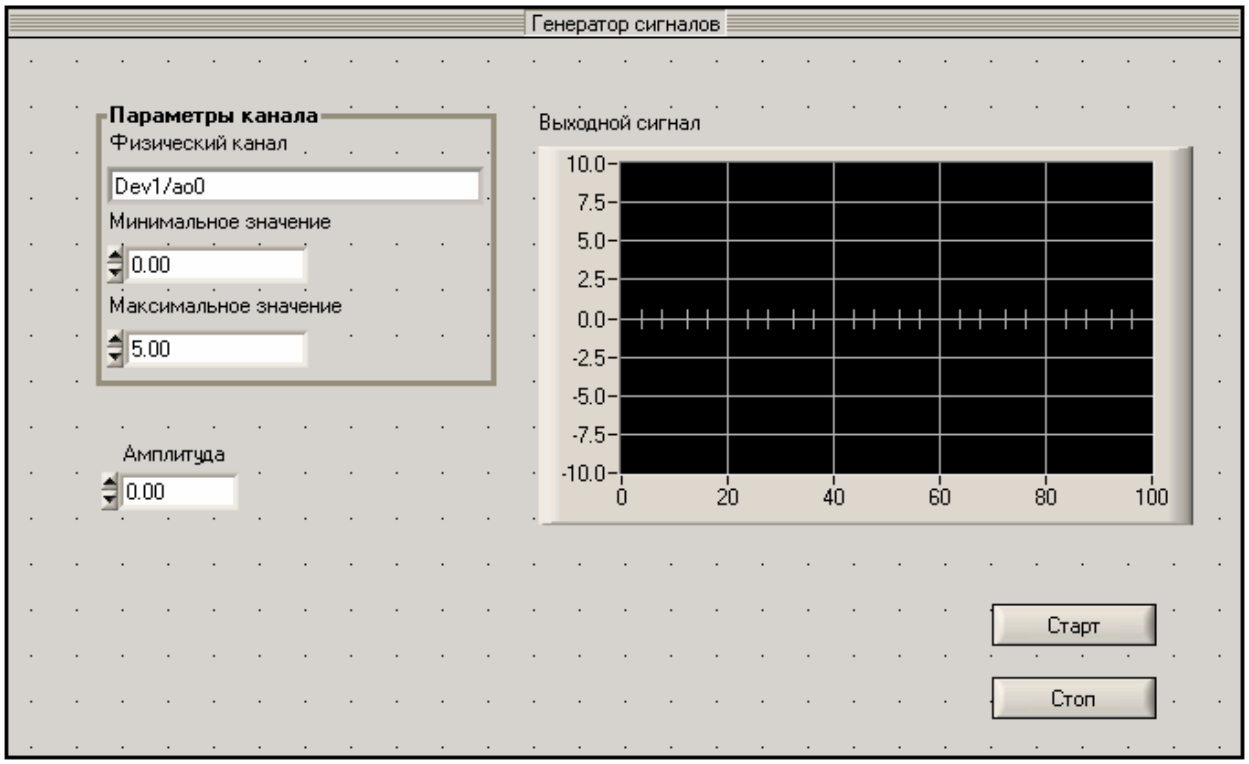
```
                GetCtrlVal (panel, PANEL_SW_1, &data[1]);
                GetCtrlVal (panel, PANEL_SW_2, &data[2]);
                GetCtrlVal (panel, PANEL_SW_3, &data[3]);
                GetCtrlVal (panel, PANEL_SW_4, &data[4]);
                GetCtrlVal (panel, PANEL_SW_5, &data[5]);
                GetCtrlVal (panel, PANEL_SW_6, &data[6]);
                GetCtrlVal (panel, PANEL_SW_7, &data[7]);

                DAQmxCreateTask ("", &taskHandle);

                DAQmxCreateDOChan (taskHandle, chan, "",
DAQmx_Val_ChanForAllLines);

                DAQmxStartTask (taskHandle);

                DAQmxWriteDigitalLines (taskHandle, 1, 1, 10.0,
DAQmx_Val_GroupByChannel, data, NULL, NULL);

                DAQmxStopTask (taskHandle);

                DAQmxClearTask (taskHandle);

                        break;
                }
        return 0;
}
```



Примеры проектов в LabWindows/CVI 8.0
Аналоговый вывод

```
#include <utility.h>
#include <NIDAQmx.h>
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "ContGen.h"

static int panelHandle;
char       chan[256];
TaskHandle  taskHandle=0;
        int        waveformType;
        double     min,max,frequency,rate,amp;
        uInt32     sampsPerCycle;
        float64    cyclesPerBuffer;
        uInt32          bufferSize;
        bool32     done=0;
        int32           written;
        int        i,rr;
        double     data[20000];

int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)
                return -1;      /* out of memory */
        if ((panelHandle = LoadPanel (0, "ContGen.uir", PANEL)) < 0)
                return -1;
        DisplayPanel (panelHandle);
        RunUserInterface ();
        DiscardPanel (panelHandle);
        return 0;
}

int CVICALLBACK StartCallback (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                GetCtrlVal(panel,PANEL_CHANNEL,chan);
                GetCtrlVal(panel,PANEL_MINVAL,&min);
                GetCtrlVal(panel,PANEL_MAXVAL,&max);
                GetCtrlVal(panel,PANEL_AMP,&amp);
                SetCtrlAttribute(panel,PANEL_GRAPH,ATTR_XAXIS_GAIN,1.0/rate);
                for (i=0;i<100;i++) { data[i]=0;}
                DAQmxCreateTask("",&taskHandle);
        DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,NU
LL);
                DAQmxStartTask(taskHandle);
```

```
                for (i=0;i<100;i++){ data[i]=i/20.0;
                DAQmxWriteAnalogScalarF64 (taskHandle, 1, 1.0, data[i], 0);
                DeleteGraphPlot(panel,PANEL_GRAPH,-1,VAL_DELAYED_DRAW);

        PlotY(panel,PANEL_GRAPH,data,100,VAL_DOUBLE,VAL_THIN_LINE,VAL_EM
PTY_SQUARE,VAL_SOLID,1,VAL_RED);
                 Delay(0.05);
                }
                SetCtrlAttribute(panel,PANEL_START,ATTR_DIMMED,1);
                ProcessDrawEvents();
                DAQmxStopTask(taskHandle);
                DAQmxClearTask(taskHandle);
          break;
                }
        return 0;
}
int CVICALLBACK StopCallback (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                QuitUserInterface (0);
                break;
                }
        return 0;
}
```



Работа 2. Высокоскоростные системы сбора данных и управления с интерфейсом ISA

Целью лабораторной работы является изучение методики проектирования на языке С библиотек функций для работы с аппаратурой, подключенной к компьютеру с помощью интерфейса ISA, создания программ реального времени автоматизации эксперимента, сервисной части программ, включающей средства графического пользовательского интерфейса, и математической обработки, разработки виртуальных приборов.

Теоретические основы

Встраиваемый в компьютер модуль с интерфейсом ISA может содержать несколько функционально законченных устройств (мультиплексор, АЦП, ЦАП, цифровой регистр ввода/вывода, таймер и др.)

Принцип программного управления аппаратурой с интерфейсом ISA

Взаимодействие компьютера с внешними устройствами (мультиплексором, АЦП, ЦАП, цифровым регистром ввода/вывода, таймером и др.) заключается в передаче команд управления и пересылке данных (от компьютера внешнему устройству и, наоборот, от внешнего устройства компьютеру). Во всех случаях взаимодействие происходит в форме пересылки данных из процессора компьютера в цифровой регистр или псевдорегистр внешнего устройства или из регистра внешнего устройства в процессор компьютера.

Передача данных производится через порты ввода/вывода (ПВВ).

Программные средства управления аппаратурой с интерфейсом ISA

Программное управление аппаратурой с интерфейсом ISA от ЭВМ может быть выполнено с использованием двух команд пересылки: записи и чтения.

Команды пересылки могут программироваться на ассемблере и на языках высокого уровня.

Ассемблер IBM PC содержит команды IN и OUT для организации обмена данными между процессором и ПВВ.

Форматы команд:

IN<аккумулятор>,<порт>

OUT<порт>,<аккумулятор>

В качестве операнда "аккумулятор" может быть использован оперативный регистр AL или AX, в качестве операнда "порт" - номер порта, имя переменной или регистр DX, в который занесен адрес порта.

Примеры команд:

IN AL,200h

```
IN AL, PORT_B
OUT 300h,AX
OUT DX,AL
```

Пример фрагмента программы передачи через порт ввода/вывода 300h кода COMMAND и последующего ввода данных в аккумулятор через порт 301h на ассемблере:

```
_asm
        {
        MOV DX,300h
        MOV AL,COMMAND
        OUT DX,AL    ;вывод данных
        MOV DX,301h
        IN AL,DX       ;ввод данных
        }
```

На языке C/C++ для ввода и вывода данных предусмотрены функции _inp, _outp (при использовании драйвера inpout32 – Inp32 и Out32).

Фрагмент программы на языке C++, рассчитанный на исполнение в среде Visual C++ , использование драйвера inpout32 и выполняющий те же функции, что и приведенный выше фрагмент программы на ассемблере, имеет вид:

```
Out32(0x300, COMMAND);
data = Inp32(0x301);
```

При создании базовых программных средств для работы с аппаратурой, подключенной к компьютеру с помощью интерфейса ISA, обычно создаются библиотеки программных модулей (процедур, функций, макрокоманд). Каждый из программных модулей обеспечивает выполнение одного конкретного действия с определенным элементом аппаратуры. Примером может быть функция включения канала мультиплексора, запуска АЦП.

Обращение к портам ввода/вывода в прикладной программе, работающей в ОС Windows 2000/XP/NT

Обращение к портам ввода/вывода в прикладной программе, работающей в ОС Windows 2000/XP/NT, возможно только при использовании специального драйвера, например GiveIO.sys. Одним из простых средств, обеспечивающих доступ к портам ввода/вывода, является программный комплекс на основе библиотеки inpout32.dll.

Комплекс включает:

- Собственно библиотеку inpout32.dll;
- Статическую библиотеку inpout32.lib, необходимую на этапе сборки программы;
- Заголовочный файл h.h с прототипами используемых функций.

Для создания проекта в Visual C++ 6.0 или Visual C++ 2005, в котором программа, написанная на языке C++, содержит обращения к портам ввода/вывода, после создания проекта нужно:

- Добавить в папку с проектом файлы h.h, inpout32.lib и inpout32.dll;
- В текст программы добавить #include “h.h”

Ниже приведен пример программы на язке Visual C++, выполняющей последовательную передачу битов в порт LPT1. Адрес 888 (в гексокоде – 0x378) представляет адрес регистра данных LPT1.

```
#include "stdafx.h"
#include "h.h"
#include <conio.h>
#include <stdlib.h>
#include <windows.h>

int main()
{
	int i = 0, Address = 888, Data = 0;

	printf("Turn off lightdiods \n");
	Out32(Address, Data);
	Sleep(1000);
	Data = 1;
	for (i = 0; i < 8; i++)
	{
		printf("Lightdiod %d\n", i+1);
		Out32(Address, Data);
		Sleep(1000);
		Data *= 2;
	}
	printf("Output!\n");
	return 0;
}
```

Структура лабораторной установки

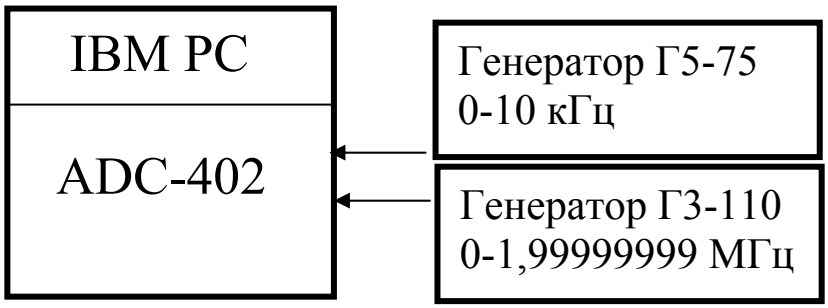


Рис. 1

Программа работы 1

1. Разработка библиотечных функций для работы с модулем ADC-402, подключенным к компьютеру с помощью интерфейса ISA.
2. Разработка программ реального времени для автоматизации измерения эхо-сигналов спектрометра, накопления и обработки.
3. Измерение времени выполнения операций ввода/вывода и основных функций разработанного виртуального прибора.
4. Разработка виртуальных приборов на базе лабораторной установки.

Указания к выполнению лабораторной работы 1

1. При выполнении п.1 программы изучить техническое описание модуля ADC-402, таблицу основных адресуемых регистров, формат данных в регистрах, приведенные в приложении, составить следующие функции для работы с модулем ADC-402:

1.1 Функция 1: set_freq
Функция устанавливает частоту отсчетов модуля ADC-402 при регистрации.
1.2. Функция 2: set_numbsamp
Функция устанавливает размер выборки (количество отсчетов в одной выборке по каждому каналу).
1.3. Функция 3: set_sinch_comp
Функция устанавливает вид синхронизации момента начала регистрации с использованием компаратора модуля ADC-402.
1.4. Функция 4: progr_start
Функция выполняет программный запуск регистрации по каналам A(C) и B(D) одновременно.
1.5. Функция 5: reset_cnt_ram
Функция выполняет программный сброс счетчика адреса ОЗУ модуля ADC-402. Функцию необходимо использовать после окончания регистрации перед чтением данных из ОЗУ модуля ADC-402.
1.6. Функция 6: reset_inhibit_start

Функция выполняет программное снятие блокировки регистрации. Блокировка регистрации в модуле ADC-402 автоматически устанавливается после начала регистрации (старта) для того, чтобы исключить возвожность повторного старта в течение цикла регистрации.

1.7. Функция 7: set_adc_sign

Функция устанавливает диапазон входного сигнала АЦП - однополярный, от 0 до +2.048В, или двухполярный, от -1.024В до +1.024В .

1.8. Функция 8: set_inp_select

Функция обеспечивает выбор источников аналоговых сигналов, подключаемого к входам АЦП . На вход каждого из АЦП: АЦП1 и АЦП2 ,- могут быть подключены один из двух источников аналогового сигнала. Для этого предусмотрены входы аналоговых сигналов А, В, С, Д. Кроме этого, с целью проведения калибровки АЦП, к их входам могут быть подключены вместо выходов источников регистрируемых сигналов выходы калибровочного ЦАП или входы АЦП могут быть подключены к аналоговой земле.

1.9. Функция 9: set_start

Функция обеспечивает выбор источника сигнала синхронизации для запуска регистрации по двум каналам одновременно.

1.10.Функция 10: get_ready

Функция обеспечивает опрос готовности модуля ADC-402 . Если возвращаемое значение равно 0 - значит модуль ADC-402 занят регистрацией, если возвращаемое функцией значение равно 1 - значит регистрация закончилась и можно выполнять чтение данных из ОЗУ ADC-402 в IBM PC.

2. При выполнении п.2 программы изучить принципиальную схему и схему соединений приборов, входящих в состав лабораторной установки и инструкцию по эксплуатации, обратив особое внимание на порядок подготовки установки к работе, порядок включения и отключения. ***Внимание! Несоблюдение правил включения и отключения может привести к выходу из строя компьютера и модуля ADC-402!*** При отладке программы автоматизации измерения эхо-сигналов спектрометра использовать в качестве имитатора эхо-сигнала спектрометра генератор сигналов низкочастотный прецизионный Г3-110, в качестве источника запускающих импульсов - генератор импульсов точной амплитуды Г5-75. Частоту синусоидального сигнала установить в диапазоне 100кГц-1,99 МГц, амплитуду - не более 1В. Частоту запускающих импульсов установить не более 1 кГц, амплитуду 3,5-4,0В. полярность - положительную.

Рекомендуемые параметры настройки генератора Г5-75: T=15,0; D=00,5; τ=0.01; K=2; U=3,5; Nchan=0. Параметры сигналов контролировать с помощью осциллографа С1-64.

Внимание! Включать питание генераторов сигналов и устанавливать параметры сигналов нужно только отключив выходы генераторов от входов модуля ADC-402. Несоблюдение этих условий может привести к выходу из строя компьютера и модуля ADC-402!

2. При выполнении п.2 программы дополнить программу регистрации эхо-сигналов спектрометра, разработанную по п. 2, включив в нее выполнение быстрого преобразование Фурье (БПФ) и отображение полученного спектра в дополнительном графическом окне на панели пользовательского интерфейса. Произвести тестирование разработанной программы, выполнив несколько измерений сигналов имитатора эхо-сигналов частот в диапазоне

частот от 100кГц до 1,99 МГц. Оценить точность автоматического определения частоты при различных значениях амплитуды измеряемого сигнала и отношения сигнал/шум.

3. Измерение времени выполнения элементарных операций ввода/вывода и фрагментов программы на языке C/C++ производить с помощью следующего программного модуля:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main( void )
{
  long    i = 600000000L;
  clock_t start, finish;
  double  duration;
  start = clock();
   for (i=0;i<1000000;i++)
 /* Здесь нужно записать текст фрагмента  программы  */
   finish = clock();
  duration = (double)(finish - start) / CLOCKS_PER_SEC;
  printf( "Duration is %f mksec\n", duration );
}
```

4. При выполнении п.3 программы разработать на базе лабораторной установки один из виртуальных приборов:

- Двухканальный цифровой компьютерный осциллограф (Рис. 2);
- Двухканальный коррелятор;
- Компьютерный анализатор спектра.

Содержание отчета

1. Схема лабораторной установки.
2. Задание к работе.
3. Тексты разработанных библиотечных модулей.
4. Текст прикладной программы регистрации и обработки данных выполненный в среде LabWindows/CVI 8.0.
5. Графический экран разработанного среде LabWindows/CVI 8.0 пользовательского интерфейса.
6. Результаты измерения времени выполнения основных операций ввода/вывода в среде Visual C++2005 и LabWindows/CVI 8.0.
7. Выводы.

Рис. 2

Приложение 1. Модуль быстродействующего двухканального АЦП с памятью ADC-402

Функциональное назначение

Модуль быстродействующего АЦП с памятью ***ADC-402*** предназначен для регистрации и ввода в IBM PC/AT -совместимый компьютер аналоговых сигналов синхронно по двум каналам.

Адресация регистров модуля ADC-402

Обмен данными между модулем ADC-402 и компьютером происходит через порты ввода/вывода. В адресном пространстве портов ввода/вывода компьютера модуль использует 9 адресов. Начальный, базовый, адрес (BASE) устанавливается переключателями на плате.

В таблицах адреса портов указаны относительно базового адреса. Абсолютный адрес портов вычисляется как сумма базового адреса и адреса из таблицы 1. Адреса портов в таблице приведены в гексокоде.

Таблица 1

Адрес	Запись	Чтение
0	Регистр данных ЦАП калибровки 2	ОЗУ канала I
1	Регистр данных ЦАП компаратора3	ОЗУ канала II
2	Регистр статуса №1	Регистр статуса №1
3	Регистр статуса №2	Регистр статуса №2
4	Не используется	Регистр статуса №3
5,6,7	Не используется	Не используется
8	ППИ, регистр данных, порт А	ППИ, регистр данных, порт А
9	ППИ, регистр данных, порт В	ППИ, регистр данных, порт В
10	ППИ, регистр данных, порт С	ППИ, регистр данных, порт С
11	ППИ, регистр управления	ППИ, запрещенный адрес для чтения
12-15	Не используется	Не используется

Ввод/вывод аналоговых сигналов

Модуль ADC-402 предоставляет возможность программно задавать режим калибровки или регистрации, устанавливать способ синхронизации при регистрации, задавать частоту дискретизации и объем выборки при регистрации.

Формат данных в регистре статуса №1. (BASE+2) Таблица 2

Бит	Запись	Чтение
0	программный старт, последовательная запись 0-1-0	контроль
1	программный сброс блокировки регистрации, последовательная запись 0-1-0	контроль
2	программный сброс счетчика адреса ОЗУ, последовательная запись 0-1-0	контроль
3	режим входа АЦП : 1 - биполярный, 0 – униполярный	контроль
4-5	выбор источника запуска , бит 0	контроль
6-7	Выбор аналогового входа	контроль

[2] Задание напряжения источника сигнала калибровки
[3] Задание уровня синхронизации

Формат данных в разрядах 6, 7 регистра статуса №1. Выбор аналогового входа.

Таблица 2.1

Бит 7	Бит 6	Назначение
0	0	аналоговый вход 0
0	1	аналоговый вход 1
1	0	контроль характеристики преобразователей
1	1	замыкание входов преобразователей на землю

Формат данных в разрядах 4, 5 регистра статуса №1.Выбор источника запуска.

Таблица 2.2

Бит 5	Бит 4	Назначение
0	0	Старт от внешнего источника синхронизации, по отрицательному фронту (спаду) входного сигнала
0	1	Программный старт
1	0	Старт от внутреннего компаратора
1	1	Старт от внешнего источника через оптронную развязку

Формат данных в регистре статуса №2. (BASE+3) Таблица 3

Бит	Запись	Чтение
0-2	Выбор частоты дискретизации	Контроль
3-5	Размер выборки	Контроль
6-7	Выбор источника синхронизации от компаратора, бит 0	Контроль

Формат данных в разрядах 0, 1, 2 регистра статуса №2.
Выбор частоты дискретизации Таблица 3.1

Бит 2	Бит 1	Бит 0	Назначение
0	0	0	40 МГц
0	0	1	20 МГц
0	1	0	10 МГц
0	1	1	5 МГц
1	0	0	2,5 МГц
1	0	1	1,25 МГц
1	1	0	0,625 МГц
1	1	1	Внешний источник частоты дискретизации

Формат данных в разрядах 5, 4, 3 регистра статуса №2.
Размер выборки. Таблица 3.2

Бит 5	Бит 4	Бит 3	Назначение
0	0	0	0,5 Кб
0	0	1	1 Кб
0	1	0	2 Кб
0	1	1	4 Кб
1	0	0	8 Кб
1	0	1	16 Кб
1	1	0	32 Кб
1	1	1	64 Кб

Формат данных в разрядах 7, 6 регистра статуса №2.
Выбор источника синхронизации от компаратора. Таблица 3.3

Бит 7	Бит 6	Назначение
0	0	Внутренняя, от канала А
1	0	Внутренняя, от канала В
0	1	Внешняя, DB-37/10, -5…+5В
1	1	Внешняя, DB-37/10, 0…+5В

Формат данных в регистре статуса №3. (BASE+4) Таблица 3.4

Бит	Чтение
0	Флаг готовности (регистрация закончилась)
1	Флаг текущего состояния (0-идет регистрация, 1-не идет регистрация)
2	Аналоговый сигнал канала I превышал установленный диапазон во время регистрации
3	Аналоговый сигнал канала II превышал установленный диапазон во время регистрации
4-7	Не используются, всегда 0

Работа 3. Системы цифрового управления с интерфейсом ISA.

Целью лабораторной работы является изучение методики проектирования на языке С библиотек функций для работы с аппаратурой, подключенной к компьютеру с помощью интерфейса ISA, создания программ реального времени, сервисной части программ, включающей средства графического пользовательского интерфейса, и математической обработки, разработки виртуальных приборов.

Теоретические основы

Системы, использующие механические перемещения с помощью шаговых двигателей, применяются очень широко. Шаговый привод применяется в координатографах, плоттерах, накопителях на магнитных дисках, в прецизионных системах лазерной резки и гравировки и т.д.

Шаговый двигатель имеет импульсное многофазное (3-6 фазное) управление. Временная диаграмма сигналов, необходимых для управления четырехфазным шаговым двигателем приведена на рис. 1

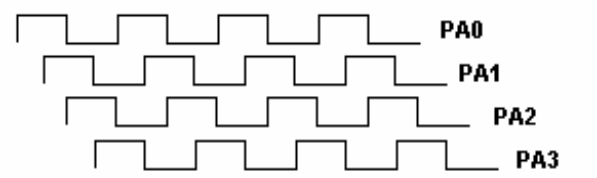


Рис. 1

Формирование многофазных сигналов необходимой мощности требует создания блока формирователей и усилителей мощности. Входным сигналом блока являются однофазная последовательность импульсов. Устройство, формирующее такую являются однофазную последовательность импульсов и потенциальных сигналов для управления перемещениями шаговых двигателей, обычно предназначено для управления тремя шаговыми двигателями (X, Y, Z). Примером может служить модуль PCL-839 фирмы Advantech. Программное управление модуля PCL-839 включает 15 функций, обеспечивающих управление тремя шаговыми двигателями в различных режимах. В частности, функции line и arc, обеспечивают выполнение линейной и круговой интерполяции при управлении двухкоординатным шаговым приводом.

Функции контроллера шаговых двигателей могут быть реализованы с помощью модуля цифрового ввода/вывода, такого как ACL-7124.

Принцип работы БИС ППИ

БИС ППИ представляет собой многорежимное устройство, содержащее три однобайтовых цифровых регистра ввода/вывода A, B и C. Эти регистры называют также портами A, B и C. Базовая микросхема БИС ППИ Intel 8255 рассчитана на максимально простое подключение к микропроцессору Intel 8088, используемому в IBM PC.

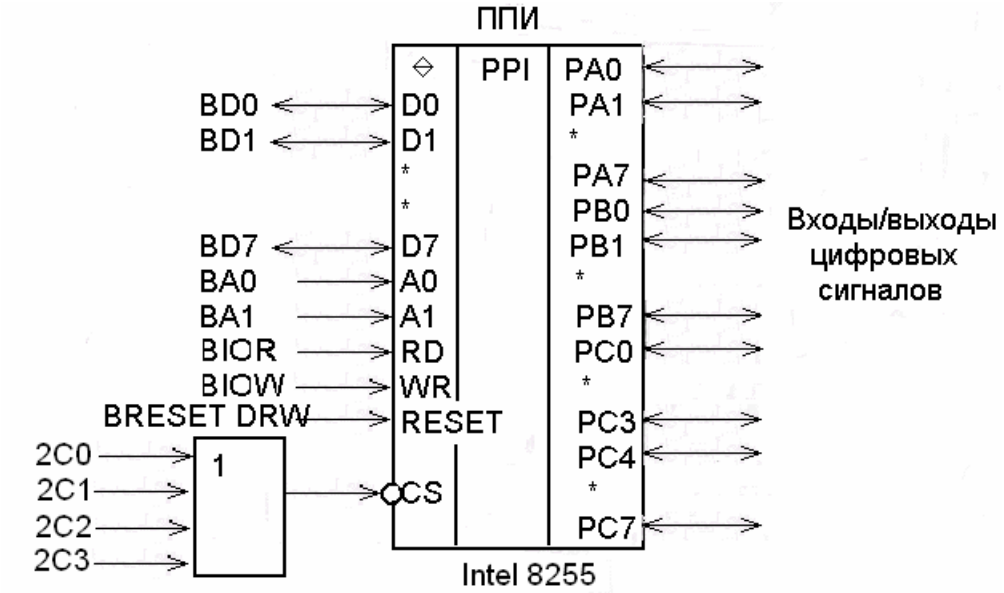


Рис. 2

БИС ППИ содержит три 8-разрядных порта ввода/вывода: A, B и C и один 8-разрядный адресуемый регистр управления.

Назначение управляющих и информационных входов:

D0…D7 - линии ввода/вывода данных (сопряжение с микропроцессором или ЭВМ);

A0,A1 - входы кода адреса регистров ППИ. 00 - регистр A, 01 - регистр B, 10 - регистр C, 11 - регистр управления;

RD - вход разрешения ввода (из ППИ в ЭВМ). Активный уровень - низкий.

WR - вход разрешения вывода (из ЭВМ в ППИ). Активный уровень - низкий.

CS - выбор кристалла. Низкий уровень на входе CS разрешает работу D0…D7. При высоком уровне на входе CS линии D0...D7 находятся в 3-м состоянии (отключено).

PA0…PA7 - линии данных порта A;

PB0…PB7 - линии данных порта B;

PC0…PA3 - линии данных порта C (нижний);

PC4…PA7 - линии данных порта C (верхний);

RESET - сброс всех регистров ППИ, установка всех портов на ввод.

Режим работы каждого из портов A, B и C, в том числе направление передачи данных, определяется управляющим словом, заносимым в регистр управления. Формат управляющего слова приведен ниже в таблице 1.

Таблица 1

Бит	Функция, режим, флаг
D0	Функция порта C нижний. 1- ввод. 0 - вывод
D1	Функция порта B. 1-ввод. 0 - вывод
D2	Режим порта B. 0 - режим 0, 1 - режим 1
D3	Функция порта C верхний. 1 - ввод, 0 - вывод
D4	Функция порта A. 1 - ввод, 0 - вывод
D6, D5	Режим порта A. 00 - режим 0, 01 - режим 1, 10 - режим 2
D7	Флаг режима. 1 - установка режимов портов A, B, C; 0 - сброс/установка разрядов порта C

Основные режимы работы БИС ППИ

- Режим 0. Ввод/вывод общего типа. Регистры A, B, C верхний, C нижний программируются независимо друг от друга на ввод или вывод. Основной режим обмена - синхронный.
- Режим 1. Стробируемый однонаправленный ввод/вывод. Основной режим обмена - асинхронный.
- Режим 2. Стробируемый двунаправленный ввод/вывод по каналу A.

Пример управляющего слова режима (режим 0, Порт A - на ввод, B - на вывод):

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	X	0	0	x

Более подробная информация о режимах работы БИС ППИ имеется в справочных руководствах.

Схема подключения БИС ППИ к линиям интерфейса ISA IBM PC приведена на рис. 2.

В лабораторной установке для адресации регистров ППИ использованы порты ввода/вывода 2C0…2C3 интерфейса ISA. К блоку светодиодной индикации и к входам контроллера шаговых двигателей X и Y подключены выводы разрядов D0…D3 порта A. Назначение битов при управлении шаговыми двигателями приведено в таблице 2.

Таблица 2

Бит	Назначение	Описание
D3	FX	Выход импульсов управления ШД X
D2	DIR_X	Выход управления направлением движения ШД X
D1	FY	Выход импульсов управления ШД Y
D0	DIR_Y	Выход управления направлением движения ШД Y

Каждый импульс управления направлением движения ШД X или ШД Y вызывает дискретный поворот оси двигателя на один шаг.

Пример программирования

Программа на языке Visual C++ для установки бита 2 порта A может иметь вид:

```
#include "stdafx.h"
#include <conio.h>

int main(int argc, char* argv[])
{
        _outp(0x2C3, 0x80); // установка режима: порты A,B,C – на вывод
        _outp(0x2C0, 2);    // установка бита 2 порта A
return 0;
}
```

Описание лабораторной установки

Лабораторная установка рис. 3 содержит компьютер IBM PC со встроенным модулем цифрового ввода/вывода ACL-7124. Выход модуля соединен с блоком индикации и входами управления контроллера шаговых двигателей ШДР-711.

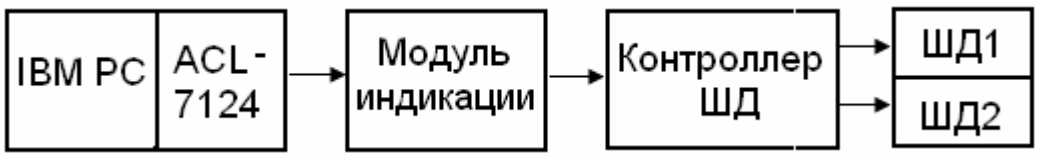


Рис. 3

Основным функциональным элементом модуля ACL-7124 является БИС параллельного программируемого интерфейса Intel 8255.

Модуль индикации отображает 4 младших бита кода порта А ППИ. Светодиод горит, если соответствующий бит установлен в "0", не горит - если в "1".

Программа лабораторной работы

1. Разработка тестовой программы ввода и вывода цифровых данных через порты A, B, C.
2. Разработка библиотеки программ для работы с модулем цифрового ввода/вывода, подключенным к компьютеру с помощью интерфейса ISA, и ориентированного для управления двухкоординатным шаговым приводом.
3. Разработка программ реального времени для управления шаговым приводом.
4. Разработка средств пользовательского интерфейса для программ управления двухкоординатным шаговым приводом.

Указания к выполнению лабораторной работы

1. При выполнении п. 1 Программы изучить техническое описание модуля ACL-7124, таблицу основных адресуемых регистров, формат данных в регистрах, приведенные в приложении, изучить тексты демонстрационных программ[4] и ее функционирование при отключенном питании контроллера ШД, составить и отладить функции побайтного вывода данных через порт А БИС ППИ модуля ACL-7124 таким образом, чтобы на выходах FX и FY генерировались сигналы с частотой 1 Гц, а на выходах DIR_X и DIR_Y – «1» или «0». Программное обеспечение создавать в среде LabWindows/CVI (ОС Windows 2000). Результаты работы программы контролировать по светодиодному блоку индикации.
2. При выполнении п.2 программы изучить описание библиотеки функций модуля PCL-839 управления шаговыми двигателями, составить и отладить функции line и arc для лабораторной установки.
3. При выполнении п. 3 Программы составить программу формирования импульсов управления двумя четырехфазными шаговыми двигателями ШДР-711 в соответствии с таблицей 2 или по заданию преподавателя.

Таблица 2.

	Относительное перемещение в шагах			
X	50	0	-50	0
Y	0	50	0	-50

[4] lab1.prj – в каталоге C:\CVIPROG\DEMO.

Проверку правильности функционирования программы производить с использованием сигнальных индикаторов модуля индикации и по стрелочным указателям на стенде шаговых двигателей..

4. При выполнении п. 3 спроектировать в среде САПР LabWindows/CVI панель средств пользовательского интерфейса для тестирования программы, разработанной по п. 3.

На панели предусмотреть:

- командные кнопки запуска приложения и выхода из программы;
- переключатель для задания периода управляющих тактовых импульсов шагового двигателя (в пределах от 0.01 до 1 сек);
- переключатели для задания количества шагов по X и Y;
- переключатели для управления направлением движения шаговых двигателей.

Содержание отчета

4. Схема лабораторной установки.
5. Задание к работе.
6. Тексты разработанных библиотечных модулей.
7. Текст прикладной программы управления шаговыми двигателями выполненный в среде LabWindows/CVI.
8. Графический экран разработанного среде LabWindows/CVI пользовательского интерфейса.
9. Выводы

Примечание. Тексты программных модулей и программ должны содержать необходимые комментарии.

Работа 4. Высокоскоростные системы обработки потоков данных с интерфейсом PCI

Целью лабораторной работы является изучение методики проектирования на языке C++ библиотек функций для работы с аппаратурой, подключенной к компьютеру с помощью интерфейса PCI, создания программ реального времени высокоскоростного сбора и обработки потоков данных, сервисной части программ, включающей средства графического пользовательского интерфейса, и математической обработки, разработки виртуальных приборов.

Теоретические основы

Наиболее высокоскоростные системы сбора и обработки потоков данных в темпе их поступления реализуются с помощью интерфейса PCI. Необходимость в таких системах возникает, например, при обработке спутниковых данных, когда скорость потока данных достигает 10 - 80 Мб/сек, а объем данных за один сеанс приема достигает несколько сотен мегабайт. Аппаратура сбора данных должна в этом случае обеспечить как минимум запись потока данных на магнитный диск компьютера, а желательно и предварительную обработку данных в темпе поступления.

Высокоскоростные системы сбора и обработки потоков данных в темпе их поступления содержат буферное ОЗУ типа FIFO (first in, first out). Содержимое буфера FIFO после его заполнения переносится в компьютер Непрерывный процесс ввода данных в реальном времени ограничен объемом буфера FIFO и скоростью переноса данных из буфера в ОЗУ компьютера. При использовании интерфейса PCI перенос данных осуществляется массивом 32-разрядных слов в режиме КПДП, но может производиться и через порты ввода/вывода. Благодаря этому скорость переноса данных достигает 10 - 80 Мб/сек. Примером модулей с интерфейсом PCI для высокоскоростного ввода в компьютер потоков данных являются модули PCI-7200 (максимальная скорость ввода - 12 Мб/сек), PCI-7300A (80 Мб/сек), модуль АЦП с памятью ЛА-н10М6PCI.

Программное управление модулем ЛА-н10М6PCI с интерфейсом PCI от компьютера должно включать, как минимум, следующие функции:

```
int InitADC(void);          //инициализация платы (задание указателя, установка базового адреса, DRQ и IRQ)
int     ADCSetting (void);      //настройка работы платы
int StopADC(void);              //прекращение работы с платой
int DataRead(void);             //чтение данных из памяти платы и вывод их на экран с одновременной записью в файл
int PrintData(void);        //вывод полученных данных на экран и запись в файл
```

Тексты функций для работы с платой ЛА-н10М6PCI приведены в приложении.

Схема лабораторной установки

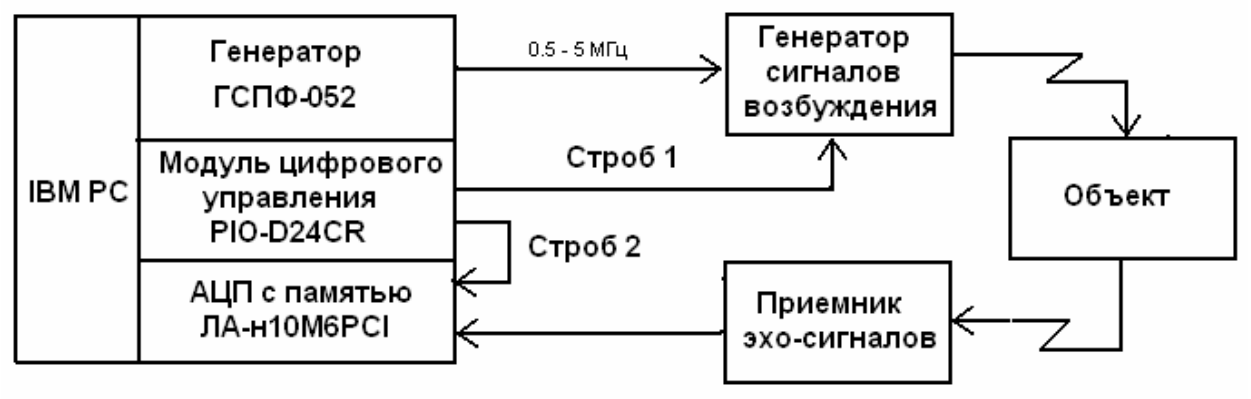


Рис. 1

Генератор ГСПФ-052 обеспечивает формирование синусоидального сигнала частоты от 0 до 10 МГц за счет прецизионного (14 двоичных разрядов) цифро-аналогового преобразования кодового образа, загружаемого компьютером. Модуль АЦП с памятью ЛА-н10М6PCI обеспечивает регистрацию аналогового сигнала в виде серии отсчетов.

Табл. 1. Технические характеристики АЦП с памятью ЛА-н10М6PCI

Число аналоговых входов	2 синхронных канала
Конфигурация аналоговых входов (не изолированы)	Однополюсные
Входное сопротивление (Импеданс)	1МОм, 30пФ
Диапазоны входного напряжения, (устанавливаются программно)	±5В; ±2,5В; ±1В; ±0,5В
Максимальное входное напряжение	±5В
Защита по напряжению аналоговых входов (питание включено)	±15В
Объем буфера памяти	256КСлов (128 КСлов на канал)
Передача данных АЦП	По чтению бита готовности или по прерыванию IRQ.
Тип АЦП	Параллельный
Разрешение	8 бит
Время преобразования	20нс
Максимальная частота дискретизации	100МГц в одноканальном режиме (канал 0).
	50МГц в двухканальном режиме.
Временное разрешение в режиме стробоскопа	1нс
Эквивалентная частота дискретизации в режиме стробоскопа	1ГГц
Запуск АЦП	От внутреннего кварцевого генератора или

	от внешней тактовой частоты

Модуль цифрового управления PIO-D56/24 обеспечивает 24 линии цифрового ввода/вывода.

Для получения эхо-сигнала необходимо обеспечить формирование радиоимпульса в генераторе сигналов возбуждения.

Типичный вид радиоимпульса и эхо-сигнала показан на рис.2.

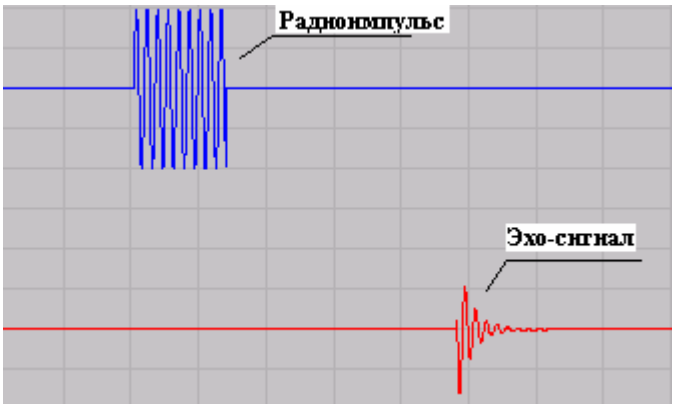


Рис. 2

Для решения данной задачи непрерывный сигнал на несущей частоте в диапазоне от 0,5 МГц до 5 МГц формируется в модуле генератора ГСПФ-052 и передается на вход генератора сигналов возбуждения. Для формирования радиоимпульса на вход управления генератора сигналов возбуждения с выхода модуля цифрового ввода/вывода PIO-D24CR передается программно формируемый строб 1 (см. рис. 2).

Функции программных средств

Программа должна включать генерацию сигнала заданной частоты в диапазоне модуля генератора ГСПФ-052 от 0,5МГц до 5 МГц , формировать с помощью модуля цифрового ввода/вывода PIO-D24CR стробы 1 и 2, регистрировать сигналы эхо с выхода приемника эхо-сигналов с помощью модуля ЛА-н10М6PCI и вычислять значение частоты дискретизированного эхо-сигнала.

Программа работы

1. Изучить структуру и состав элементов лабораторной установки, состав и функции библиотеки программ для работы с модулями ЛА_н10М6PCI, ГСПФ-052, PIO-D24CR .

2. Разработать программы формирования сигналов возбуждения с помощью модуля ГСПФ-052, регистрацию эхо-сигнала с помощью модуля ЛА_н10М6PCI и частотный анализ эхо-сигнала.
3. Оценить экспериментально максимально достижимые характеристики скорости ввода потока данных при чтении содержимого буферного ОЗУ модуля ЛА_н10М6PCI.
4. Разработать виртуальный прибор для обнаружения и идентификации веществ по эхо-сигналам.

Указания к выполнению лабораторной работы

1. При выполнении п. 1 Программы изучить состав и функции библиотек программ для работы с модулями ЛА_н10М6PCI, ГСПФ-052, PIO-D24CR, приведенную в приложении, произвести тестирование работы лабораторной установки с использованием программы DEMO1. При выполнении п.1 Программы необходимо также изучить принципиальную схему и схему соединений приборов, входящих в состав лабораторной установки рис. 1, обратив внимание на порядок подготовки установки к работе, порядок включения и отключения. ***Внимание! Несоблюдение правил включения и отключения может привести к выходу из строя компьютера и модулей!***
2. При выполнении п.2 Программы разработать программу обнаружения и идентификации веществ по эхо-сигналам.
3. При выполнении п.3 Программы измерить скорость ввода потока данных при чтении содержимого буферного ОЗУ модуля ЛА_н10М6PCI.
   Оценить масксимально достижимую скорость обнаружения искомого вещества.

4. При выполнении п.4 Программы разработать в среде LabWindows/CVI программу просмотра данных, зарегистрированных в файле после окончания регистрации, результаты частотного анализа.

Содержание отчета

1. Схема лабораторной установки.
2. Задание к работе.
3. Текст программы, выполненной в среде LabWindows/CVI 8.0 (Visual C++).
4. Графический экран разработанного пользовательского интерфейса.
5. Результаты измерения достигнутой максимальной скорости ввода потока данных при чтении содержимого буферного ОЗУ модуля ЛА_н10М6PCI и максимально достижимой скорости обнаружения искомого вещества.

6. Графический экран разработанного виртуального прибора для обнаружения и идентификации веществ по эхо-сигналам .
7. Выводы.

Работа 5. Распределенные системы удаленного сбора данных и управления с интерфейсом RS-485

Целью лабораторной работы является изучение методики проектирования на языке С библиотек функций для работы с модульной аппаратурой ICP CON, предназначенной для создания распределенных систем удаленного сбора данных и управления промышленными объектами и подключенной к компьютеру с помощью интерфейса RS-485, создания программ реального времени автоматизации управления технологическим оборудованием, сервисной части программ, включающей средства графического пользовательского интерфейса.

Теоретические основы

Особенностью работы систем сбора данных и управления промышленными объектами являются значительная протяженность объекта (до нескольких км) и неблагоприятные условия эксплуатации (высокий уровень помех, шумов, неблагоприятная внешняя среда). Для работы в таких условиях используются магистральные системы FieldPoints, ICP CON, ADAM на основе последовательного интерфейса RS-485.

ICP CON представляет семейство I-7000 модулей для удаленного сбора данных и управления. Каждый из модулей выполнен в герметичном корпусе, имеет автономное питание и представляет собой функционально законченный прибор, выполняющий определенную функцию (аналого-цифрового или цифроаналогового преобразования, ввода/вывода цифровых сигналов, таймера-счетчика, человеко-машинного интерфейса и др.). Модули управляются дистанционно по командам компьютера или модуля-микроконтроллера по двухпроводной магистрали RS-485.

I-7000 можно подразделить на следующие 10 групп по функциональному назначению:

1. Конверторы и повторители.
2. Модули цифрового ввода
3. Модули АЦП
4. Модули таймеры-счетчики
5. Модули человеко-машинного интерфейса (дисплеи).

6. Модули управления мощными реле Процессорные модули (Автономные программируемые контроллеры ветви ICP CON, ветвь может содержать до 256 модулей).
7. Модули беспроводной модемной связи
8. Модули - источники питания :

Общая структура распределенной системы измерения и управления на базе модулей ICP CON приведена на рис.1.

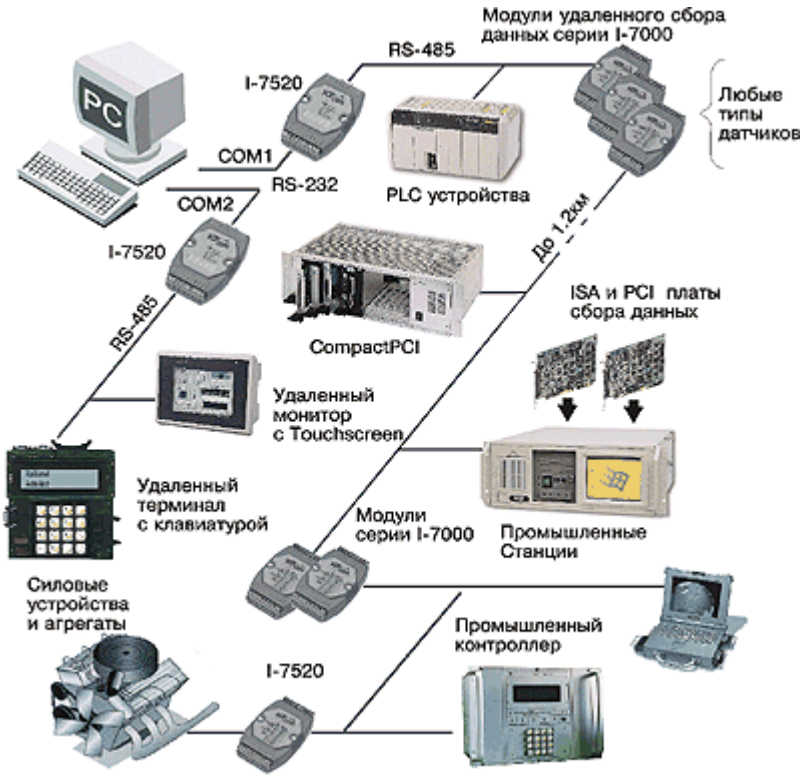


Рис. 1

ICP CON I-7000 RS-485 сеть является мощной и гибкой двухпроводной системой. Эта система поддерживает разнообразные скорости передачи и формат данных.

Примером использования распределенной системы удаленного сбора данных и управления с интерфейсом RS-485 может служить система управления станком для лазерной резки и гравировки. Автоматизация управления станком требует использования модулей цифрового ввода/вывода и модулей управления мощными реле.

Принцип программного управления модулями ICP CON и система команд

Команды управления передаются от компьютера к модулям I-7000 в виде строки символов ASCII через контроллер интерфейса RS232 (порт COM1 или COM2) компьютера.
Формат команды: ПН/Адрес/Команда/КС/ПК
ПН - признак начала посылки (Символы $, #, @, %, ~) ;
ПК - признак конца посылки (Символ CR);
КС - контрольная сумма (Два символа ASCII гексокода контрольной суммы).

Схема лабораторной установки

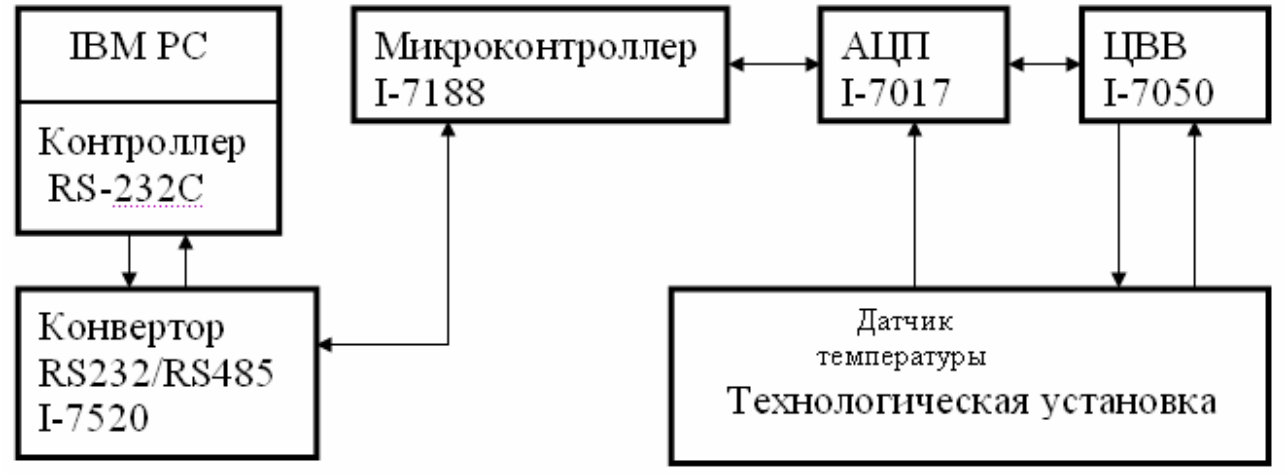


Рис. 2

В качестве лабораторной технологической установки используется станок с программным управлением. Рабочим инструментом станка является сверло, схват и пишущий элемент. В ручном или программном режиме можно производить выбор рабочего инструмента (сверло и пишущий инструмент или схват), включать и выключать схват, запускать и останавливать вращение сверла, включать и выключать перемещение рабочего инструмента в трех направлениях; по оси X (влево/вправо), по оси Y (вперед/назад) и по оси Z (вверх/вниз).

Средства автоматизации управления станком выполнены на базе модулей ICP CON I-7520 (модуля-конвертора сигналов интерфейсов RS232/RS485), I-7050 (модуля цифрового ввода/вывода), I-7017 (модуля АЦП) и I-7188 (модуля-микроконтроллера ветви ICP CON). Модуль I-7050 обеспечивает цифровое управление станком, модуль I-7017 подключен к термодатчику, установленному на корпусе электромагнита механизма схвата и используется для контроля температуры электромагнита.

Программное управление исполнительными элементами станка производится цифровым кодом модуля I-7050 в соответствии с таблицей 1.

Питание модулей ICP CON производится от автономного источника (не от компьютера!) напряжением от 12 до 30В. Использование автономного источника питания предотвращает выход из строя компьютера при аварии питания модулей ICP CON или технологической установки. Гальваническая развязка по питанию производится в модуле-конверторе сигналов интерфейса I-7520.

Таблица 1

Бит	Значение	Действие
D0	1	Вращение сверла выключено
	0	Вращение сверла включено
D1	1	Схват выключен
	0	Схват включен
D2	1	Движение по оси Z вниз
	0	Нет движения по оси Z вниз
D3	1	Движение по оси Z вверх
	0	Нет движения по оси Z вверх
D4	1	Движение по оси Y от себя
	0	Нет движения по оси Y от себя
D5	1	Движение по оси Y на себя
	0	Нет движения по оси Y на себя
D6	1	Движение по оси X влево
	0	Нет движения по оси X влево
D7	1	Движение по оси X вправо
	0	Нет движения по оси X вправо

Программное обеспечение лабораторной установки функционирует в среде Visual C++ 6.0 и включает:

а)библиотеку UART.C для работы с контроллером последовательного порта RS-232C. Функции, входящие в состав библиотеки обеспечивают инициализацию порта COM1, COM2 или COM3 и прием/передачу через порт команд управления модулями ICP CON, которые имеют формат строк символов в коде ASCII;

б)файл заголовков для библиотеки UART.C;

в)демонстрационную программу тестирования модулей ICP CON.

Для создания программы управления нужно создать в среде Visual C++ 6.0 и скомпилировать проект, включающий собственно программу на языке C++, библиотеку UART.C и файл заголовков UART.H.

Программное управление аппаратурой ICP CON с интерфейсом RS-485 от компьютера IBM PC может быть выполнено с использованием двух команд пересылки строки символов через порт RS-232C: запись и чтение.

В состав программного обеспечение аппаратуры ICP CON входят 3 функции на языке C++ :

- Инициализации COM-порта:

int OPEN_COM(int iPort, long int lBaudRate)

- записи:

int SEND_CMD(int iPort, char cCmd[], long int lTimeout, int iChksum)

- чтения:

int RECEIVE_CMD(int iPort, char cCmd[], long int lTimeout, int iChksum)

Здесь
iPort - номер используемого COM-порта;
cCmd[] - строка символов (команда ICP CON);
lTimeout - значение таймаута (времени ожидания ответного сообщения модуля ICP CON на команду. Если ответное сообщение не поступило - происходит прерывание программы с сообщением об ошибке);
iChksum - контрольная сумма для передаваемой строки символов;
iBaudRate - назначаемая скорость передачи данных по линиям интерфейса. Может устанавливаться в пределах от 9600 бод до 11520бод.

Программирование операций управление аппаратурой с интерфейсом RS-232 в ОС Windows 98

Программное управление аппаратурой с интерфейсом RS-232 от ЭВМ может быть выполнено с использованием двух команд пересылки: записи и чтения.

Команды пересылки могут программироваться на ассемблере и на языках высокого уровня.

На языке C/C++ для ввода и вывода данных предусмотрены функции _inp и _outp (в ОС Windows 98).

Фрагмент программы на языке C++, рассчитанный на исполнение в среде Visual C++ 6.0, имеет вид:

```
_out(0x300, COMMAND);
data = _inp(0x301);
```

Обращение к портам ввода/вывода в прикладной программе, работающей в ОС Windows 2000/XP/NT, возможно только при использовании специального драйвера, например GiveIO.sys.

Пример программирования (команда $01M - чтение имени модуля):

```
#define   TIMEOUT  60000L
int iComPort,iChksum;
long int lBaudRate;
char cCmd[50];
int iRet;

strcpy(cCmd,"$01M");
SEND_CMD(iComPort,cCmd,TIMEOUT,0);
printf("\nCommand =$01M, ",iComPort);
iRet=RECEIVE_CMD(iComPort, cCmd, TIMEOUT,0);
if (iRet==0) printf("Receive=%s",cCmd);
else if (iRet==1) printf("Receive =com value error (must 1/2/3/4)");
else if (iRet==2) printf("Receive =Timeout");
else if (iRet==3) printf("Receive =chksum error");
```

Программа работы

1. Изучить описание лабораторной установки и систему команд управления модулями ICP CON.
2. Изучить программу тестирования аппаратуры ICP CON и произвести тестирование модулей ICP CON, входящих в состав лабораторной установки.
3. Изучить принцип программного управления станком с программным управлением, входящий в состав лабораторной установки.
4. Разработать библиотеку элементарных функций управления станком с программным управлением (шаг по оси X и У вперед и назад, шаг по оси Z вверх и вниз, включение и выключение сверла и схвата).
5. Разработать библиотечные функции программной линейной и круговой интерполяции.
6. Разработать программы выполнения технологических операций:
   а) сверления отверстий в печатных платах;
   б) рисование графического изображения;
   в) транспортировки изделий
   по файлу чертежа, полученному в среде САПР AutoCAD.
7. Запрограммировать микроконтроллер ICP CON для автономного (без участия компьютера) управления технологическими установками при выполнении технологических операций сверления, рисования и транспортировки.
8. Выполнить действия по п.п. 2-7 в среде операционной системы Linux.

Указания к выполнению лабораторной работы

1. При выполнении п.2 программы изучить принципиальную схему и схему соединений приборов, входящих в состав лабораторной установки рис. 1 и инструкцию по эксплуатации, обратив особое внимание на порядок подготовки установки к работе, порядок включения и отключения. Подготовьте оборудование к работе в следующем порядке:

a) Включите питание станка с ЧПУ;
b) Включите режим работы станка РУЧНОЙ;
c) Протестируйте управление станком в ручном режиме ;
d) Включите блок питания модулей ICP CON;
e) Выполните программу adam1.exe начальной инициализации модулей ICP CON;
f) Включите режим работы станка АВТ;
g) Запустите демонстрационную программу demo1.exe.

Внимание! После отработки программы demo1.exe включите режим работы станка «РУЧНОЙ». Несоблюдение правил включения и отключения может привести к выходу из строя компьютера и модулей ICP CON!

2. При выполнении п. 2 Программы изучить систему команд управления модулями ICP CON I-7000, исходный текст на языке C++ демонстрационной программы DEMO1.C. Проанализировать результат выполнения функции 3 в программе DEMO1.

При выполнении п. 4 Программы, используя данные таблиц 1 и 2, составить программу выполнения элементарных функций управления станком с программным управлением (шаг по оси X и У вперед и назад, шаг по оси Z вверх и вниз, включение и выключение сверла и схвата. Среда разработки программы – Visual C++ 6.0 (Windows 98/2000).

Внимание! Передаваемый модулю I-7050 цифровой код необходимо брать из следующих возможных:

Таблица 2.

Значение кода В гексокоде	Действие
83h	X вправо, схват выкл., сверло выкл.
43h	X влево, схват выкл., сверло выкл.
23h	Y на себя, схват выкл., сверло выкл.
13h	Y от себя, схват выкл., сверло выкл.
0Bh	Z вверх, схват выкл., сверло выкл.
7h	Z вниз, схват выкл., сверло выкл.
1h	схват включен, сверло выкл.
2h	Сверло вкл., схват выкл.

Ограничения возможных кодов необходимы, чтобы исключить возможность включения схвата на длительное время (более 10 сек.) и его выход из строя из-за перегрева и программирования невыполнимых действий, таких как X вправо и X влево одновременно, что также может привести к выходу из строя аппаратуры управления двигателями станка.
Величину элементарного шага по оси X, Y и Z подобрать минимально возможной с помощью временной задержки.
Ниже приведен пример фрагмента программы шага по X вправо.

```
strcpy(cCmd,"@0183"); //X вправо, схват выкл., сверло выкл.
printf("\nCommand =@0183, ",iComPort);
SEND_CMD(iComPort,cCmd,TIMEOUT,0);
iRet=RECEIVE_CMD(iComPort, cCmd, TIMEOUT,0);
if (iRet==0) printf("Receive=%s",cCmd);
else if (iRet==1) printf("Receive =com value error (must 1/2/3/4)");
else if (iRet==2) printf("Receive =Timeout");
else if (iRet==3) printf("Receive =chksum error");
ourdelay(3000); //задержка, в течение которой действительна предыдущая команда
strcpy(cCmd,"@0103"); //Стоп, схват выкл., сверло выкл.
printf("\nCommand=@0103, ",iComPort);
SEND_CMD(iComPort,cCmd,TIMEOUT,0);
iRet=RECEIVE_CMD(iComPort, cCmd, TIMEOUT,0);
if (iRet==0) printf("Receive =%s",cCmd);
else if (iRet==1) printf("Receive =com value error (must 1/2/3/4)");
else if (iRet==2) printf("Receive =Timeout");
else if (iRet==3) printf("Receive =chksum error");
```

3. При выполнении п. 5 Программы составить и отладить программные модули линейного перемещения рабочего инструмента в плоскости X-Y из одной точки с заданными координатами в другую. Использовать принцип программной интерполяции с помощью оценочной функции:

$$P = (Y_c-Y_a)*(X_b-X_a) - (Y_b-Y_a)*(X_c-X_a)$$

где X_a, Y_a - координаты исходной точки;
X_b, Y_b - координаты конечной точки;
X_c, Y_c - координаты текущей точки.

Перемещение рабочего инструмента в плоскости X-Y из одной точки с заданными координатами в другую при этом производится элементарными единичными шагами по X или Y. Оценочная функция вычисляется после каждого шага в текущей точке по формулам:
$P_{i+1} = P_i - (Y_b-Y_a)$ если был сделан шаг по X;
$P_{i+1} = P_i - (X_b-X_a)$ если был сделан шаг по Y.

Если значение оценочной функции >0, то следующий шаг делается по X, если <0, - то по Y. Подробное описание алгоритма линейной интерполяции с помощью оценочной функции, пример расчета и рабочая программа приведены в приложении.

5. При выполнении п.6 Программы разработать программы выполнения технологических операций:

а) сверления отверстий в печатных платах по чертежу, заданному преподавателем;

б) рисования (гравировки) графического изображения (геометрической фигуры или товарного знака) по графическому файлу <>.plt в формате HPGL.

Пример содержимого части графического файла tvs.plt в формате HPGL, содержащее только строки управления (признаком является наличие в заголовке строк PU (pen up - поднять перо) и PD (pen down - опустить перо))

PU-3157 4229;
PD3803 4229;
PD3803 -4011;
PD-3157 -4011;
PD-3157 4229;
SP0;

Синтаксис строки:
PU (PD) <Координата X> <Координата Y>

Файл содержит описание квадрата с координатами диагональных углов (-3157, 4229), (3803, -4011).

Использовать в качестве прототипа программу лазерной гравировки товарного знака Optcorel.c, имеющуюся рабочем каталоге C:/TC/OAFE.

в) транспортировки изделий из одной точки рабочего стола станка в другую по заданию преподавателя.

10. При выполнении п.7 Программы занести программу выполнения технологической операции по п. 5 в микроконтроллер I-7088 и организовать автономное ее выполнение под управлением микроконтроллера без участия компьютера.

Содержание отчета

1. Схема лабораторной установки.
2. Задание к работе.
3. Тексты разработанных библиотечных функций управления станком с программным управлением (шаг по оси X и У вперед и назад, шаг по оси Z вверх и вниз, включение и выключение сверла и схвата)
4. Тексты разработанных библиотечных функции программной линейной и круговой интерполяции.
5. Тексты разработанных программ выполнения технологических операций:

а) сверления отверстий в печатных платах;
б) рисование графического изображения;
в) транспортировки изделий

6. Графический экран разработанного пользовательского интерфейса.

Примечание. Тексты программных модулей и программ должны содержать необходимые комментарии.

7. Выводы.

Приложение 1. Примеры команд управления модулями ICP CON

1. $AA2

Описание: Чтение конфигурации модуля

Синтаксис: $AA2[KC](CR)
$ - признак начала команды;
AA - адрес модуля, от 00h до FFh.

Ответное сообщение:
а) !AATTCCFF[KC](CR) -если команда правильная;
б) ?AA[KC](CR) - если команда неправильная;
в)нет сообщения, если в команде есть синтаксическая ошибка или произошла ошибка при передаче команды.
где
! - ограничитель (стартовый символ) отклика на правильную команду;
? - ограничитель (стартовый символ) отклика на неправильную команду;
AA - адрес модуля (от 00h до FFh);
TT - код типа модуля (должен быть равен 40h);
CC - код скорости передачи команд

CC	03	04	05	06	07	08	09	0A
Скорость, бод	1200	2400	4800	9600	19200	38400	57600	115200

FF - формат данных модуля

D7	D6	D5	D4	D3	D2	D1	D0
1*	2*	0	0	0	3*		

1* - условие срабатывания счетчиков: 0 - по заднему фронту; 1 - по переднему фронту.
2* - байт контрольной суммы: 0 - запрещен; 1 - разрешен.
3* - тип модуля: 000 - 7050; 001 - 7060; 010 - 7052; 011 - 7053.

Пример

Команда: $012 Отклик: !01400600

2. $AAM

Описание: Чтение имени модуля

Синтаксис: $AAM[KC](CR)
$ - признак начала команды;
AA - адрес модуля, от 00h до FFh;
M - команда чтения имени модуля.

Ответное сообщение:
а) !AA(Data)[KC](CR) -если команда правильная;
б) ?AA[KC](CR) - если команда неправильная;
в)нет сообщения, если в команде есть синтаксическая ошибка или произошла ошибка при передаче команды.
где
! - ограничитель (стартовый символ) отклика на правильную команду;
? - ограничитель (стартовый символ) отклика на неправильную команду;
AA - адрес модуля (от 00h до FFh);
Data - имя модуля.

Пример

Команда: %01M Отклик: !017050
Чтение имени модуля по адресу 01, возвращаемое имя модуля – 7050

3. $AAF

Описание: Чтение фирменной модели модуля

Синтаксис: $AAF[KC](CR)
$ - признак начала команды;
AA - адрес модуля, от 00h до FFh;
F - команда чтения фирменной модели модуля.

Ответное сообщение:
а) !AA(Data)[KC](CR) -если команда правильная;
б) ?AA[KC](CR) - если команда неправильная;
в)нет сообщения, если в команде есть синтаксическая ошибка или произошла ошибка при передаче команды.
где

! - ограничитель (стартовый символ) отклика на правильную команду;
? - ограничитель (стартовый символ) отклика на неправильную команду;
AA - адрес модуля (от 00h до FFh);
Data - имя фирменной модели модуля.

Пример

Команда: $01F Отклик: !01A1.3
Чтение имени модуля по адресу 01, возвращаемое имя фирменной модели модуля - A1.3.

4. @AA(Data)

Описание: Установка цифрового выхода.

Синтаксис: @AA(Data)[KC](CR)
@ - признак начала команды;
AA - адрес модуля, от 00h до FFh;
Data - выводимое значение (от одного до четырех символов в зависимости от типа модуля, для модуля 7050 - два символа).

Ответное сообщение:
а) >[KC](CR) -если команда правильная;
б) ?[KC](CR) - если команда неправильная;
в) ![KC](CR) -если команда проигнорирована;
в)нет сообщения, если в команде есть синтаксическая ошибка или произошла ошибка при передаче команды.
где
> - ограничитель (стартовый символ) отклика на правильную команду;
? - ограничитель (стартовый символ) отклика на неправильную команду;
! - ограничитель (стартовый символ) отклика на проигнорированную команду. Является признаком того, что модуль находится в режиме таймаута и его выход установлен в состояние SafeValue;

Пример

Команда: @0200 Отклик: >
Вывод значения 00 в модуль по адресу 02, возвращаемый символ ">"- признак успешности выполнения операции.

5. %AANNTTCCFF

Описание: Задание конфигурации модуля I-7017.

Синтаксис: %AANNTTCCFF [KC](CR)
% - признак начала команды;
AA - адрес модуля, от 00h до FFh;
NN – новый адрес модуля, от 00h до FFh;
TT – диапазон измеряемого сигнала:
08 - +/- 10 В; 09 - +/- 5В; 0A - +/- 1В; 0B - +/- 500мВ; 0C - +/-150 мВ; 0D - +/- 20 мА.
CC - скорость обмена:
03 – 1200 бод, 04 – 2400; 05 – 4800; 06 – 9600; 07 – 19200; 08 – 38400; 09 – 57600; 0A – 115200.
FF – формат данных:
D7: 0=50Гц, 1=60Гц.
D6: 0=контрольной суммы нет, 1= контрольная сумма есть.
D5: 0-нормальная передача; 1-быстрая передача.
D4, D3, D2 – не используются, равны 0.
D1,D0: 00-формат данных Engineering (со знаком и десятичной точкой); 01 – формат процентный (от максимального значения); 10- формат HEX .
Ответное сообщение:
а) ! AA [KC](CR) -если команда правильная;
б) ?AA [KC](CR) - если команда неправильная;
где
! - ограничитель (стартовый символ) отклика на правильную команду;
? - ограничитель (стартовый символ) отклика на неправильную команду;
AA – адрес модуля (от 00 доFF);

6. #AAN

Описание: Чтение аналогового входа канала N модуля I-7017.

Синтаксис: #AAN[KC](CR)
- признак начала команды;
AA - адрес модуля, от 00h до FFh;
N – номер канала, от 0 до 7.

Ответное сообщение:
а) > (Data) [KC](CR) -если команда правильная;
б) ?AA [KC](CR) - если команда неправильная;
в)нет сообщения, если в команде есть синтаксическая ошибка или произошла ошибка при передаче команды,
где
> - ограничитель (стартовый символ) отклика на правильную команду;
? - ограничитель (стартовый символ) отклика на неправильную команду;
AA – адрес модуля (от 00 доFF);
(Data) – значение аналогового сигнала.

Пример

Команда: #032 Отклик: >+02.513

Чтение результата измерения по адресу 03, канал 02. Возвращаемый символ ">"- признак успешности выполнения операции, +02.513 – значение измеренного сигнала в вольтах.

Команда: #029 Отклик: ?02

Чтение результата измерения по адресу 03, канал 09. Возвращаемый символ ?02 – признак того, что модулем 02 команда не выполнена.

Работа 6. Распределенные системы удаленного сбора данных и управления с интерфейсом CAMAC

Целью лабораторной работы является изучение методики проектирования на языке С библиотек функций для работы с модульной аппаратурой CAMAC, предназначенной для создания распределенных систем удаленного сбора данных и управления промышленными объектами, создания программ реального времени автоматизации измерения и управления, сервисной части программ, включающей средства графического пользовательского интерфейса.

Теоретические основы

Интерфейс CAMAC (Computer Application for Measurement And Control) используется при создании сложных (до 1200 приборов) и/или значительно территориально распределенных (до 30 км) систем измерения и управления. Интерфейс CAMAC был разработан для автоматизации наиболее сложных экспериментов в ядерной физике.

Основной стандартизованой частью системы CAMAC является унифицированный блочный каркас (крейт) в который устанавливается до 23 функциональных модулей (ФМ) и контроллер крейта (КК). Функциональный модуль CAMAC представляет собой законченный прибор, имеющий определенное функциональное назначение.

Основные группы ФМ:

1. Модули ввода/вывода аналоговых сигналов (усилители с программируемым коэффициентом усиления, аналоговые мультиплексоры, АЦП, ЦАП).
2. Модули ввода/вывода цифровых сигналов (цифровые регистры ввода/вывода, счетчики импульсов).
3. Модули временной синхронизации (генераторы импульсов, таймеры, модули времени, цифровые часы).
4. Модули-контроллеры исполнительных устройств (электродвигателей, шаговых двигателей, реле).

Количество крейтов в комплексе CAMAC может быть от 1 до 62. Максимальная территориальная удаленность крейта от ЭВМ составляет 500м * N, где N - количество крейтов в комплексе.

Принцип программного управления ФМ CAMAC

ФМ в крейте полностью подчинены командам КК и, как правило, не имеют ручного управления.

КК управляет ФМ с помощью адресуемых команд, имеющих структуру NAF[W], где

N - номер станции в крейте, в которую установлен ФМ. (N=1..23).
A - субадрес, адрес устройства внутри ФМ, (A = 0..15).
F - функция, (F=0..31).
W - слово данных (W1..W24)

и безадресных команд

Z - общий сброс,
C - селективный сброс.

ФМ является источником сигналов:

R - слово данных (R1..R24).

X - команда принята. Вырабатывается функциональным модулем как подтверждение расшифровки модулем адресованной ему команды. X=0 может означать, что:

- адресуемый модуль в крейте отсутствует;
- адресуемый модуль неисправен;
- адресованная модулю команда не может быть выполнена, так как она отсутствует в списке команд.

Q - сигнал готовности. Вырабатывается в ответ на команду "Проверка Q" как признак окончания операции в ФМ или готовности модуля к циклу обмена данными с контроллером крейта.

L - запрос на обслуживание. Инициативный сигнал ФМ. Используется при организации обмена по прерыванию.

Каждый ФМ CAMAC имеет определенный набор команд управления.

Программные средства управления аппаратурой CAMAC.

Программное управление аппаратурой CAMAC от ЭВМ реализуется с помощью библиотеки программных модулей (процедур, функций, макрокоманд. Набор функций на языке С, предоставляющие возможности программисту написания наглядных и эффективных программ, и примеры программ управления CAMAC - аппаратурой приведены в приложении.

Контроллер крейта CAMAC типа ККМ содержит 10 программно адресуемых регистров и рассчитан на управление от интерфейсной карты ISA.

Обращение к регистрам контроллера ККМ производится через так называемые порты ввода вывода. Адреса регистров контроллера и формат данных в регистрах приведены в таблице 1

Таблица 1

Номер регистра	Адрес ПВВ	Формат данных	Функция
0	3A0	W24…..W17	Старший байт записываемых данных
1	3A1	W16…W9	Средний байт
2	3A2	W8…W1	Младший байт
3	3A3	x x x x A8 A4 A2 A1	Субадрес ФМ
4	3A4	x x x F16 F8 F4 F2 F1	Номер функции F
5	3A5	x x x N16 N8 N4 N2 N1	Номер станции N
6	3A6	x x x x x x C Z	Функции C, Z
7	3A7	x x x x x x x x	Запуск генератора цикла CAMAC
8	3A8	L L16 L8 L4 L2 L1 X Q	LAM, X, Q
9	3A9	R24…R17	Старший байт считываемых данных
A	3AA	R16…R9	Средний байт
B	3AB	R8…R1	Младший байт
F	3AF	x x NC NC x x x x	Номер крейта

При написании оптимальных по быстродействию программ управления аппаратурой CAMAC целесообразно использовать в необходимых случаях программирование на языке C или на Ассемблере. Кроме того большее быстродействие может быть достигнуто, если использовать для программирования операций управления аппаратурой CAMAC не библиотеки процедур или функций, а программировать CAMAC-операции на физическом уровне, используя адреса регистров контроллера крейта и порты ввода-вывода, через которые производится обращение к ним.

Для программирования CAMAC - операций на физическом уровне необходимо производить пересылки необходимых данных через порты ввода-вывода 3A0-3AF в регистры контроллера крейта.

Базовый адрес платы сопряжения с контроллером крейта ККМ устанавливается равным 3A0h по умолчанию, но может быть изменен пользователем.

Обращение к портам ввода/вывода в прикладной программе, работающей в ОС Windows 2000

Обращение к портам ввода/вывода в прикладной программе, работающей в ОС Windows 2000, возможно только при использовании специального драйвера, например GiveIO.sys. Одним из простых средств, обеспечивающих доступ к портам ввода/вывода является программный комплекс на основе драйвера totalio.sys.

Файл драйвера totalio.sys должен быть скопирован в папку C:\WINNT\System32\drivers. В системный реестр должны быть внесены следующие дополнения:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\totalio]
"ErrorControl"=dword:00000001
"Type"=dword:00000001
"Start"=dword:00000002
"ImagePath"="system32\\DRIVERS\\totalio.sys"
```

При создания проекта в Visual C++ 6.0, в котором программа, написанная на языке C++, содержит обращения к портам ввода/вывода нужно использовать операторы _inp и _outp.

Пример программирования операции NA(0)F(25):

на языке C++:

```
_outp(0x3A5,N);
_outp(0x3A3,0);
_outp(0x3A4,25);
_outp(0x3A7,0);
```

на Ассемблере:

```
MOV DX,3A5h;  MOV AL,N;    OUT DX,AL
MOV DX,3A3h;  MOV AL,0;    OUT DX,AL
MOV DX,3A4h;  MOV AL,25;   OUT DX,AL
MOV DX,3A7h;  MOV AL,0;    OUT DX,AL
```

Программирование операций управления аппаратурой и передачи данных на ассемблере целесообразно использовать в следующих случаях:

- при написании процедур (функций) библиотеки (аналогичной CAMACLIB);
- для достижения большей скорости выполнения CAMAC - операций в критических с точки зрения быстродействия участках программы, написанной на языке высокого уровня. Для этого фрагмент критической части программы выполняется в виде ассемблерной вставки.

Программирование основных видов обмена данными

Основными видами взаимодействия ЭВМ с аппаратурой САМАС являются передача команд управления от ЭВМ к ФМ и обмен данными. Команды и данные могут передаваться в режимах синхронного, асинхронного обмена, обмена по КПДП и по прерыванию.

Схема лабораторной установки

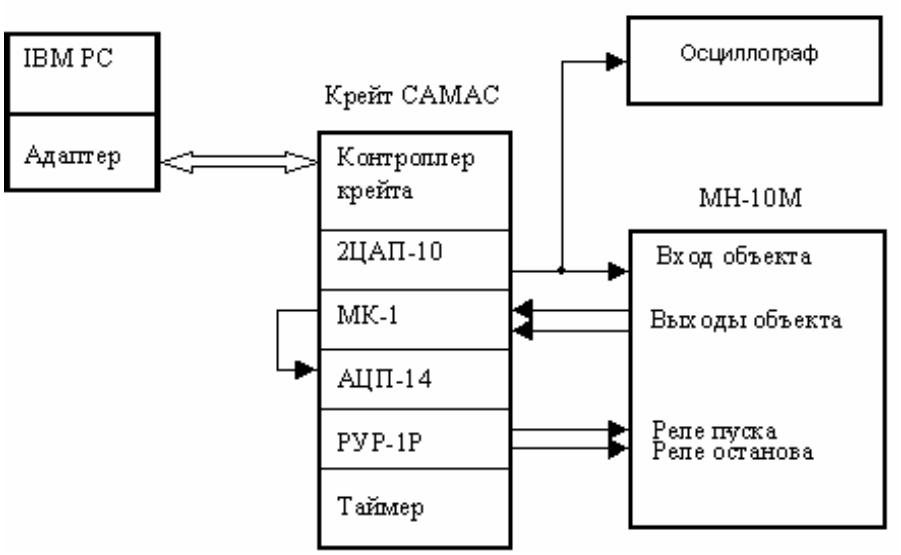


Рис. 1

Программа работы

1. Программирование элементарных функций управления и ввода/вывода.

1. Составить и отладить программы выполнения элементарных операций управления аппаратурой САМАС:

- Z;
- Передача кода 1000 в оба ЦАП, модуль 2ЦАП-10;
- Запуск АЦП, модуль АЦП-14;
- Чтение результата преобразования АЦП, модуль АЦП-14;
- Запись кода включенного канала, модуль РУР-1Р.

2. Составить с учетом имеющейся таблицы соединений аппаратуры САМАС с объектом и отладить программы выполнения элементарных операций управления объектом (МН-10М).

- Включение реле пуска и реле останова;
- Передача аналогового сигнала на вход объекта;
- Ввод в ЭВМ аналогового сигнала с выхода объекта

3. Измерить с помощью системного таймера время выполнения основных САМАС-операций.

Таблицу результатов измерения времени выполнения CAMAC-операций составить в форме:

Операция	Система программирования		
	Ассемблер	ANSI C LabWindows/CVI	Visual C++
Запись в порт			
Чтение из порта			
Zero, Clear			
Act			
ActW2B			
ActR2B			

4. Составить программу генерации тестовых сигналов с помощью модуля 2ЦАП-10:

- пилообразного;
- гармонического;
- импульсного.

Контролировать тестовые сигналы с помощью осциллографа.

2. Программирование в реальном времени.

1. Изучить основные способы привязки процессов управления аппаратурой и обработки данных в программах автоматизации эксперимента к реальному времени.
2. Освоить технику измерений точности временной привязки процессов управления аппаратурой и обработки данных в программах автоматизации эксперимента к реальному времени.
3. Освоить приемы оптимизации программ реального времени по критерию максимально возможной точности временной привязки .

Указания к выполнению

1. При выполнении п.1 задания изучить приведенные примеры временной синхронизации процессов управления аппаратурой и обработки данных в программах автоматизации эксперимента и составить программы по одному из вариантов задания:

1.1. В программе реализовать:

1) регистрацию с помощью модуля АЦП-14 значений аналогового сигнала с шагом дискретности по времени - 10 мсек;

2) формирование и вывод управляющего воздействия Y=1+Sin(t) с шагом дискретности по времени 100 мксек.

1.2. Использовать один из способов привязки к реальному времени:
   1) с программируемыми временными задержками;
   2) с ожиданием готовности источника синхронизации ;
   3) с прерыванием от источника синхронизации;
   4) с управлением от источника синхронизации.

1.3. В качестве источника синхронизации использовать;
   1) Таймер цифровой;
   2) Генератор импульсов "Clock Generator";
   3) Модуль времени МВ-1.

Примечание: выход генератора импульсов или модуля времени МВ-1 соединить с входом L модуля 2ЦАП-10, модуль 2ЦАП-10 использовать как источник сигнала готовности и запроса на прерывание.
Использовать ФМ САМАС: АЦП-14, Таймер цифровой, Clock Generator, МВ-1, 2ЦАП-10.

2. При выполнении п.2 задания изучить приведенные примеры (файлы cam_in3, cam_in4, cam_in5, cam_in6) измерения шага временной синхронизации с помощью системного таймера; используя системный таймер IBM PC, измерить фактически полученный шаг дискретности измерений во времени в программе по п.1 и вычислить погрешность измерения временной привязки.
3. При выполнении п.3 задания оптимизировать программы по п.2, используя в необходимых случаях программирование операций управления аппаратурой САМАС на физическом уровне на языке C++ и Ассемблер, с целью уменьшения погрешности временной привязки. Измерить фактически полученный шаг дискретности измерений во времени после оптимизации программы и измерить достигнутую погрешность временной привязки.

Содержание отчета

1. Схема лабораторной установки.
2. Задание к работе.
3. Тексты разработанных программ с комментариями.
4. Таблицы результатов измерений временных характеристик.
5. Текст программы автоматизации эксперимента.
6. Вид панели пользовательского интерфейса.
7. Результаты измерений и обработки данных эксперимента (в графической форме).
8. Выводы.

Приложение 1

Система команд управления модулями САМАС.

АЦП-14

(Диапазон входных сигналов +/- 7В, 13 разрядов, 14-й разряд - знаковый, масштаб преобразования - 1 ед. кода/мВ, время преобразования - 2 мс)

F(25)A(0) - пуск АЦП. L=1 после окончания преобразования.
F(0)A(0) - чтение результатов преобразования АЦП с линий R1-R14, R14 - знаковый разряд, 1- минус.
F(8)A(0) - проверка L.
F(10)A(0) - сброс L.
F(26)A(0) - разрешение L.
F(24)A(0) - запрет L.

2ЦАП-10

(Двухканальный 10-разрядный ЦАП, масштаб преобразования - 5 мВ/ед.кода, время преобразования - 10 мкс; одноразрядный регистр запросов)

F(16)A(0) - передача кода в первый ЦАП с линий W1-W10.
F(16)A(1) - передача кода во второй ЦАП с линий W1-W10.
F(17)A(0) - передача кода в оба ЦАП одновременно с линий W1-W10.
F(18)A(0) - передача кода в первый ЦАП и +1 - во второй.
F(8)A(0) - проверка L.
F(10)A(0) - сброс L.
F(26)A(0) - разрешение L.
F(24)A(0) - запрет L.
C - сброс регистров данных ЦАП.
Z - сброс LAM - статуса, запрет L.

Модуль - коммутатор МК-1

(Релейный 16-канальный коммутатор, время коммутации - 10 мс)
F(16)A(0) - запись кода номера канала с линий W1-W4, L=1 через 10 мс после приема команды.
F(25)A(0) - включение следующего по номеру канала, L=1 через 10 мс после приема команды.
F(0)A(0) - чтение кода включенного канала.
F(8)A(0) - проверка L.
F(10)A(0) - сброс L.
F(26)A(0) - разрешение L.
F(24)A(0) - запрет L.
F(27)A(0) - опрос L.
Z - сброс LAM - статуса, запрет L, включение первого канала.

Регистр управления реле РУР-1Р

F(16)A(0) - запись номеров включаемых каналов с линий W1-W16, каждый бит соответствует одному каналу.
F(26)A(0) - разрешение выдачи выходного сигнала.
F(0)A(0) - чтение номеров включенных каналов с линий R1-R16.
F(9)A(0) - сброс входного регистра.
F(24)A(0) - сброс статусного регистра и запрет выдачи выходного сигнала.
F(27)A(0) - опрос статусного регистра. Ответ - по Q. Q=1, если выдача разрешена.
Z, C - сброс входного и статусного регистров.

Таймер цифровой
(Тик таймера = 1 мкс)

F(16)A(0) - запись кода интервала времени, разряды 1-24.
F(16)A(1) - запись кода интервала времени, разряды 25-40.
F(16)A(2) - запись кода диапазона интервала времени.
F(26)A(1) - пуск таймера.
F(8)A(15) - проверка L.
F(10)A(0) - сброс L.
F(26)A(0) - разрешение L.
F(24)A(0) - запрет L.

```
/*************************************************************/
/*          ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ ДЛЯ                    */
/*    УПРАВЛЕНИЯ САМАС -АППАРАТУРОЙ  (Visual C++ 6.0)        */
/*      Start   СТАРТ САМАС-ЦИКЛА                            */
/*      Zero    ГЕНЕРАЦИЯ СИГНАЛА ZERO                       */
/*      Clear   ГЕНЕРАЦИЯ СИГНАЛА CLEAR                      */
/*      Inhibit ГЕНЕРАЦИЯ СИГНАЛА INHIBIT                    */
/*      Numb    УСТАНОВКА НОМЕРА АКТИВНОГО КК                */
/*      Enable  РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ ОТ IRQ2                */
/*     Disable  ЗАПРЕТ ПРЕРЫВАНИЯ ОТ IRQ2                    */
/*      NumLAM  ОПРЕДЕЛЕНИЕ НОМЕРА СТАНЦИИ МОДУЛЯ            */
/*        ИСТОЧНИКА ЗАПРОСА НА ПРЕРЫВАНИЕ                    */
/*      TSTL    ПРОВЕРКА СИГНАЛА L                           */
/*      TSTQ    ПРОВЕРКА СИГНАЛА Q                           */
/*      TSTX    ПРОВЕРКА СИГНАЛА X                           */
/*      Act     ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ          */
/*        БЕЗ ПЕРЕДАЧИ ДАННЫХ                                */
/*      ActW1B  ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ          */
/*        С ПЕРЕДАЧЕЙ  8-ми РАЗРЯДНОГО СЛОВА ДАННЫХ         */
/*        ЭВМ - САМАС                                        */
/*      ActW2B  ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ         */
/*        С ПЕРЕДАЧЕЙ 16-ти РАЗРЯДНОГО СЛОВА                 */
/*        ДАННЫХ ЭВМ - САМАС                                 */
/*      ActW3B  ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ         */
/*        С ПЕРЕДАЧЕЙ 24-х РАЗРЯДНОГО СЛОВА                  */
/*        ДАННЫХ ЭВМ - САМАС                                 */
/*      ActR1B  ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ         */
/*        С ПЕРЕДАЧЕЙ 8-ми РАЗРЯДНОГО СЛОВА                  */
/*        ДАННЫХ САМАС - ЭВМ                                 */
/*      ActR2B  ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ         */
/*        С ПЕРЕДАЧЕЙ 16-ти РАЗРЯДНОГО СЛОВА                 */
/*        ДАННЫХ САМАС - ЭВМ                                 */
/*      ActR3B  ВЫПОЛНЕНИЕ ОДИНОЧНОЙ САМАС-ОПЕРАЦИИ         */
/*        С ПЕРЕДАЧЕЙ 24-х РАЗРЯДНОГО СЛОВА ДАННЫХ           */
/*        САМАС - ЭВМ                                        */
/*      -Q  ВЫПОЛНЕНИЕ САМАС-ОПЕРАЦИИ                       */
/*        МЕТОДОМ ПОВТОРЕНИЯ С ПРОВЕРКОЙ СИГНАЛА Q           */
/*      -L  ВЫПОЛНЕНИЕ САМАС-ОПЕРАЦИИ                       */
/*        МЕТОДОМ ПОВТОРЕНИЯ С ПРОВЕРКОЙ СИГНАЛА L           */
/*      ПАРАМЕТРЫ: N    - НОМЕР СТАНЦИИ                     */
/*          A    - СУБАДРЕС                                 */
/*          F    - НОМЕР ФУНКЦИИ                            */
/*          dat1 - МЛАДШИЙ БАЙТ                             */
/*          dat2 - СРЕДНИЙ БАЙТ                             */
/*          dat3 - СТАРШИЙ БАЙТ                             */
/*          Num  - НОМЕР АКТИВНОГО КОНТРОЛЛЕРА КРЕЙТА       */
/*************************************************************/
```

```
void  Start(void);
void  Zero(void);
void  Clear(void);
void  Inhibit(void);
void  Numb(int);
void  Enable(void);
void  Disable(void);
 int  NumLam(void);
 int  TSTL(void);
 int  TSTQ(void);
 int  TSTX(void);

void  Act(int N, int A, int F);
void  ActW1B(int N, int A, int F, int W);
void  ActW2B(int N, int A, int F, int W);
void  ActW3B(int N, int A, int F, int W);
void  ActR1B(int N, int A, int F, int *data);
void  ActR2B(int N, int A, int F, int *data);
void  ActR3B(int N, int A, int F, int *data);


void  Act_Q(int N, int A, int F);
void  ActW1B_Q(int N, int A, int F, int W);
void  ActW2B_Q(int N, int A, int F, int W);
void  ActW3B_Q(int N, int A, int F, int W);
void  ActR1B_Q(int N, int A, int F, int *data);
void  ActR2B_Q(int N, int A, int F, int *data);
void  ActR3B_Q(int N, int A, int F, int *data);


void  Act_L(int N, int A, int F);
void  ActW1B_L(int N, int A, int F, int W);
void  ActW2B_L(int N, int A, int F, int W);
void  ActW3B_L(int N, int A, int F, int W);
void  ActR1B_L(int N, int A, int F, int *data);
void  ActR2B_L(int N, int A, int F, int *data);
void  ActR3B_L(int N, int A, int F, int *data);
```

```
int dat1, dat2, dat3;

void Start(void)
{ _outp(0x3A7,1);}

void Zero(void)
  { _outp(0x3A6,1);
    _outp(0x3A7,1); }

void Clear(void)
{ _outp(0x3A6,2);
  _outp(0x3A7,1); }

void Numb(int Num)
{ _outp(0x3AF,Num);
  _outp(0x3A7,1); }

int  TSTQ(void)
{ return(_inp(0x3A8) & 1);  }

void Act(int N, int A, int F)
{
_outp(0x3A5,N);
_outp(0x3A3,A);
_outp(0x3A4,F);
_outp(0x3A7,1);
}
void ActW1B(int N, int A, int F, int dat1)
{
_outp(0x3A5,N);
_outp(0x3A3,A);
_outp(0x3A4,F);
_outp(0x3A2,dat1);
_outp(0x3A7,1);
}
void ActR1B(int N, int A, int F, int *dat1)
{
_outp(0x3A5,N);
_outp(0x3A3,A);
_outp(0x3A4,F);
_outp(0x3A7,1);
*dat1=_inp(0x3AB);
}
}
```

```
/*****************************************************************************/
/*  Программа иллюстрирует использование функций библиотек LabWindows/CVI  */
/*  и библиотеки управления CAMAC-аппаратурой.                               */
/*  Программа  предназначена для формирования различных функциональных       */
/*  зависимостей в форме электрических сигналов. Используется модуль 2ЦАП-10  */
/*****************************************************************************/
#include <utility.h>

#include <userint.h>
#include <analysis.h>
#include <ansi_c.h>
#include "sample2.h"
#include "camlib.h"

int handle;
double datapoints[1000];
double data_adc[1000];

main()
{
   handle = LoadPanel (0, "sample2.uir", PANEL);
   DisplayPanel (handle);
   RunUserInterface ();
}

int  AcquireData(int panel, int control, int event, void *callbackData, int eventData1, int
eventData2)
{
   int trace_color, shape, i, j, dat;
   double phase=90;
   int DAC=18, C=48;
   int dat_adc, dadc;

   if (event == EVENT_COMMIT) {
   /*Выбор вида и формирование входного воздействия на объект*/
        GetCtrlVal (handle, PANEL_WFM, &shape);

        switch (shape) {
           case 0 :
               SineWave (128, 1.0, 20e-3, &phase, datapoints);
               for (i=0;i<100;i++) datapoints[i]=datapoints[i]*50+50;
              break;
           case 1 :
              SquareWave (128, 1.0, 20e-3, &phase, 50.0, datapoints);
              for (i=0;i<100;i++) datapoints[i]=datapoints[i]*50+50;
              break;
           case 2 :
              TriangleWave (128, 1.0, 20e-3, &phase, datapoints);
              for (i=0;i<100;i++) datapoints[i]=datapoints[i]*50+50;
```

```
            break;
          case 3 :
            for (i=0;i<100;i++) datapoints[i] = rand()/32767.0 * 100.0;
            break;
          }
        DeleteGraphPlot (handle, PANEL_GRAPH, -1, 1);
        PlotY (handle, PANEL_GRAPH, datapoints, 100, VAL_DOUBLE,
          VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_BLACK);

        Numb(C);Zero;Clear;   /*Начальная инициализация аппаратуры CAMAC*/
        for(j=0;j<8000;j++)       /* Внешний цикл для возможности наблюдения */
                                  /*   сигнала на осциллографе*/
        {
            /*Задание входного воздействия на объект*/
            for (i=0;i<100;i++) ActW2B(DAC,0,17,datapoints[i]*5);
        }
      }
   return(0);
}

int  Shutdown(int panel, int control, int event, void *callbackData, int eventData1, int
eventData2)
{
   if (event == EVENT_COMMIT)
      QuitUserInterface(0);
   return(0);
}
```

Рис 1. Панель интерфейса пользователя.

```
/************************************************/
/* ПРОГРАММА ИЗМЕРЯЕТ ВРЕМЯ ВЫПОЛНЕНИЯ */
/* ФРАГМЕНТА ПРОГРАММЫ                          */
/************************************************/

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <time.h>
#include "camaclib.c"

int i;

void main (void)
{
  clock_t start, end;
  start = clock();

  for (i=0;i<1000000;i++)
 /* Здесь нужно записать текст фрагмента */
 /* программы  */
  end = clock();
  printf("The time, mksec: %f\n", (end - start) / CLK_TCK);
}
```

Пример программы на Visual C++ измерения времени выполнения отдельных фрагментов программного кода приведен в разделе «Высокоскоростные системы сбора данных и управления с интерфейсом ISA»

Работа 7. Система автоматизированного проектирования программного обеспечения LabWindows/CVI 8.0

Целью лабораторной работы является изучение методики проектирования на языке С программного обеспечения автоматизированных систем измерения и управления, созданных на основе компьютера IBM PC и аппаратуры измерения и управления, подключенной к компьютеру с помощью стандартных интерфейсов, сервисной части программ, включающей средства графического пользовательского интерфейса, и математической обработки, разработки виртуальных приборов.

Теоретические основы

Программные средства автоматизированных систем измерения и управления обычно создаются на языке С/С++, так как в этом языке предусмотрены средства, обычно необходимые при управлении аппаратурой: удобные средства работы с битами, логические операции, средства работы с аппаратными прерываниями и др. Кроме того, компилятор С обеспечивает формирование исполняемой программы, близкой по скорости исполнения к программе на ассемблере.

Однако в стандартном С/С++ отсутствуют библиотеки функций для сложных видов математической обработки, таких как быстрое преобразование Фурье, цифровой фильтрации, корреляционной обработки и др., библиотеки программ управления аппаратурой, подключенной к компьютеру с помощью стандартных интерфейсов, библиотеки программ для создания объектов графического пользовательского интерфейса, обычно необходимые при создании программного обеспечения автоматизированных систем измерения и управления. В настоящее время разработаны интегрированные средства разработки программного обеспечения автоматизированных систем измерения и управления, такие как LabWindows/CVI, LabView, содержащие все эти компоненты.

LabWindows/CVI включает встроенный компилятор языка С, инструментальную библиотеку для работы с аппаратурой, сопрягаемой с компьютером с помощью стандартных интерфейсов GPIB, VXI, ISA, CAMAC, RS232, библиотеку Analysis Library для математической обработки, библиотеку Uses Interface Library для создания средств графического пользовательского интерфейса. LabWindows/CVI содержит также средства автоматизации создания текстов и отладки программ на языке С.

Программные средства, создаваемые для работы в среде LabWindows/CVI, содержат три составляющие:

1. Собственно программу на языке С, в которой содержатся необходимые функции управления аппаратурой, математической обработки, ввода-вывода, отображения. При ее создании следует пользоваться описаниями функций библиотек работы с аппаратурой (например, RS232 library), математической обработки (Analysis Library), графического пользовательского интерфейса (User Interface Library), ввода-вывода (Formatting and I/O Library).
2. Файлы <имя программы>.h, lwsystem.h, userint.h, analysis.h, formatio.h.
3. Файлы ресурсов графического пользовательского интерфейса <имя программы>.uir.

Создаваемая пользователем прикладная программа представляет проект, содержащий четыре файла:

- файл проекта <имя>.prj;
- файл основной программы <имя>.c;
- файл макета <имя>.uir;
- файл заголовков <имя>.h

Программа на языке С может быть написана и редактироваться пользователем. Файлы макета <имя>.uir и заголовков <имя>.h создаются автоматически при создании и редактировании пользователем графических панелей. Эти файлы нельзя редактировать!

LabWindows/CVI позволяет запускать и выполнять программы, состоящие из этих четырех компонентов, а также создавать EXE-модули, которые могут быть запущены на другом компьютере вне среды LabWindows/CVI.

Роль средств графического пользовательского интерфеса в программах автоматизации эксперимента.

Основными элементами любой автоматизированной системы измерения и управления является компьютер и аппаратура сопряжения компьютера с объектом управления. Аппаратура сопряжения должна быть связана с компьютером с помощью интерфейса. Физически аппаратура может быть выполнена как в виде модулей сбора данных и управления, встраиваемых в компьютер, так и в виде отдельных приборов с автономным питанием и органами ручного управления или индикации (тумблерами, кнопками, сигнальными лампочками, переключателями, стрелочными и шкальными указателями). В последнем случае, в особенности, если весь комплекс приборов встроен в стойку или шкаф, органы управления, контроля и индикации образуют так называемую приборную панель. С этой панели пользователь может управлять работой системы: задавать начальные «уставки», контролировать ход процесса управления, следить за аварийными ситуациями. Большое разнообразие традиционных средств управления, контроля и индикации на приборной панели (средств пользовательского интерфейса автоматизированной системы) являлось бесспорным преимуществом по

сравнению с обычными способами представления информации на компьютере только в цифровой или графической форме.

Новый подход к созданию средств пользовательского интерфейса в автоматизированных системах измерения и управления основан на концепции так называемых виртуальных инструментов. Виртуальными инструментами называют приборы и системы для измерения и управления, созданные на основе компьютера и сопряженной с ним аппаратурой измерения и управления. Виртуальные инструменты отличаются от традиционных главным образом тем, что функция пользовательского интерфейса реализуется с помощью компьютера программно. Все элементы управления, контроля и индикации отображаются на экране дисплея компьютера, выглядят внешне и действуют точно так же, как и физически существующие (кнопки можно нажимать, переключатели переключать, ручки потенциометров вращать, стрелки приборов движутся). Управление средствами ввода (кнопками, потенциометрами, переключателями и т. д.) производится «мышью». Состав, размещение на экране, размеры, цвет и другие атрибуты органов управления, контроля и индикации на виртуальной панели можно легко программно изменять.

Создание средств графического пользовательского интерфейса в традиционных системах программирования, таких как Visual C++ возможно, но достаточно сложно, так как для этого необходимо создание большого количества нестандартных графических объектов. Значительные удобства для создания средств графического пользовательского интерфейса предоставляют специальные системы программирования, такие как LabWindows/CVI (C Virtual Instruments) фирмы National Instruments.

Создание прикладной программы автоматизации эксперимента в среде LabWindows/CVI 8.0.

Загрузка и запуск LabWindows CVI 8.0

Для загрузки и запуска LabWindows CVI 8.0 в среде ОС Windows 2000/XP нужно последовательно вызвать из меню «Пуск»:

Programs/National Instruments /LabWindows CVI 8.0/NI LabWindows CVI

Создание проекта

1. Создать файл проекта.
   а) **File/New/*.prj**
   б) сохранить созданный файл проекта
      File/Save Untitled.prj As.../{имя файла}
2. Создать файл макета.
   а) **File/New/UserInterface (*.uir)…**

б) сохранить созданный файл макета
File/Save Untitled.uir As.../{имя файла}

В результате на экране дисплея появляется изображение панели UIR. Размеры и место расположения панели на экране можно установить с помощью мыши. Далее нужно установить атрибуты панели. Для этого перейти в режим редактирования панели двойным щелчком мыши и во всплывающей панели задать:

- Constant Name - идентификатор панели в программе, например, оставить идентификатор PANEL, предлагаемый по умолчанию;

- Panel Title - заголовок, например " Digital Oscilloscope "
- и другие.

4. Создать элементы GUI (графические окна вывода, управляющие и задающие циферблаты, лампочки, командные кнопки и т.д.) из библиотеки элементов и разместить их на созданной панели.

Create/<имя элемента GUI>

Объекты GUI (кнопки, переключатели, сигнальные лампочки и др.) выбираются из меню и размещаются на экране методом "drag and drop" с помощью "мыши".

LabWindows/CVI 8.0, работающий под управлением Windows-2000/XP, предоставляет широкий выбор выбор объектов GUI в каждом из следующих классов :

Numeric- окно цифрового ввода/вывода;
Text – окно ввода текста;
Command Button – кнопка с двумя состояниями;
Toggle Button – кнопка с двумя состояниями (включено/выключено);
LED - сигнальный индикатор;
Binary switch - выключатель с двумя состояниями;
List & Tables – списки и таблицы с элементами прокрутки;
Decoration – элементы оформления;
Graph – окно вывода массивов данных в графической форме;
Pictures – окна для ввода изображений;
Timer – таймеры;
Canvas – окно для ввода фонового изображения панели;
Splitter – средства создания многостраничной панели;
Graph - окно вывода массивов данных в графической форме;
Strip chart - окно вывода в графической форме скользящей выборки из числового массива данных .

При выполнении лабораторной работы рекомендуется использовать следующие элементы управления и ввода/вывода: Numeric, Graph, Command button.

4.1 Задание атрибутов элементов GUI.

Сделать двойной щелчок левой клавишей мыши на объекте GUI, далее во всплывающей панели задать его атрибуты, в первую очередь:

Constant Name - идентификатор элемента GUI в программе;

Label - обозначение элемента GUI на панели;

Callback Function - функция вызова.

Функция вызова - это имя функции в программе, которая будет запущена однократным щелчком мыши на объекте GUI.

Для командных кнопок, например, Start и Quit должны быть заданы атрибуты Callback Functions, для элементов ввода/вывода - Constant Name. Имена функций вызова не должны совпадать с ключевыми словами языка C++, поэтому рекомендуется использовать префиксы, например, вместо имен Start и Quit использовать sStart и sQuit.

Далее нужно сохранить сделанные установки.

File/Save (Ctrl-S)
/Save As.../{имя файла}

5. Создать программный код (Текст программы на языке C++).

а) **Code/Generate all code...**

б) на всплывающей панели “Generate All Code” в окне “Target Files” установить “Add To Current Project”;

в) в окне “Program Termination” установить флажок в строке с наименованием функции вызова (например, sStop) программы завершения;

г) [OK]

В результате этого будут созданы файлы на языке C++ с расширениями *.c и *.h . В файле *.c находится сгенерированный код.

В созданном программном коде объектам GUI, для которых определен CallbackFunction, будет соответствовать " программная оболочка" функции на языки C++. Эти функции (за исключением функции работы таймера) будут запускаться на исполнение пользователем с панели GUI. Функция работы таймера начинает работать сразу же после запуска программы. В приложении 1 приведен пример созданной таким путем «программной оболочки».

Далее в эту оболочку вносится программный код прикладной программы на языке C++.

Если производится редактирование ранее уже созданного программного кода, например, при добавлении какого-либо объекта GUI, то для добавления в ранее созданный программный код изменений, связанных с добавленными элементами GUI:

Code/Insert Function Call (Ctrl-I)

6. Добавить обращение к элементам GUI в текст программы.

Для этого:
а) открыть уже созданный автоматически по п.5 файл с текстом программы на язке С;

б)установить курсор в месте вставки строки с обращением к элементу GUI, например, Graph или Ring Slide;
в)Library/User Interface/Strip Chart;

в) выбрать соответствующую функцию, например:
GetControlValue;
DeleteGraphPlot;
PlotY.

г)заполнить во всплывающей панели параметры обращения к функции;
д)сгенерировать программный код:
Code/InsertFunctionCall

7. Добавить в программу функцию выхода.

Для этого в уже сгенерированную по п.5 функцию выхода программы необходимо внести строку обращения к функции Quit User Interface:

Library/UserInterface/QuitUserInterface

8. Добавить в программный код строки, которые автоматически не генерируются.

В результате будет получен полный программный код прикладной программы. Простой пример такой программы генерации и графического отображения массива случайных чисел приведен ниже.

9. Выполнить сборку проекта.

Edit/AddFilesToProject/Source *.c
/Include *.h
/UserInterface *.uir

После этого создание проекта закончено.

Редактирование программного кода, в том числе строк обращения к объектам GUI производится точно так же, как в любом C/C++.

10. Скомпилировать программу.

Build/Compile File

Если компилятор найдет ошибки, то устранить ошибки и повторить предыдущее действие.

11. Запустить программу.

Run/Run

Программа работы

1. Изучить состав и функции библиотек программ LabWindows/CVI 8.0 для работы с аппаратурой, математической обработки, создания средств графического пользовательского интерфейса.
2. Разработать программу генерации стандартных сигналов: гармонического, треугольного с шумом. Выбор вида сигнала, амплитуду, частоту сигналов и уровень шума сделать регулируемыми
3. Оценить экспериментально время ввода/вывода данных на экран дисплея в цифровой и графической форме средствами графического пользовательского интерфейса.
4. Разработать на базе лабораторной установки виртуальный прибор.

Указания к выполнению лабораторной работы

1. При выполнении п. 1 Программы изучить состав и функции библиотек программ LabWindows/CVI пользуясь справочной системой.
2. При выполнении п.2 Программы создать графический пользовательский интерфейс для прикладной программы генерации массива случайных чисел. Основные функции пользовательского интерфейса: задание

количества элементов вычисляемого массива и отображение числового массива в графической форме.

На панели графического пользовательского интерфейса предусмотреть:

- клавиши запуска программы и выхода из программы;
- элемент ввода количества точек вычисляемого массива;
- окно графического вывода.

3. При выполнении п.3 Программы

Разработать прикладную программу генерации различных стандартных сигналов:

- гармонического (синусоидального);
- пилообразного;
- треугольного;
- прямоугольных импульсов.

без шума и с шумом. Выбор вида сигнала, амплитуду, частоту сигналов и уровень шума сделать регулируемыми.

На панели графического пользовательского интерфейса предусмотреть:

- переключатель выбора вида сигнала;
- элемент ввода уровня шума;
- элемент ввода количества точек за период генерируемого сигнала;
- окно графического вывода генерируемого сигнала.

3.2. Дополнить прикладную программу, предусмотрев в ней возможность накопления (суммирования с усреднением) генерируемых сигналов.

На панели графического пользовательского интерфейса дополнительно предусмотреть переключатель задания количества накоплений.

4. При выполнении п.4 Программы разработать виртуальный прибор - спектроанализатор.

Для этого дополнить прикладную программу по п.3, предусмотрев в ней возможность выполнения БПФ (быстрого преобразования Фурье) накопленного сигнала и отображения результатов обработки. На панели графического пользовательского интерфейса дополнительно предусмотреть:

- окно графического вывода для отображения спектров БПФ;
- кнопки записи массивов данных и результатов обработки в файл на магнитный диск.

Для выполнения БПФ использовать встроенные функции Advanced Analysis LabWindows/CVI в соответствии с приведенным ниже фрагментом программы.

Фрагменты программы выполнения БПФ

```
#include "prthires.h"
#include<userint.h>
#include<ansi_c.h>
#include <formatio.h>
#include <utility.h>
#include <stdlib.h>
……
int handle,maxi,mini,f,KT;
double max,min,vvv;
double dat1[1024];
double dat3[1024];
double dat4[1024];
double dat5[1024];
……..
        for (i=0;i<KT;i++)
    {
    dat3[i]=dat1[i]; dat4[i]=dat1[i];    /*dat3 и dat4 нужны для программы БПФ*/
    }
    /*Функция БПФ. Массив входных данных - dat3=dat4*/
    /*После выполнения БПФ в этих массивах будут результаты вычислений,*/
    /*в dat3 массив вещественных частей, в dat4 - мнимых*/
          ReFFT (dat3, dat4, KT);
    /*Вычисление массива выходного спектра и нахождение номера элемента
массива, соответствующего максимуму функции БПФ - maxi*/
            for (i=0;i<KT;i++)
    {
dat5[i]=sqrt((dat3[i]*dat3[i])+(dat4[i]*dat4[i]))/KT;
MaxMin1D (dat5, KT, &max, &maxi, &min, &mini);
/*Переход от числа maxi, соответствующего количеству периодов во
временном окне к значению  частоты. Здесь 25нс-интервал между отсчетами
при частоте отсчетов – 40МГц, KT-количество отсчетов, множитель 1000000 –
нужен для перехода значения частоты от ГГц к кГц*/
        f=maxi*1000000/(25*KT);
/*Вывод значения найденного максимума*/
     SetCtrlVal (handle, PANEL_F, f);
    }
 /*Вывод спектра в окно графического вывода*/
 DeleteGraphPlot (handle, PANEL_GRAPH2, -1, 1);
 PlotY (handle, PANEL_GRAPH2, dat5, 128, VAL_DOUBLE,
 VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_BLUE);
     }
```

Панель пользовательского интерфейса анализатора спектров спроектировать согласно рис. 1.

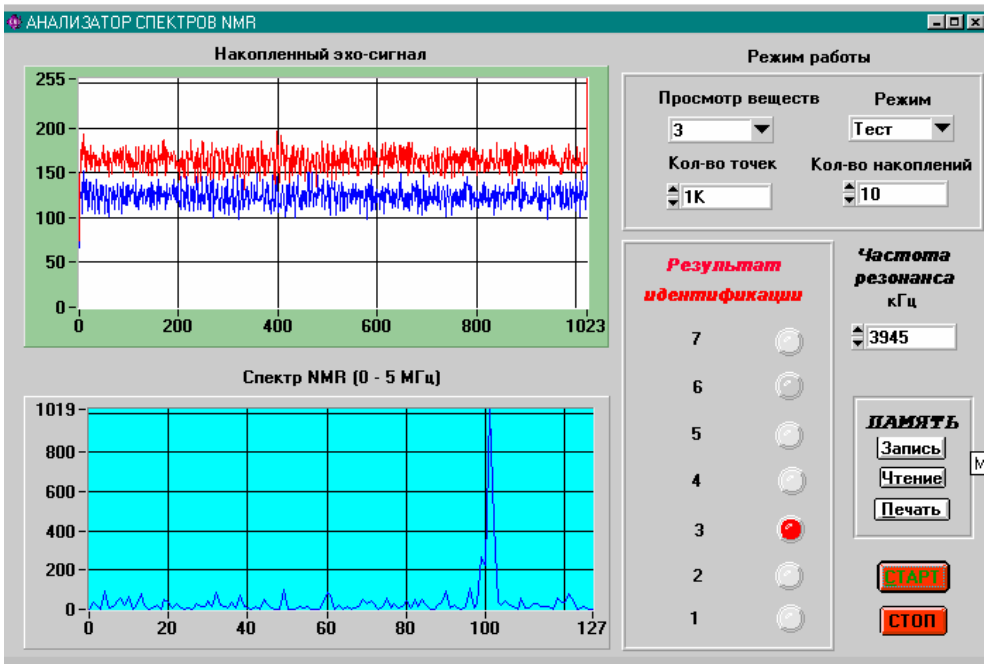


Рис. 1

Приложение 1.

Пример программного кода, генерируемого автоматически после создания панели и объектов GUI в среде LabWindows/CVI

```
#include <cvirte.h>
#include <userint.h>
#include "first.h" /*имя .h-файла, входящего в проект*/

static int panelHandle;

int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)
                return -1;      /* out of memory */
        if ((panelHandle = LoadPanel (0, "first.uir", PANEL)) < 0)
                return -1;
        DisplayPanel (panelHandle);
        RunUserInterface ();
        DiscardPanel (panelHandle);
        return 0;
}

int CVICALLBACK sStart (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:
    /*  Здесь должен будет находиться текст программы    */
      /* Функция вызова программы (CallbackFunction) - sStart*/
                        break;
        }
        return 0;
}

int CVICALLBACK sStop (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:
/*При вызове этой функции будет происходить выключение */
/*графического пользовательского интерфейса и завершение работы */
                        QuitUserInterface (0);
                        break;
        }
        return 0;
}
```

Приложение 2

Пример программного кода, генерируемого автоматически в результате интерактивного обращения к панелям функций библиотек UserInterfaceLibrary и AnalysisLibrary в среде LabWindows/CVI

```
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "first.h"
static int panelHandle;

int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)
                return -1;      /* out of memory */
        if ((panelHandle = LoadPanel (0, "first.uir", PANEL)) < 0)
                return -1;
        DisplayPanel (panelHandle);
        RunUserInterface ();
        DiscardPanel (panelHandle);
        return 0;
}
int CVICALLBACK sStart (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
int i;  /*эта строка автоматически не генерируется*/
double y[100];  /*эта строка автоматически не генерируется*/

        switch (event)
        {
        case EVENT_COMMIT:
       for (i=0;i<100;i++) y[i]=rand()/32767.0;
DeleteGraphPlot (panelHandle, PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
PlotY (panelHandle, PANEL_GRAPH, y, 100, VAL_DOUBLE, VAL_THIN_LINE,
VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);
        break;
        }
        return 0;
}
int CVICALLBACK sStop (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
        case EVENT_COMMIT:
        QuitUserInterface (0);
                break;
        }
        return 0;
```

}**Приложение 3**

Пример программы измерения времени выполнения фрагмента программы. СП - LabWindows/CVI (ANSI C)

```
#include <utility.h>
#include <formatio.h>
#include <ansi_c.h>

/*Функция временной задержки. 1 тик=1мс*/
void ourdelay(long ticks)
{
int k,j;
unsigned char a;
for (k=0;k<ticks;k++) for (j=0;j<375;a=inp(0x3A0));
}
void main()
{
clock_t start,end;

start=clock();
ourdelay(1000); /*Это пример измеряемого фрагмента программы*/
end=clock();
FmtOut("Results=%f seconds",difftime(end,start)/CLOCKS_PER_SEC);
getchar();
}
```

Работа 8. Обработка результатов однофакторного эксперимента в среде MATLAB. Подгонка кривых.

Теоретические основы.

Обработка результатов однофакторного эксперимента включает 3 этапа: выбор вида аппроксимирующей зависимости, расчет коэффициентов аппроксимирующей зависимости и проверку адекватности полученной зависимости экспериментальным данным. Наиболее совершенным программным средством для решения данной задачи является пакет программ MATLAB. MATLAB является средой разработки программ обработки данных и, одновременно, содержит большое количество готовых программ, в частности, программу подгонки кривых CurveFitting.
Программа подгонки кривых CurveFitting предоставляет следующие виды функций для аппроксимации и интерполяции одномерных массивов данных:
а) экспоненциальную;

$$y = ae^{bx}$$
$$y = ae^{bx} + ce^{dx}$$

б)гауссиан

$$y = \sum_{i=1}^{n} a_i e^{\left[-\left(\frac{x-b_i}{c_i}\right)^2\right]}$$

в)Фурье

$$y = a_0 + \sum_{i=1}^{n} a_i \cos(nwx) + b_i \sin(nwx)$$

г)полиномиальную;

$$y = \sum_{i=1}^{n+1} p_i x^{n+1-i}$$

д)показательную;

$$y = ax^b$$
$$y = a + bx^c$$

е)рациональную.

$$y = \frac{\sum_{i=1}^{n+1} p_i x^{n+1-i}}{x^m + \sum_{i=1}^{m} q_i x^{m-i}}$$

Критерием качества подгонки являются вычисляемые программой коэффициенты множественной детерминации (R-square и Adjusted R-square). Коэффициент R-square показывает насколько успешно разброс данных относительно аппроксимирующей кривой может быть объяснен наличием случайных погрешностей в данных. Максимально возможное значение R-square равно 1.

Коэффициент Adjusted R-square характеризует то же самое, но учитывает количество степеней свободы дисперсий воспроизводимости и адекватности. Поэтому Adjusted R-square лучше характеризует степень соответствия экспериментальных данных аппроксимирующей кривой в том случае, если мы увеличиваем степень аппроксимирующего полинома и хотим проверить, происходит ли при этом улучшение качества подгонки. . Максимально возможное значение Adjusted R-square также равно 1.

Программа работы.

1. Освоить технологию работы с программой CurveFitting пакета . MATLAB.
2. Научиться производить обоснованный выбор наилучшей аппроксимирующей зависимости из возможных.

Указания к выполнению.

1. При выполнении п.1. Программы изучить теоретические основы подгонки кривых, положенные в основу программы Curve Fitting и описание библиотеки аппроксимирующих функций Fitting Library в MATLAB Help: CurveFittingToolbox/FittingData/ParametricFitting (Library Models, Custom Equations, Evaluation of goodness of Fitting и др.).
2. При выполнении п.2 выполнить подгонку кривых, выбрать наилучшую функциональную зависимость и обосновать сделанный выбор для зашумленного массива данных: файл census.mat (censu____.mat) в папке C:\MATLAB6p5\toolbox\mathlab\demos.
3. При работе с программой CurveFitting подгонку кривых производить в следующей последовательности:

а) Запустить MATLAB.

б) открыть окно Workspace (View/Workspace) и загрузить в это окно mat-файл данных по заданию преподавателя (census.mat, censu1.mat, censu2.mat и т.д.)

в) открыть окно LaunchPad. Для этого выбрать пункт меню MATLAB: View/LaunchPad. (В MATLAB 7.0.1 открытие окна LaunchPad не требуется)

г) из окна LaunchPad открыть панель CurveFitting: Toolboxes/ CurveFitting/ CurveFittingTool; (В MATLAB 7.01 для открытия панели CurveFitting выбрать Start/ Toolboxes/ CurveFitting)

д) на панели CurveFittingTool активизировать кнопку Data. В окнах Xdata, Ydata станут доступными составляющие cdate (X) и pop (Y) из файла census.mat;

е) произвести выбор cdate и pop, затем активизировать кнопку CreateDataSet;

ж) активизировать кнопку Fitting. В результате появится сдвоенное окно FitEditor и Tables of Fits . Окна нужно «раздвинуть» как это показано на рис.2, иначе второе окно будет закрывать доступ к инструментам во втором окне.

з) выбрать вид аппроксимирующей функции и активизировать кнопку Apply.

Результаты аппроксимации будут представлены в графической форме в окне графического вывода на панели CurveFittingTool (рис.1) и в

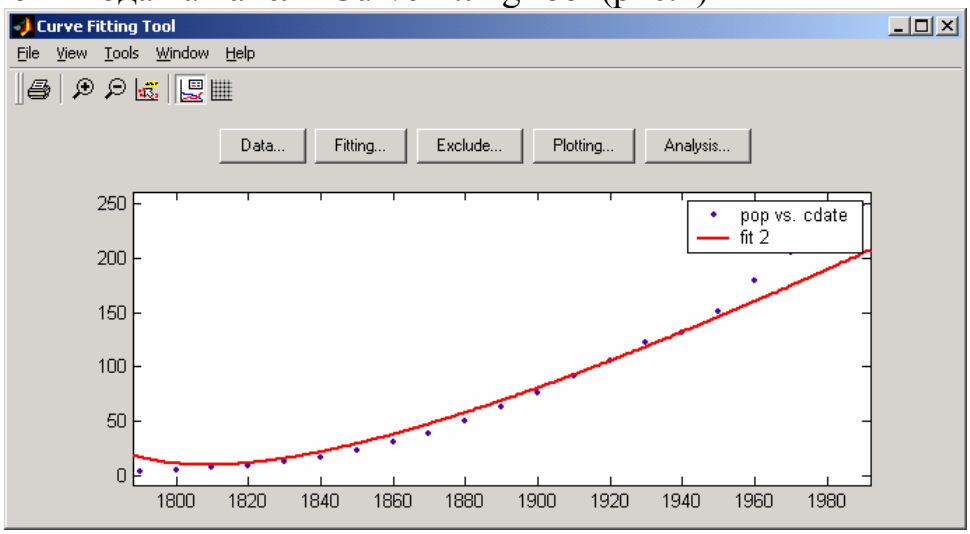


Рис. 1.

численной форме в окне Results на панели Tables of Fits (рис. 3).

При выборе пунктов меню View/Residuals/Line или View/Residuals/Scatter в окне графического вывода на панели CurveFittingTool будут одновременно выведен график ошибки интерполяции (см. пример на рис. 4). Окно графического вывода можно вывести на печать или выполнить в виде рисунка: File/Print to Figure.

Полученные в процессе аппроксимации результаты занести в таблицу:

Вид модели	Порядок модели	Качество подгонки (+ или -)	R-square	Adjusted R-square

Исследовать таким образом все возможные виды моделей, порядок модели - до 5.

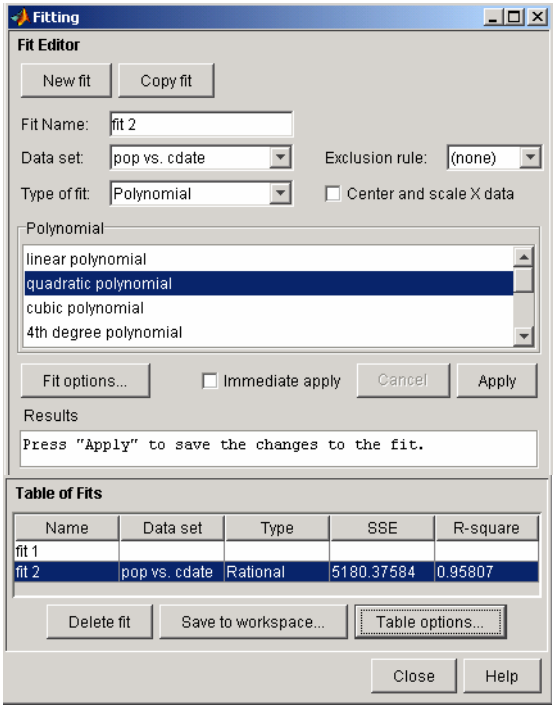


Рис.2

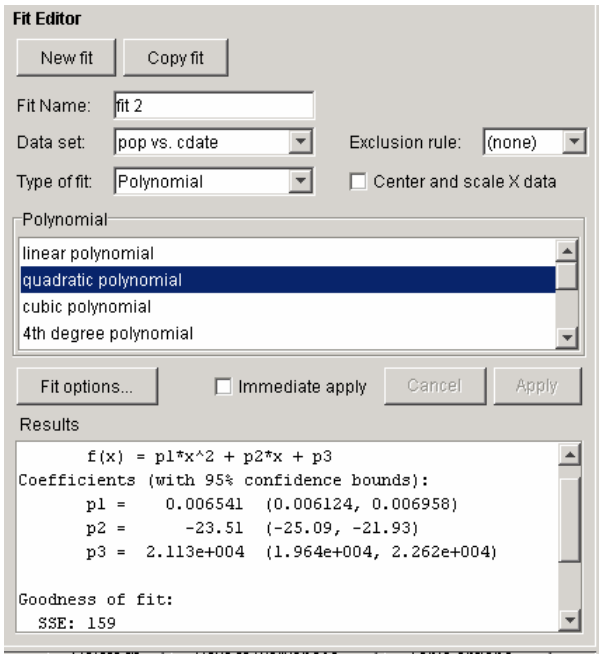


Рис. 3

е) по полученным и отраженным в таблице данным выбрать три наилучшие варианта аппроксимирующих функций. Для этих вариантов получить значения график аппроксимирующей функции и график погрешности аппроксимации, аналитические выражения с численными значениями коэффициентов для аппроксимирующей зависимости.

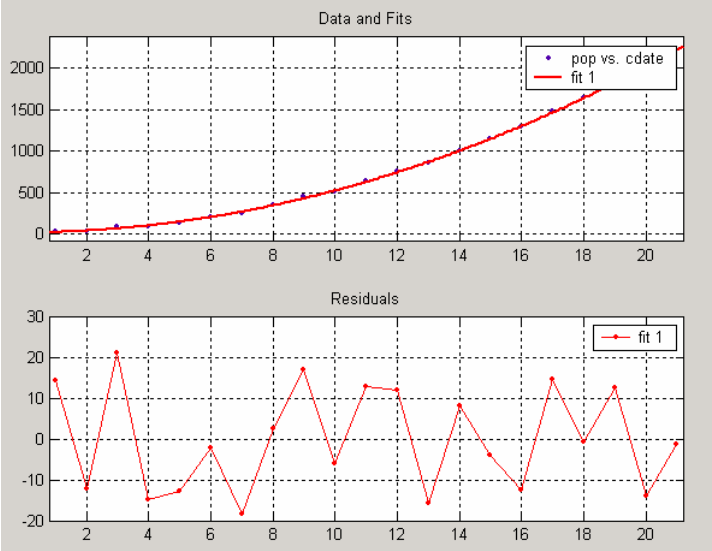


Рис. 4

Содержание отчета.

1. Постановка задачи.
2. Результаты аппроксимации экспериментальных данных из файла census.mat и censusN.mat и показатели качества подгонки R-square и Ajusted R-square для всех основных классов аппроксимирующих функций (экспоненциальной, гауссиана, показательной, Фурье, полиномиальной, показательной, рациональной) в форме таблицы результатов, содержащей название класса аппроксимирующей функции, порядка модели, значений R-square и Ajusted R-square.
3. Для трех наиболее подходящих аппроксимирующих зависимостей - график аппроксимирующей функции и график погрешности аппроксимации, аналитические выражения для аппроксимирующей зависимости.
4. Обоснование выбора наилучшей аппроксимирующей зависимости.
5. Выводы.

Работа 9. Планирование и обработка результатов полного многофакторного эксперимента

Цель работы: изучение методики планирования и обработки результатов полного многофакторного эксперимента.

Теоретические основы

Задача обработки экспериментальных данных многофакторного эксперимента состоит в нахождении математического описания многопараметрического явления, т.е. в построении его математической модели в виде

$$y = F(x1, x2, x3, \dots , xk)$$

Задача эта решается в следующем порядке.

1. Определение вида частных зависимостей Z1 (x1),Z2(x2), … ,Zk(xk), под которыми x1, … ,xk входят в общее выражение при фиксированных значениях остальных аргументов.
2. Определение общего вида математической модели, т.е. вида объединения частных функций между собой при образовании общей модели

$$1.\ X = \Psi [z1(x1), z2(x2), \dots zk(xk)]$$

3. Определение числовых значений коэффициентов модели.
4. Определение значимости отдельных членов полученного выражения и исключение малозначимых членов для получения наиболее компактной математической модели.
5. Уточнение числовых значений коэффициентов модели после исключения малозначимых членов.
6. Проверка адекватности полученной модели экспериментальным данным.

1. Определение вида частных зависимостей.

Для определения общего вида каждой частной зависимости проводят несколько опытов при фиксированном значении всех факторов, кроме одного. Значение же этого, единственного, фактора изменяют во всем диапазоне варьирования от минимального до максимального значения. Минимальное количество опытов зависит от характера предполагаемой зависимости. Если зависимость линейная - достаточно двух опытов, квадратичная - трех и т. д. При наличии случайных погрешностей измерения в каждом опыте полезно проводить повторные измерения.

Подбор вида модели при ручной обработке производится качественно, путем выбора из справочника типовых кривых функции, качественно похожей на полученную в результате аппроксимации "на глаз" экспериментальных данных.

2. Определение общего вида математической модели.

Определение общего вида математической модели на основе частных зависимостей может быть произведено при анализе семейств частных зависимостей. Например, для случая функции двух переменных это $Z_{1i} = Z_{1i}(x_1)$ $x_{2i} = C_i$, $C_{1i} = const$, $Z_{2i} = Z_{2i}(x_2)$, $x_{1i} = C_{2i}$.

На рис. 15.1а-г приведены примеры семейств частных зависимостей, соответствующих различным видам общей математической модели.

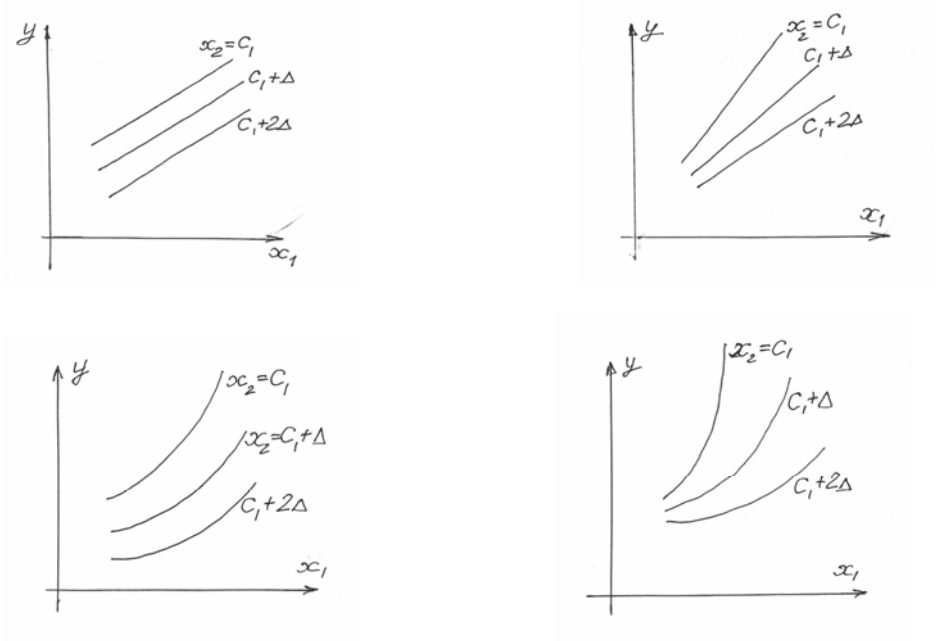


Рис.1а-г

В результате такого графоаналитического анализа может быть получен общий вид модели с неопределенными или, в лучшем случае, с приближенными значениями коэффициентов, например:

$$Y=b_0+b_1*x_3+b_2*\sin(x) \qquad (1)$$

3. Определение числовых значений коэффициентов модели.

Определение искомых значений коэффициентов производится обычно методом наименьших квадратов (МНК).

Минимальное количество опытов, которые необходимо выполнить для определения коэффициентов в уравнении регрессии методом наименьших квадратов, равно количеству коэффициентов в уравнении.

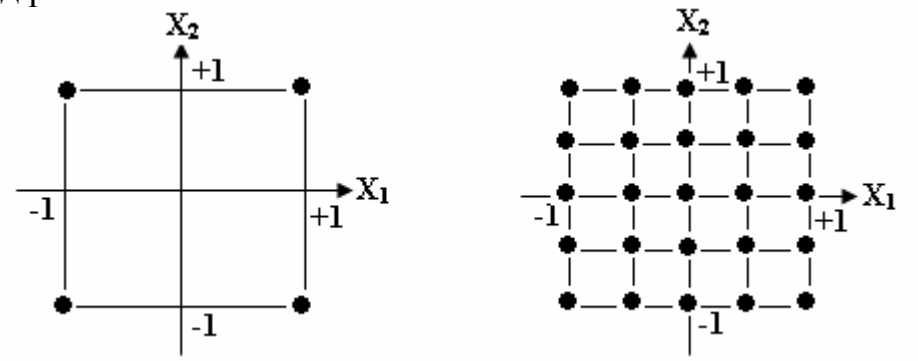
От выбора комбинаций значений факторов, при которых производятся опыты, зависит точность расчета коэффициентов модели.

Принцип целенаправленного выбора комбинаций факторов, который обеспечивает максимально высокую точность расчета значений коэффициентов по МНК при минимальном количестве опытов, называют планированием эксперимента.

Наиболее простым, наглядным и часто используемым является т.н. прямоугольный план, который для случая двух независимых переменных (факторов) и линейной зависимости

$$Y = b_0 + b_1*x_1 + b_2*x_2$$

имеет вид рис. 2а.



а) Рис.2 б)

Здесь независимые переменные центрированы и нормированы, т.е. приведены к виду

$$X_1 = \frac{x_1 - \bar{x}_1}{\Delta x_1}; \quad X_2 = \frac{x_2 - \bar{x}_2}{\Delta x_2}$$

Центрированные и нормированные факторы X1 и X2 изменяются в пределах от -1 до +1.

В общем случае, если модель не является линейной, количество участков разбиения диапазона варьирования каждого из факторов должно соответствовать показателю степени, с которой данный фактор входит в уравнение. Так при зависимости

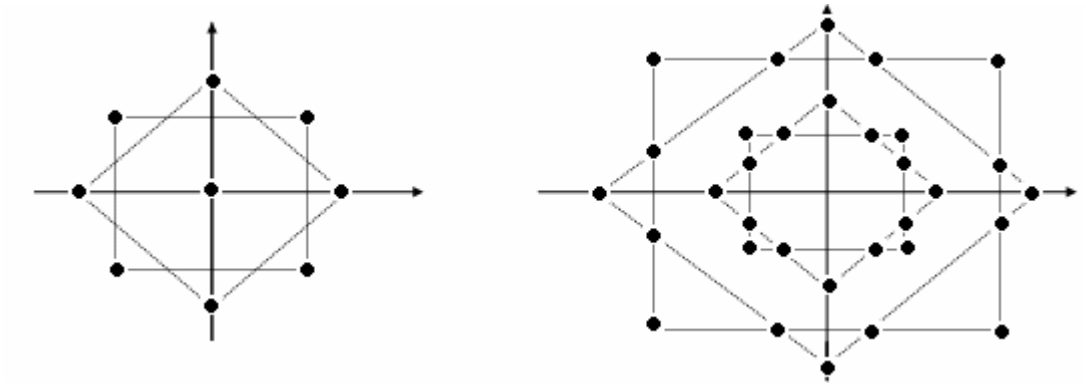
$$Y = b_0 + b_1*x_1^4 + b_2*x_2^4$$

прямоугольный план имеет вид рис 2б. Шаг разбиения желательно выбирать из условия

$$\Delta y = \Delta y(\Delta x) = \text{const}$$

Наиболее совершенным видом плана эксперимента, обеспечивающим наибольшую точность получения модели, является прямоугольный

ротатабельный план. Для случаев, когда оба фактора входят в уравнения регрессии во второй и четвертой степенях прямоугольный ротатабельный план имеет вид рис. 3а и 3б соответственно:



а) Рис.3 б)

4.Отбор наиболее значимых факторов

В связи с тем, что общий вид модели выбирается экспериментатором, может оказаться, что некоторые члены модели не имеют функциональной связи с величиной Y.

4.1. Отбор наиболее значимых факторов с использованием коэффициентов значимости

Идея метода заключается в следующем. Исходное уравнение регрессии

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_kx_k$$

вначале преобразуется к виду

$$y_i = a_0 + a_1x_{1i} + a_2x_{2i} + \ldots + a_kx_{ki}$$

и

$$\bar{y} = a_0 + a_1\bar{x}_1 + a_2\bar{x}_2 + \ldots + a_k\bar{x}_k$$

так как уравнение модели применимо для любого значения y, в том числе и для среднего значения. Затем производится вычитание одного уравнения из другого

$$y_i - \bar{y} = a_1(x_{1i} - \bar{x}_1) + a_2(x_{2i} - \bar{x}_2) + \ldots + a_k(x_{ki} - \bar{x}_k)$$

и далее

$$\frac{y_i - \bar{y}}{\sigma_y} = a_1\frac{\sigma_{x1}}{\sigma_y} * \frac{(x_{1i} - \bar{x}_1)}{\sigma_{x1}} + a_2\frac{\sigma_{x2}}{\sigma_y} * \frac{(x_{2i} - \bar{x}_2)}{\sigma_{x2}} + \ldots + a_k\frac{\sigma_{xk}}{\sigma_y} * \frac{(x_{ki} - \bar{x}_k)}{\sigma_{xk}}$$

Затем производится замена переменных

$$\beta_j = a_j \frac{\sigma_{xj}}{\sigma_y} \quad u \quad t_j = \frac{(x_{ji} - \bar{x}_j)}{\sigma_{xj}}$$

В результате замены переменных все переменные выразятся в долях среднеквадратического отклонения, т.е. последнее выражение будет представлено в виде:

$$t_y = \beta_1 t_1 + \beta_2 t_2 + ... + \beta_k t_k$$

где

$$t_y = \frac{y_i - \bar{y}}{\sigma_y}; \quad t_1 = \frac{x_{1i} - \bar{x}_1}{\sigma_{x1}}; \quad t_2 = \frac{x_{2i} - \bar{x}_2}{\sigma_{x2}}; \quad \ldots$$

$$\beta_1 = a_1 \frac{\sigma_{x1}}{\sigma_y}; \quad \beta_2 = a_2 \frac{\sigma_{x2}}{\sigma_y}; \quad \ldots$$

В последнем уравнении все t будут одного порядка, а т.к. $\bar{I}\ x_{ij}-x_i\ I < 3\sigma$ (т.н. правило трех сигм), то $t < 3$. Поэтому соотношение между собой $b_1, b_2, \ldots b_k$ указывает на относительный вес вклада в общую сумму слагаемых xji - xi.
Правда, в силу того, что функции распределения вероятностей xi могут быть различны, значения β характеризуют значимость факторов с погрешностью порядка 50%. Поэтому только если, например, b_2 существенно меньше b_j, $j >< 2$ (в 5 - 10 раз), то можно на этом основании исключать член, содержащий x_2 , из уравнения модели.
Значения σ_{xi} могут быть рассчитаны до эксперимента в соответствии с планом эксперимента по значениям x_{ji}, значение σ_y - по результатам эксперимента. Расчет коэффициентов β может быть произведен только после определения коэффициентов b_i .

5.Уточнение числовых значений коэффициентов модели

Для уточнения числовых значений коэффициентов модели после исключения малозначимых членов вновь применяется МНК. При этом используются имеющиеся массивы экспериментальных данных y_i, x_{ji}. или уточняется план эксперимента и проводятся дополнительные измерения.

6. Проверка адекватности модели экспериментальным данным

Проверку адекватности модели при наличии случайных погрешностей измерения принято производить по критерию Фишера или, что то же, по F - критерию:

$$F_{расч} = \frac{Max(S^2_{ад}, S^2_{воспр})}{Min(S^2_{ад}, S^2_{воспр})} < F_{табл}$$

где $S^2_{ад}$ - дисперсия адекватности;
$S^2_{воспр}$ -дисперсия воспроизводимости.

Оценка дисперсии воспроизводимости вычисляется на основе экспериментальных данных по формуле:

$$\sigma^2{}_{воспр} = \frac{\sum_{q=1}^{n}(y_{0q} - \bar{y}_0)^2}{n-1}$$

где y_{0q} -экспериментально полученное значение отклика в результате q-того повторного измерения в центре плана, т.е. при $x_i = \bar{x}_i$;

$\bar{y}_o$ -среднее арифметическое значение результатов всех повторных измерений в центре плана;

n - количество опытов в центре плана;

n-1 - количество степеней свободы для дисперсии воспроизводимости.

Оценка дисперсии адекватности вычисляется по формуле:

$$\sigma^2{}_{ад} = \frac{\sum_{i=1}^{N}(y_{i\,расч} - \bar{y}_{i\,эксп})^2}{N-m}$$

$y_{i\,расч}$ -значения отклика, полученные расчетным путем по уравнению модели (по аппроксимирующей функции);

$\bar{y}_{i\,эксп}$ - среднее значения функции отклика, полученное экспериментально в i -том опыте;

N -количество опытов;

m -количество членов в уравнении регрессии, включая a0;

N-m - количество степеней свободы для дисперсии адекватности.

Гипотеза об адекватности модели принимается в том случае, если рассчитанное значение F критерия не превышает табличного для выбранного

уровня доверительной вероятности и степеней свободы числителя и знаменателя, т.е. при условии

$$F_{расч} <= F_{табл}$$

Объект исследования: свойства объекта исследования реализованы программно. Математическая модель объекта имеет вид

$$y=a_0 + a_1z_1 + a_2z_2 + a_3z_3 + Random(k)$$

где z_1, z_2, z_3 могут иметь вид x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3
Уровень шума К задается программно. Величина К примерно соответствует величине шума в процентах от y_{max}.
Интерфейс пользователя приведен ниже на рис. 6

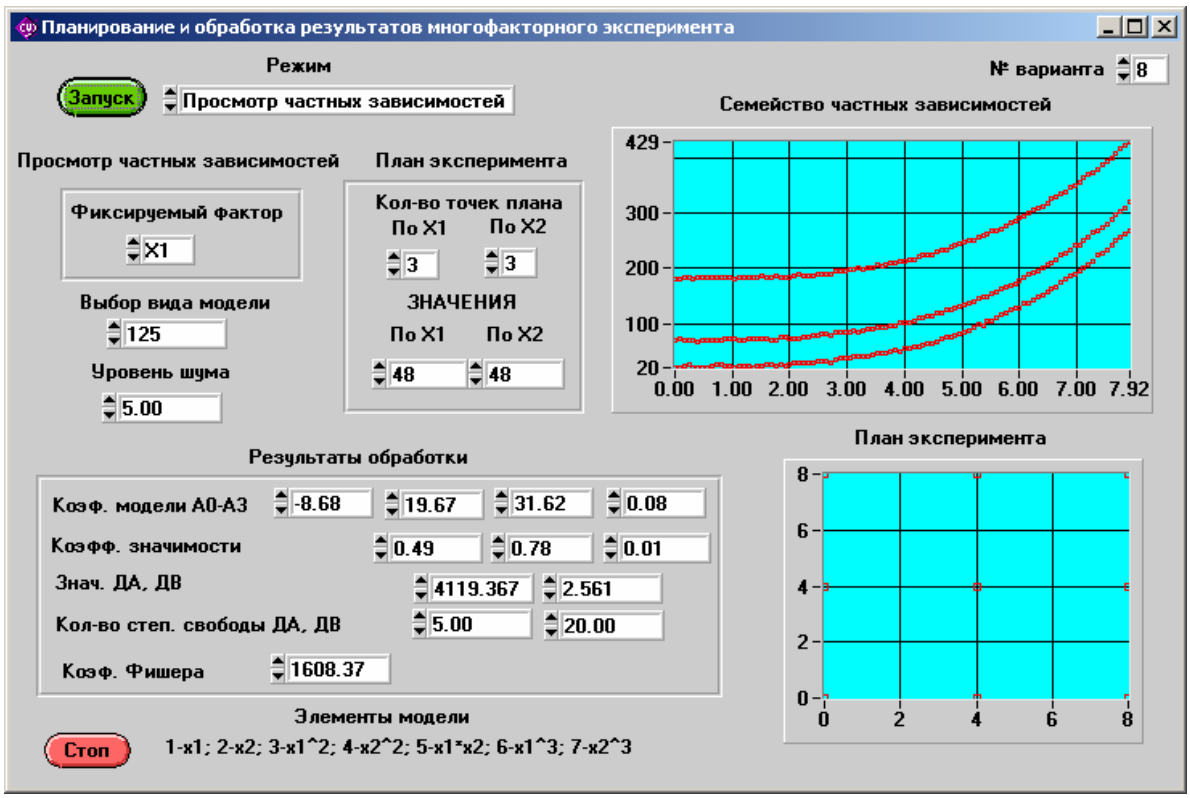


Рис. 6

Цель исследования: определить по экспериментальным данным вид модели (выполняющему лабораторную работу он неизвестен), произвести расчет численных значений коэффициентов, доказать адекватность уравнения регрессии экспериментальным данным, а также исследовать влияние плана эксперимента и уровня шума в экспериментальных данных на точность расчета коэффициентов уравнения регрессии.

Порядок работы

1. Задать № варианта и начальный уровень шума, равный 5.

2. Установить режим «Просмотр частных зависимостей» и запустить кнопкой «Запуск» программу демонстрации семейств частных зависимостей $y=f_1(x_1)$ при $x_2=const$, и $y=f_2(x_2)$ при $x_1=const$ и по виду кривых сделать предположение об общем виде уравнения регрессии (написать уравнение регрессии в общем виде).
3. Установить режим «Эксперимент» и задать количество точек по осям X1 и X2 при прямоугольном плане эксперимента с равномерным расположением точек. Исходить из того, что диапазон варьирования факторов x_1 и x_2 - от 0 до 8, т.е. при количестве точек по оси X1 или X2, равном трем, необходимые значения факторов 0, 4, 8. Количество точек по осям должно определяться с учетом предполагаемого вида модели, но минимальное количество точек по осям x_1 и x_2 - не менее 3;

Внимание! Использовать планы только 3x3 и 4x4.

Затем запустить кнопкой «Запуск» программу эксперимента. В результате будут вычислены значения отклика в заданных точках плана;

4. Установить режим «Обработка данных», задать вид предполагаемой модели и запустить программу обработки кнопкой «Запуск». В результате будут вычислены по МНК численные значения коэффициентов, коэффициенты значимости, значения дисперсий адекватности и воспроизводимости, количество степеней свободы дисперсий адекватности и воспроизводимости, коэффициент Фишера.
5. Установить адекватность или неадекватность выбранной модели, для чего:

а) проанализировать значения коэффициентов значимости. Если какой-то из коэффициентов значимости на порядок меньше остальных – значит соответствующий член из уравнения регрессии нужно исключить (и добавить вместо него другой, т.к. в данной работе все модели содержат 4 члена) и повторить п. 1-3;

б) сравнить полученное по программе значение коэффициента Фишера с табличным для уровня доверительной вероятности 0,95 и конкретных значений степеней свободы числителя и знаменателя и сделать вывод об адекватности или неадекватности модели. В случае обнаружения неадекватности - изменить вид модели (и, может быть, количество точек) и вернуться к п.1-3.

6. Доказать адекватность модели, полученной по п.3, путем проверки на адекватность других моделей, "похожих" на полученную в п.3. Для этого, выбирая последовательно несколько моделей, повторить п.1-3 и сравнить значения дисперсии адекватности. Окончательно выбрать модель, которая характеризуется наименьшим значением дисперсии адекватности.
7. Исследовать влияние уровня шума (случайных погрешностей) в экспериментальных данных на точность расчета коэффициентов уравнения регрессии. Для этого - повторить п.п. 1-3, задавая уровень шума больше и меньше исходного (например, 10 и 1). Сравнить показатели точности расчета коэффициентов уравнения регрессии.

8. Исследовать влияние плана эксперимента на точность расчета коэффициентов уравнения регрессии. Для этого повторить п.3 при неравномерном расположении точек плана по осям x_1 и x_2. Выяснить, можно ли получить более высокую точность.

Указания по выполнению работы

В процессе выполнения п.п. 1-8 фиксировать:
1. № варианта;
2. Системное время (час, мин, сек.);
3. Уровень шума;
4. Вид семейств частных зависимостей;
5. Предполагаемый вид модели;
6. Количество и числовые значения координат точек плана;
7. Результаты обработки:
   7.1. Коэффициенты модели;
   7.2. Коэффициенты значимости;
   7.3. Значения дисперсий адекватности и воспроизводимости;
   7.4. Рассчитанное и табличное значения коэффициента Фишера.

фиксировать в таблице:

Вид модели	**Коэф. модели**				**Коэф. значимости**			**Дисп. воспр.**	**Дисп. адекв.**	**$\frac{f_{числ}}{f_{знам}}$**	**Коэф. Фишера**	**Табл.зн. коэф. Фишера**
	A0	**A1**	**A2**	**A3**	**B1**	**B2**	**B3**					

Содержание отчета

I. Задание к работе.
II. Вид семейств частных зависимостей и определенный на основе их общий вид модели (моделей).
III. План эксперимента (экспериментов), проведенного для расчета численных значений коэффициентов модели и обоснование плана.
IV. Результаты обработки для проведенных экспериментов с использованием трех моделей:
   A. Найденные по МНК численные значения коэффициентов модели;
   B. Дисперсия адекватности;
   C. Дисперсия воспроизводимости;
   D. Степени свободы дисперсии адекватности и воспроизводимости;
   E. Коэффициент Фишера;
   F. Табличное значение коэффициента Фишера;
V. Выбор адекватной модели с объяснением.
VI. Результаты обработки по п. IV B-F для проведенных экспериментов при использовании найденной адекватной модели и различных уровнях шума.

VII. Планы экспериментов и результаты обработки по п.6 раздела «Порядок работы».
VIII. Выводы.

Приложение 1.

Таблица 1. Коэффициенты Фишера для Ф=0.95

fчисл fзнам	1	2	3	4	5	6	12	24	∞
1	164	199	216	225	230	234	245	249	254
2	18,5	19,2	19,2	19,3	19,3	19,3	19,4	19,5	19,5
3	10,1	9,6	9,3	9,1	9,0	8,9	8,7	8,6	8,5
4	7,7	6,9	6,6	6,4	6,3	6,2	5,9	5,8	5,6
5	6,6	5,8	5,4	5,2	5,1	5,0	4,7	4,5	4,4
6	6,0	5,1	4,8	4,5	4,4	4,3	4,0	3,8	3,7
12	4,8	3,9	3,5	3,3	3,1	3,0	2,7	2,5	2,3
24	4,3	3,4	3,0	2,8	2,6	2,5	2,2	2,0	1,7
∞	3,8	3,0	2,6	2,4	2,2	2,1	1,8	1,5	1,0

Тутыгин Владимир Семенович
ОСНОВЫ АВТОМАТИЗАЦИИ
ФИЗИЧЕСКОГО ЭКСПЕРИМЕНТА
Лабораторный практикум
Сводтемплан 2006г.
Лицензия ЛР № 020593 от 07.08.97
Налоговая льгота – Общероссийский классификатор продукции ОК 005-93, т.2; 95 3005 – учебная литература

Подписано в печать . Формат 60x84/16.
Усл. печ. л. 6,0. Уч. изд. л. 6,0 Тираж 100.

Отпечатано с готового оригинал макета, предоставленного составителем, в типографии Издательства СПбГПУ.
195251, Санкт-Петербург, Политехническая, 29.
Отпечатано на ризографе RN-2000FP
Поставщик оборудования – фирма «Р - ПРИНТ»
Телефон: (812) 110 – 65 – 09
Факс: (812) 315 – 23 - 04