Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра электронных приборов

ИЗМЕРЕНИЕ ПАРАМЕТРОВ ВАКУУМНОЙ СИСТЕМЫ НА ЭВМ В РЕАЛЬНОМ РЕЖИМЕ ВРЕМЕНИ

Методические указания к лабораторной работе
для студентов направлений «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

2012

Орликов, Леонид Николаевич

Измерение параметров вакуумной системы на ЭВМ в реальном режиме времени: методические указания к лабораторной работе для студентов направлений «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства» / Л. Н. Орликов; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Томский государственный университет систем управления и радиоэлектроники, Кафедра электронных приборов. - Томск : ТУСУР, 2012. - 33 с.

Цель работы: измерение параметров вакуумной системы на ЭВМ в реальном режиме времени

Предназначено для студентов очной и заочной форм, обучающихся по направлению «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства») по курсу «Спец. вопросы технологии»

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и
радиоэлектроники»

Кафедра электронных приборов

УТВЕРЖДАЮ
Зав.кафедрой ЭП
_____С.М. Шандаров
«___» _____ 2012 г.

ИЗМЕРЕНИЕ ПАРАМЕТРОВ ВАКУУМНОЙ СИСТЕМЫ НА ЭВМ В РЕАЛЬНОМ РЕЖИМЕ ВРЕМЕНИ

Методические указания к лабораторной работе
для студентов направлений «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

Разработчик
д-р техн. наук, проф.каф.ЭП
_____Л.Н.Орликов
_____2012 г

2012

Содержание

1. Языки пользователя для программирования электрофизических установок

При проведении технологических операций нет необходимости в применении больших ЭВМ. В основном требуются машины с возможностью выработки сигнала коммутации исполнительного устройства (например, машины фирмы MAKINTOSH). В настоящее время нашли применение, так называемые, языки пользователя [4-10]:

1) Ассемблер; Паскаль; Си, C^{++}, Java;
2) язык релейно-контактных символов;
3) язык «Время – команда»; «Время – параметр»;
4) язык булевых уравнений.
5) Язык SCPI
6) Язык программируемых контроллеров

1.1 Язык релейно-контактных символов

Для различных систем робототехники, систем автоматизированных станков и кузнечнопрессового оборудования широко применяется язык, основанный на аналогии включения или выключения контактов электрических схем. На рис. 1.1 представлены условные обозначения такого языка по данным фирмы MODICON.

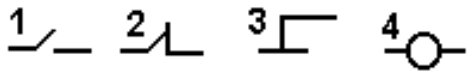


Рисунок 1.1 – Команды языка релейно–контактных символов. 1 – нормально разомкнутый; 2 – нормально замкнутый; 3 – вход в параллельную цепь; 4 – время.

При составлении программы технологических операций первоначально зарисовывают каналы технологических операций, в которых включены контакты. Сверху контакта проставляется порядковый номер операции, а снизу код операции по международному или отраслевому классификатору.

1.2 Язык Булевых уравнений

В ряде случаев для отдельного канала оказывается удобным одновременная запись того, что включено и что выключено. В таких случаях удобно пользоваться языком булевых уравнений с использованием операторов типа «и», «или», «не (инверсия)». На рис. 1.2 представлено соответствие булевых и релейных символов.

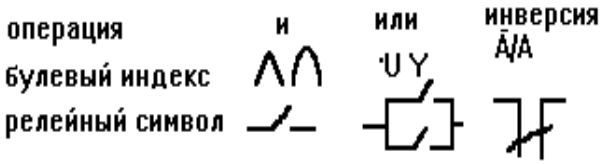


Рисунок 1.2 – Схема соответствия «булевых» и релейных символов

1.3 Языки типа: «Время – команда». «Время – параметр»

В ряде случаев подача команд на проведение последующей операции может производиться по истечении определенного времени или по достижении определенного параметра. Подобные реле времени рассчитаны на коммутацию канала в течение времен от секунд до недели и имеют до 48 каналов. Такие системы нашли широкое применение в электрофизических и электровакуумных установках под названием ОРИОН – 3. Система

ОРИОН – 3 предполагает контроль ввода сигналов, контроль хода процесса, управление процессом и индикацию параметров.

Система снабжена подсистемами управления типа ТЕХНОЛОГ и включает в себя технические нормали на материалы, инструменты, а также технические допуски на изготовление и базу данных кадрового обеспечения.

1.4 Язык управления типа «КАУТ»

Поиск наиболее универсального и простого языка общения с автоматикой привел к системе включающей контроль, анализ и управление технологией (КАУТ). Язык системы содержит три типа переменных:

1) целые числа;

2) напряжение (вольты);

3) время.

Язык снабжен ключевыми словами, вводимыми вначале программирования: (ТЕМП – температура, ДАВЛ – давление, НАГРЕВ, ТАЙМЕР, ВКЛ., ВЫКЛ., ЕСЛИ, ИНАЧЕ, КЛАПАН, ЖДАТЬ и др.). Каждой операции присваивается порядковый номер (кратный пяти для введения дополнительных команд). В начале программы записываются контролируемые величины, регулируемые, включаемые, а также наименование отслеживаемого параметра.

Пример. Программа для установки термического напыления материалов в вакууме.

Процессу напыления пленки предшествуют операции очистки материалов, их монтаж в вакуумной камере, откачка воздуха из камеры, прогрев всех деталей.

Первоначально составляется последовательность технологических операций. Каждой операции присваивается буква и номер. Присвоение букв производится по признакам подобия операций, а присвоение цифр – кратно пяти. Такое присвоение позволяет дополнительно включать промежуточные операции, возникающие при оптимизации технологического процесса. В нашем случае последовательность технологических операций для очистки изделий (индекс О), текущих операций (индекс М) и собственно напыления (индекс Н) выглядит в следующем виде:

(О5, О10, О15...), (М5, М10, М15, М20...), (Н5, Н10, Н15...).

Далее в программу записываются контролируемые, регулируемые, включаемые и выключаемые величины:

контролируемые: К 1 – давление; К 2 – температура;

регулируемые: Р1 – нагрев;

включаемые: В1 – клапан; В2 – откачка; В3 – нагрев;

параметры: П1 – температура; П2 – счетчик.

Пример конкретного исполнения

1. Включение клапана В1.

5. Включение откачки В2

10. ВКЛ. НАГРЕВ

15. ЖДАТЬ 150 (ждать 150 секунд пока нагреется)

20. М 10

25. ЕСЛИ К ТЕМП = 250, ПЕРЕХОД НА М 45 (напыление)

ИНАЧЕ М20 (нагрев) (Комментарий: если температура подложки достигнет 250 градусов, перейти на напыление, в противном случае продолжить нагрев).

В такой последовательности составляется вся программа. Справа программы возможны комментарии.

1.5 Язык программирования измерительной аппаратуры SCPI

Язык SCPI (Standard Commands for Programmable Instruments – Стандартные команды для программируемых приборов) – это язык приборных команд на основе стандартного кода ASCII международного стандарта IEEE-488, предназначенный для программирования измерительных приборов.

Команды языка SCPI имеют иерархическую структуру. Родственные команды сгруппированы в общем узле. В качестве примера показана часть подсистемы SENSE (СЧИТЫВАНИЕ).

SENSe: (считывание)
VOLTage: (вольты)
DC:RANGe {<range>/MINimum| MAXimum}VOLTage: (порядок считывания, минимум, максимум, вольты)
DC:RANGe? [MINimum| MAXimum] (порядок считывания)
FREQuency: (частота)
VOLTage: RANGe {<range>/MINimum| MAXimum} FREQuency:
VOLTage: RANGe? [MINimum| MAXimum]
DETector:
BANDwidth{3| 20| 200 | MINimum| MAXimum} DETector:
BANDwidth? [MINimum| MAXimum]
ZERO:
AUTO {OFF I ONCE| ON} ZERO:
AUTO?

Синтаксис командного языка показывает большинство команд в виде наборов прописных и строчных букв.

SENSe является ключевым словом команды. VOLTage и FREQuency – ключевыми словами второго уровня. DC и VOLTage – ключевыми словами третьего уровня. Ключевое слово предыдущего уровня от ключевого слова более низкого уровня разделяется двоеточием (:).

Например: VOLTage: DC: RANGe {<range>/MINimum| MAXimum} означает измерение напряжения; DC – постоянный ток; цифровой разряд показаний, в скобках указаны предел, минимальные и максимальные значения.

Командная строка заканчивается символом возврата каретки <cr> и символом новой строки <nl>.

Фигурные скобки ({}) заключают варианты параметров и с командной строкой не передаются.

Вертикальная черта (|)используется для разделения нескольких вариантов параметра командной строки.

Угловые скобки (<>) показывают, что пользователь должен указать значение заключенного в скобки параметра.

Двоеточие (:) разделяет команды от ключевого слова более низкого уровня.

Аппаратура автоматически программируется на измерения командой MEASure и для более точных измерений командой CONFigure.

Например:

MEASure: VOLTage: DC? {<range>| MIN| MAX| DEF}, {<resolution>| MIN| MAX| DEF}– производится измерение постоянного напряжения. Показания пересылаются в буфер вывода. DEF- означает автоматическое переключение пределов равное 51/2 разрядам

MEASure: VOLTage: AC? {<range>/MUST|MAX| DEF}– производится измерение переменного напряжения;

MEASure:RESistance? – производится измерение сопротивления резистора.

Среди различных простых команд наиболее распространенными являются:

AUTO- автоматическое переключение пределов;

BUS – по шине; OFF/ON- выкл/вкл;

FETCh- передача показаний из внутренней памяти прибора в буфер вывода и далее в контроллер.

1.6 Языки, для программирования логических контроллеров

Международной электротехнической комиссией рекомендованы некоторые стандартные языки для программирования контроллеров.

На рис.1.3 представлены языки для программирования контроллеров.

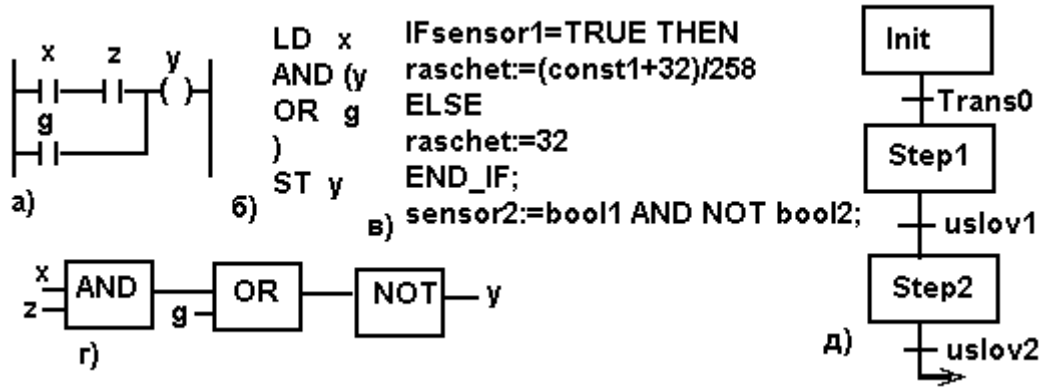


Рисунок 1.3 - Команды языков программирования контроллеров; а) – LD; б) – IL; в) – ST; г) – FBD; д) CFC

1.7 Язык LD

Язык LD по сути повторяет язык релейно-контактных символов. Собрав логическую последовательность, можно коммутировать входную и выходную шины, расположенные на экране.

Язык IL – это программирование на ассемблере, используемое всеми производителями.

Язык ST – это аналог языков C, Pascal.

Язык FBD – это язык функциональных блоков. Язык подобен языку релейно-контактных символов и удобен для специалистов, проектирующих системы на элементах логики.

Язык CFC – это язык функциональных блоков FBD, снабженный дополнительными командами выдержки времени или параметров перехода между операциями.

2 Экспериментальная часть

Измерение параметров вакуумной системы проводится с помощью мультиметра В7-78.

Перед началом работы следует изучить техническое описание и инструкцию по эксплуатации, а также ознакомиться с расположением и назначением органов управления и контроля на передней и задней панелях прибора.

Разместив прибор на рабочем столе, обеспечив удобство работы и условием естественной вентиляции.

Ручку прибора, возможно, использовать не только для переноски, но и как подставку, обеспечивающую необходимый угол, зафиксируйте на одном из доступных положений.

Проверить надежность заземления

Переключатель напряжения сети привести в соответствие с параметрами сети. Значение установленного напряжения питания отмечено стрелкой на корпусе держателя сетевого предохранителя.

Для изменения значения напряжения сетевого питания необходимо поддеть отверткой с плоским жалом держатель сетевого предохранителя, изъять его и установить на место таким образом, что бы значение напряжения питания совпало с указателем стрелки

При изменении напряжения питания требуется установка следующих значений сетевых предохранителей (Табл. 2.1)

Таблица 2.1 - Значения сетевых предохранителей для прибора

Установленное значение напряжения, В	Диапазон входных напряжений, В	Значение предохранителя	
100	90..110	0,2 А	
120	108..132		
220	198..240	0,1 А	
230	207..250		

Подсоединить шнур питания к сети. Тумблер «сеть» должен находиться в положении выключенном (Выкл).

Тумблер «сеть» устанавливается в положение «ВКЛ». Сразу после включения тумблера осуществляется автоматическая проверка функционирования прибора и исправности отображения дисплея, высвечиваются версия программного обеспечения и адрес GPIB (при установленной опции 1) и прибор устанавливает следующие режимы работы:

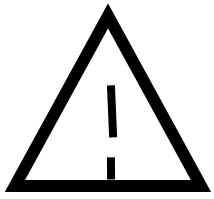
- измерение постоянного напряжения
- запуск – автоматический
- выбор пределов измерения – автоматический
- разрядность индикатора 4 1\2 (если в меню не установлено иное разрешение)

Для проведения работ необходимо прогреть прибор в течении 15 минут.

В случае, если органами управления, расположенными на передней панели, нельзя,

Перестроить параметры прибора (произошел сбой, а программе микро ЭВМ), достаточно выключить прибор и через 10-15 минут включить вновь.

2.1 Измерение напряжения



Предупреждение

В случае, когда неизвестна величина измеряемого напряжения, необходимо использовать режим автоматического выбора предела измерения и начинать измерение в режиме V.

Для реализации всех возможностей прибора используйте следующие функции:

- ручной или автоматический выбор пределов измерения. При проведении

большого числа однотипных измерений, вы можете сократить время измерения, зафиксировав предел измерения. Прибор не будет перебирать поддиапазоны измерения, а начнет измерения на выбранном пределе.

- циклический или разовый пуск измерения.
- изменения разрядности индикатора.
- относительные измерения
-допусковый контроль.
- определение минимальных, максимальных или средних значений.

Для измерения постоянного напряжения нажмите кнопку [**U---**]

На индикаторе появится надпись «**VCD**» с соответствующей размерностью (mV или V).

Для измерения переменного напряжения нажмите кнопку [**U~**]

На индикаторе появится надпись «**VAC**» с соответствующей размерностью (mV или V).

Произведите подключение источника напряжения к вольтметру, как показано на рис 1.4 (данная схема подключения при измерении напряжения, сопротивления по 2-х проводной схеме, частоты, периода звуковой прозвонки целостности цепи и тестирования полупроводниковых диодов). На рис.2.1 представлена схема измерения падения напряжения на резисторе.

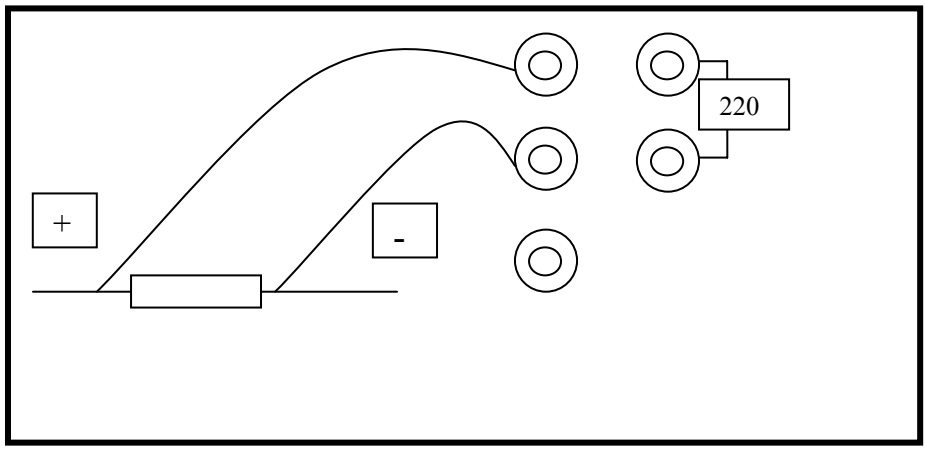


Рисунок 2.1– Схема измерения падения напряжения на резисторе

Произведите отсчет результата измерения.

2.2 Измерение тока

Для измерения постоянного тока нажмите последовательно [**ПРЕФ**] и [**U---**]

на индикаторе появится надпись «**ADC**» с соответствующей размерностью(mA или A)

Для измерения переменного тока нажмите последовательно [**ПРЕФ**] и [**U~**] на индикаторе появится надпись «**AAC**» с соответствующей размерностью(mA или A)

Произведите подключение источника напряжения к вольтметру, как показано на рис 1.5

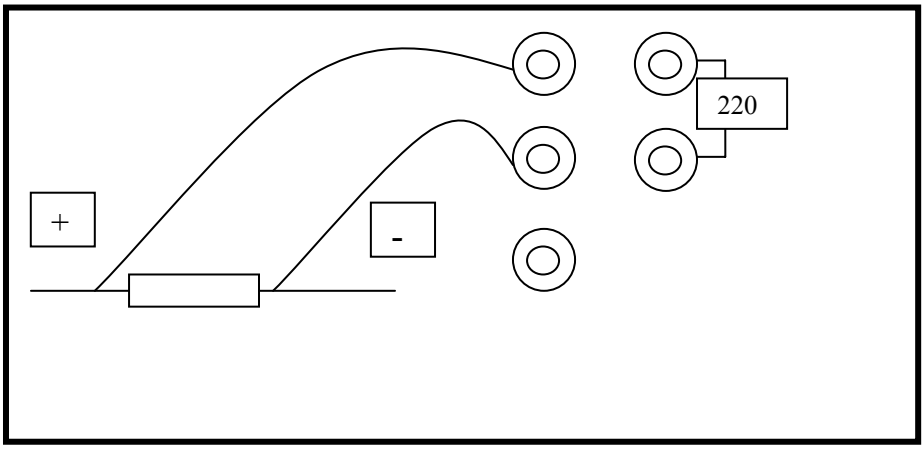


Рисунок 2.2 - Схема измерения тока

Для реализации всех возможностей прибора используйте следующие функции:

- ручной или автоматический выбор пределов измерения. При проведении большого числа однотипных измерений, вы можете сократить время измерения, зафиксировав предел измерения. Прибор не будет перебирать поддиапазоны измерения, а начнет измерения на выбранном пределе.
- циклический или разовый пуск измерения.
- изменения разрядности индикатора.
- относительные измерения
-допусковый контроль.
- определение минимальных, максимальных или средних значений.

Произведите отсчет измерения

2.3 Измерение сопротивления

ПРЕДУПРЕЖДЕНИЕ: измеряемая цепь предварительно должна быть отключена от источника питания. В случае, когда неизвестно присутствует ли на сопротивлении какое-либо напряжение, необходимо использовать режим автоматического выбора предела измерения и начинать измерение в режиме V/

При измерении сопротивления по 2-х проводной схеме произведите подключение согласно рис 2.3

При измерении сопротивления по 4-х проводной схеме произведите подключение согласно рис 2.3, при этом обратите внимание на правильное подключение токовых и потенциальных концов.

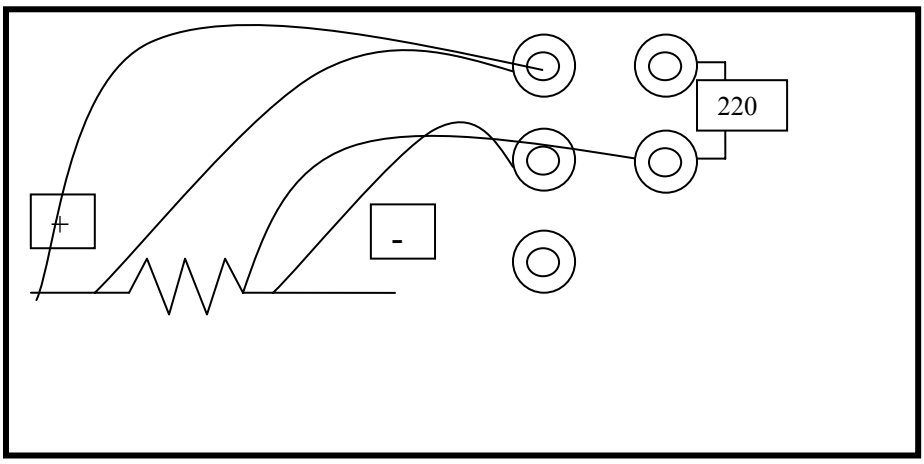


Рисунок 2.3 - Схема измерения сопротивления

Для измерения сопротивления по 2-х проводной схеме нажмите кнопку **[2прΩ].**
На индикаторе появится надпись «ОНМ».

Для измерения сопротивления по 4-х проводной схеме нажмите последовательно кнопки **[ПРЕФ]** и **[2прΩ]** ,индикаторе появится надпись «**ОНМ^4w**» **.**

Для реализации всех возможностей прибора используйте следующие функции:

- ручной или автоматический выбор пределов измерения. При проведении большого числа однотипных измерений,

вы можете сократить время измерения, зафиксировав предел измерения. Прибор не будет перебирать поддиапазоны

измерения, а начнет измерения на выбранном пределе.

- циклический или разовый пуск измерения.
- изменения разрядности индикатора.
- относительные измерения
-допусковый контроль.
- определение минимальных, максимальных или средних значений.

Произведите отсчет измерения

Примечание: сопротивление более 10Мом измеряются аналогично токовым по 2-х проводной схеме.

2.4 Измерение частоты и периода

ПРЕФ, HZ – Измерение частоты
ПРЕФ, U~ – измерение периода (на индикаторе появится надпись «**S**»)

Для измерения частоты нажмите кнопку **[ПРЕФ]**, на индикаторе появится надпись «**HZ**» с соответствующей размерностью(k).

Для измерения периода нажмите последовательно кнопки **[ПРЕФ]** и **[U ~]**,
на индикаторе появится надпись «**S**» с соответствующей размерностью (m или μ)
Произведите подключение источника напряжения к вольтметру.

2.5 Измерение температуры

Для измерения температуры в градусах Цельсия нажмите кнопку **[Темп]** на индикаторе появится надпись «◦**C**»

Для измерения температуры в градусах по шкале Фаренгейта нажми последовательно кнопки **[ПРЕФ]** и **[Темп]** на индикаторе появится надпись «◦**F**»

Подключить через адаптер датчик температуры (как показано на рис 2.4), термопару поместить в измеряемую среду.

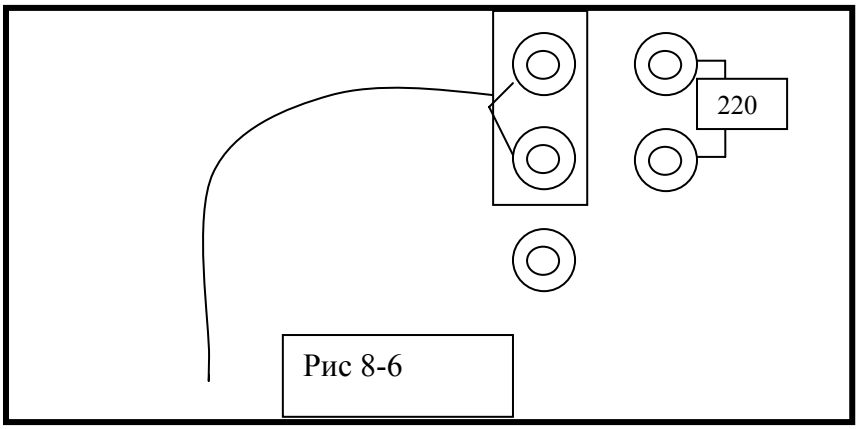


Рисунок 2.4 - Схема измерения температуры

Произведите отсчет результата измерения.

Примечание: при необходимости расширения вольтметра при измерении температуры (температуры жидкостей, гелей, поверхности и т.п.) используйте другие термопары типа «K»

2.6 Программное обеспечение

Запускаем программу GOTOWORK.EXE

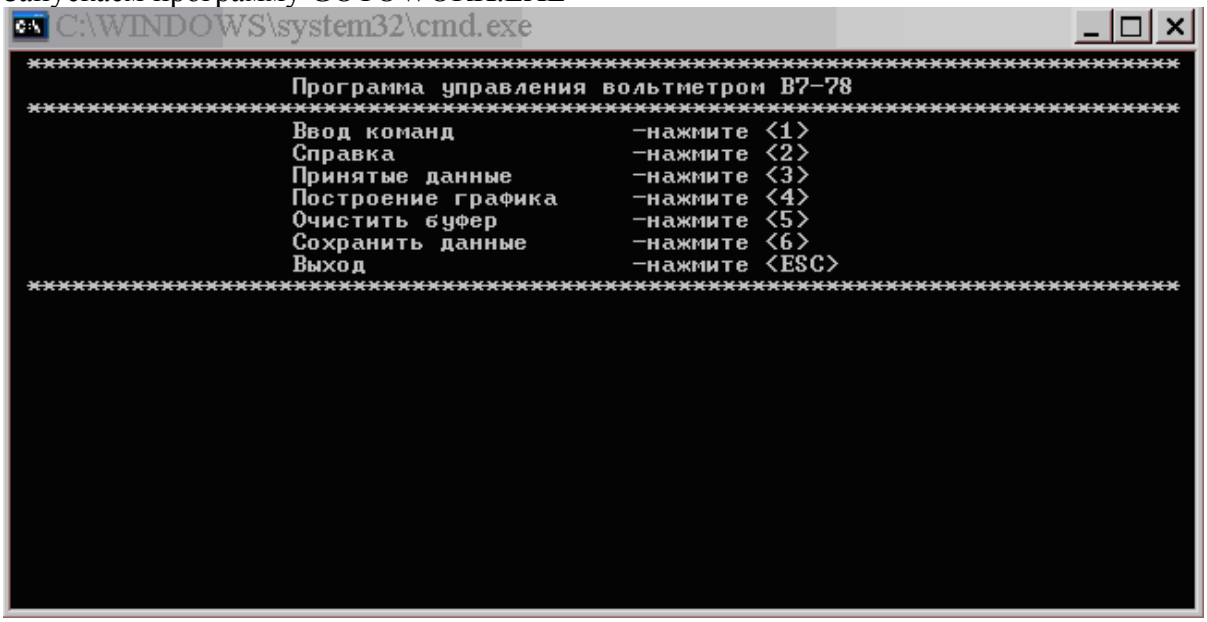


Рисунок 2.1 - Фрагмент запуска программы

Основные подмодули программы:
1 Подмодуль «ЗАДАНИЕ КОМАНД ИП»

Подмодуль «ЗАДАНИЕ КОМАНД ИП» предоставляет возможность оператору задать команду для конфигурирования ИП по интерфейсу RS-232. В логике модуля предусмотрено задание команд как в ручную оператором, так и программно.

Задание команд оператором заключается в вводе строки команды в меню и нажатии клавиши **<Enter>,** при запуске команды на исполнение. Команда оператора должна соответствовать формату команд SCPI. Далее программа сама разбивает команду на байты и выставляет данные на COM-порту, откуда они считываются ИП. Если не возникло никакой ошибки при передаче данных, то данные переносятся в память ИП и могут быть считаны. Для этого программа опрашивает COM-порт ЭВМ на наличие данных.

Программное задание команд предусматривает считывание значений постоянного напряжения и тока с максимальной частотой 2 показания в секунду. Смысл производимых программой операций заключается в следующем, вольтметр конфигурируется на работу в удаленном режиме управления командой «**SYST:REM**», затем в зависимости от того какого рода значения хочет получить оператор, производится посылка команды: «**CONF:VOLT:DC**» - для измерения напряжения, «**CONF:CURR:DC**» - тока. Далее оператору предоставляется возможность сконфигурировать процесс измерения, задав значения интервала между считываниями показаний и число отсчетов. Считывание данных в буфер вывода происходит по команде «**READ?**». Алгоритм работы подмодуля приведен на рисунке 2.2.

2 Подмодуль «ВЫЗОВ СПРАВКИ»

Подмодуль «ВЫЗОВ СПРАВКИ» осуществляет вывод на экран монитора справки по пользованию программой. Вывод справки на экран осуществляется процедурой **HelpMenu** реализованной в модуле INTERFAC. Листинг данного модуля приведен в Приложении Б.

3 Подмодуль «ВЫВОД ДАННЫХ НА ЭКРАН»

Подмодуль «ВЫВОД ДАННЫХ НА ЭКРАН» осуществляет удобный вывод данных на экран монитора полученных от ИП. Вывод осуществляется только при наличии хотя бы одного значения в буфере. Вывод осуществляется по 10 значений, затем оператору необходимо нажать клавишу **<Enter>** для просмотра следующих десяти. Реализация данной функции изложена в модуле MAIN.

4 Подмодуль «ВЫВОД ГРАФИКА НА ЭКРАН»

Подмодуль «ВЫВОД ГРАФИКА НА ЭКРАН» осуществляет построение графика по полученным значениям и вывод его на экран монитора. Все процедуры необходимые для вывода графика на экран реализованы в модуле PAINT. Для построения изображения на экране монитора необходимо наличие цветного дисплея и драйвера адаптера VGA. Полученное изображение имеет разрешение 640*480 пикселей. Максимальное число точек построения 512. Процедуры по выводу и построению графика реализованы в модуле PAINT. Листинг модуля PAINT приведен в Приложении В. На рисунке 2.2 изображена блок-схема данного подмодуля

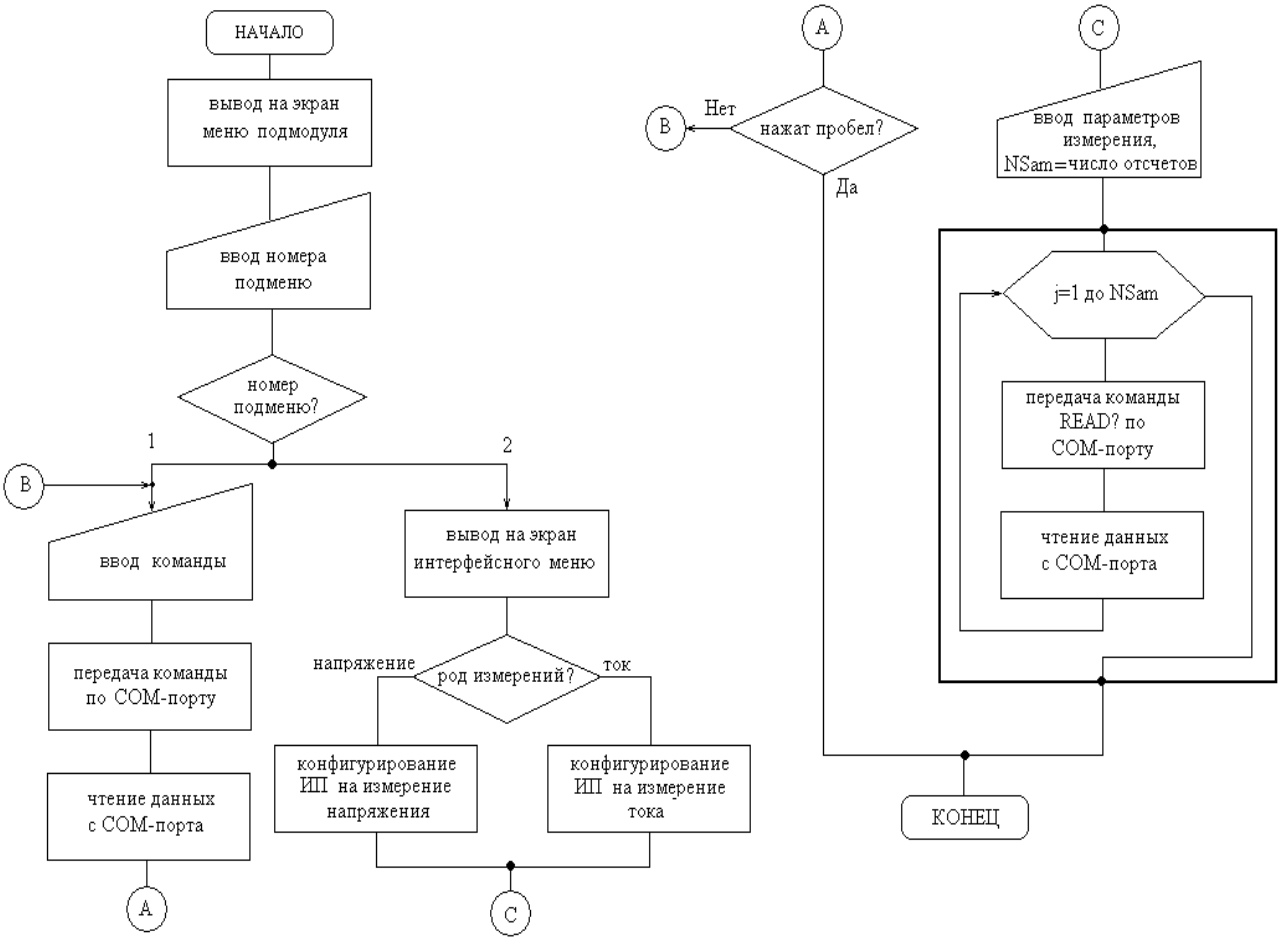


Рисунок 2.2 – Блок - схема работы подмодуля «ЗАДАНИЕ КОМАНД ИП»

5 Подмодуль «СОХРАНЕНИЕ ДАННЫХ»

Подмодуль «СОХРАНЕНИЕ ДАННЫХ» осуществляет сохранение данных находящихся в буфере на жесткий диск ЭВМ в файл указанный оператором. Ошибка сохранения может возникнуть, если указанного диска не существует, указанный диск переполнен или диск защищен от записи. При сохранении файла по умолчанию создается папка В7-78 в корневом каталоге указанного диска. Функцию сохранения данных осуществляет процедура **SaveFile** реализованная в модуле MAIN. На рисунке 2.3 изображена блок-схема данного подмодуля.

6 Подмодуль «ОЧИСТКА БУФЕРА»

Подмодуль «ОЧИТСКА БУФЕРА» осуществляет удаление данных из буфера и освобождение занятой памяти.

7 Подмодуль «ВЫХОД»

Подмодуль «ВЫХОД» осуществляет корректный выход из программы с освобождением занятой памяти для работы, закрытие COM-порта.

.

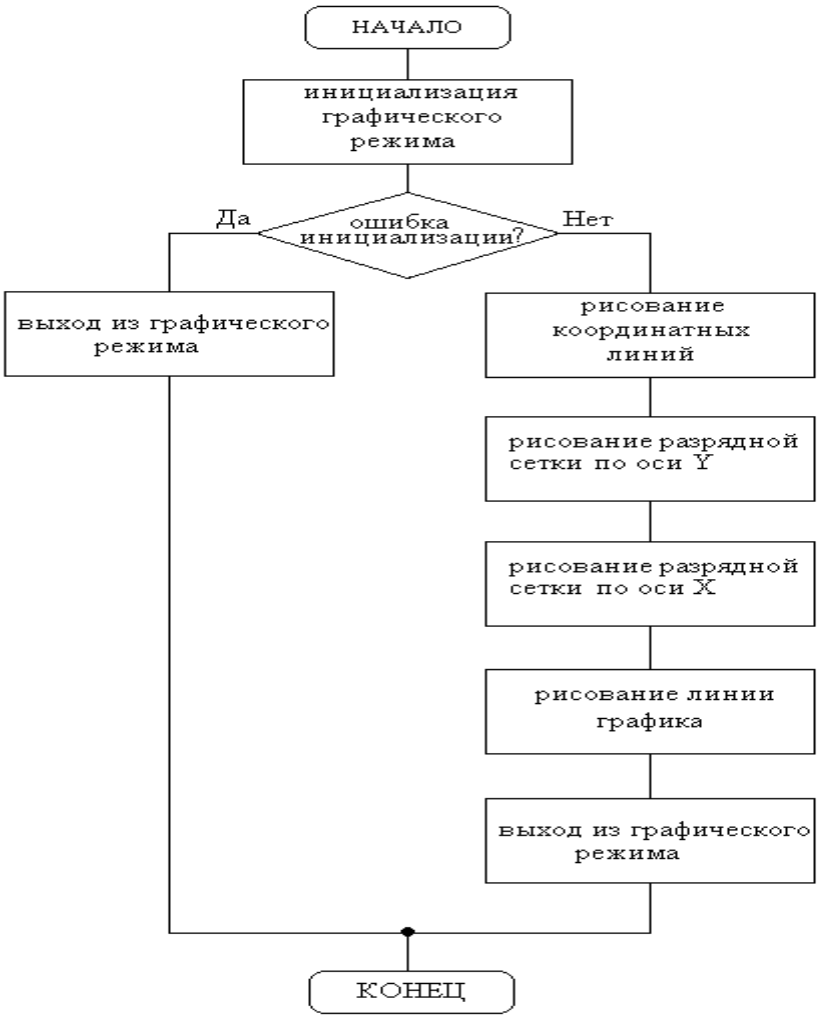


Рисунок 2.3 – Блок-схема подмодуля «ВЫВОД ГРАФИКА НА ЭКРАН»

3 Содержание отчета

1. При составлении отчета необходимо руководствоваться общими требованиями и правилами оформления отчета о лабораторной работе.

2. В соответствующих разделах отчета необходимо представить:

1) задание;

2) схему устройства;

3) таблицы экспериментальных данных;

4) результаты расчетов, предусмотренных заданием;

5) выводы;

6) листинг программы

Приложение А

```
Unit MAIN;

INTERFACE

{Использованные модули                                              }
Uses Crt,RS232Dos,Interfac, Dos, Paint;

{Запускает программный комплекс                                      }
Procedure RUN;

IMPLEMENTATION

 {Запускает программный комплекс                                    }
  Procedure RUN;
  Label lab1;

  {Размер буфера памяти                                             }
  Const  BuffSize=512;

  {Тип определяет строку и ее длину 80                              }
  Type
   Str80=String[80];
  {Тип определяет данные, которые хранит буфер и его размер     }
   ArStr80=Array [1..BuffSize]Of Str80;
   Str2=String[2];
   Str11=String[11];

  Var
   { DelayTime - Задержка времени }
   DelayTime:real;
   { ExitY       - условие "выход?"                                  }
   { HaveFolder  - условие "имеется папка?"                          }
   ExitY,HaveFolder : Boolean;
   { B           - принятый байт                                     }
   B : Byte;
   {TimeOut - delay }
   TimeOut:LongInt;
   { C           - клавиша выбора                                    }
   { Ch          - клавиша меню "Ввод команд"}
   C,Ch  : Char;
   {  I  - переменная счета вхождений в цикл                         }
   {  N  - число элементов сохраненных в буфере                      }
   {  P  - счетчик числа элементов сохраненных в буфере              }
   {  E  - код ошибки при переводе из буквенного формата в цифровой }
   {   Q   - число строк выводимых на экран при просмотре принятых
значений}
   {  CountE - число ошибок при получении данных}
   {  DelayTime - время задержки между замерами }
   {  CS - число замеров }
   I,J,N,P,E,Q,CountE,NSam,DT : Integer;
   { Buffer    - буфер принятых значений                             }
   Buffer : ^ArStr80;
   { Izmerenie - массив принятых значений                            }
   { Watch     - массив значений таймера                          }
   Izmerenie    : Mas;
   Watch        : ArStr20;


{*********************************************************************}
```

```
         { Читаем данные с COM1-порта ручной                                  }
         { L - счетчик числа вхождений в процедуру                            }
         { M - счетчик числа вхождений после получения байта                  }
        Procedure ReadFromComMan(K:Integer;var T:Integer);
        Begin
          {Увеличиваем счетчик буфера                                         }
          K:=K+1;
          {Заносим данные в буфер                                             }
          Buffer^[K]:='';
          While Not KeyPressed Do
             Begin
               {Ожидаем данные на порту                                       }
               If ReadData(B,2000) Then
                  Begin
                    {Заполняем буфер                                         }
                    If        (B<=126)        and        (B>=32)        then
Buffer^[K]:=Buffer^[K]+chr(B);
                  End;
             End;
          {Если очередное сообщение пустое, то не заносим его в буфер}
          If Buffer^[K]='' Then K:=K-1;
          {Сохраняем значения числа элементов буфера                          }
          T:=K;
        End;


{*********************************************************************}
         {      Читаем      данные      с      COM1-порта      автоматически
}
         { L - счетчик числа вхождений в процедуру                            }
         { M - счетчик числа вхождений после получения байта                  }
        Procedure ReadFromComAuto(L:Integer; Var M:Integer);
        Var Q:Integer;
        Begin
          L:=L+1;
          Buffer^[L]:='';
          {Ожидаем данные на порту                                  }
             For Q:=1 To 16 Do
                Begin
                  If ReadData(B,1500) Then
                     Begin
                       {Заполняем буфер                                      }
                       If        (B<=126)        And        (B>=32)        Then
Buffer^[L]:=Buffer^[L]+chr(B);
                     End;
                End;
          If L<>1 Then
           Begin
             If Buffer^[L]='' Then Buffer^[L]:=Buffer^[L-1];
           end;
          M:=L;
        End;

       {*********************************************************************}
          {Сохраняем данные в файл }
          {CountOfBuffer - число элементов сохранненых  в буфере             }
          {HaveF         - условие "данные сохранены?"                       }
          Procedure SaveFile(CountOfBuffer:Integer;var HaveF:Boolean);
          {F        - файловая переменная}
          {FileName - имя файла}
          {DiskName - диск на котором находится папка "B7-78"}
          {J        - переменная счета вхождений в цикл  }
          {IOR      - ошибка ввода/вывода}
          Var F:Text;
```

```
            FileName,DiskName,Commentary:String[30];
            Year,Mon,Day,Dow,Hour,Min,Sec,Msec:word;
            J,IOR:Integer;
          Begin
            {Задаем файл на диске, на котором будут храниться данные}
            Writeln(' Введите имя диска');Write(' ');
            Readln(DiskName);
            Writeln(' Введите имя файла');Write(' ');
            Readln(FileName) ;
            {$I-}
            MkDir(DiskName+':\B7-78');
            {$I+}
            IOR:=IOResult;
            {Если ошибка, то не сохраняем данные}
            If IOR<>0 Then
               If IOR<>5 Then
               Begin
                 {Устанавливаем условие "данные сохранены?"  }
                 HaveF:=False;
                 Writeln(' Ошибка при открытии файла для записи!');
                 {Чистим буфер}
                 While Keypressed Do Readkey;
                 {Выход из подпрограммы}
                 Exit;
               End
            {Если ошибки нет, то сохраняем}
            Else
               Begin
                 {Создаем и записываем файл }
                 Assign(F,DiskName+':\B7-78\'+FileName);
                 Rewrite(F);
                 {Комментарий}
                 Writeln(' Комментарий');
                 Write(' ');
                 Readln(Commentary);
                 {Устанвливаем условие}
                 HaveF:=True;
                 {Передаем в файл дополнительную информацию : время, дату,
комментарий}
                 GetTime(hour,min,sec,msec);
                 GetDate(year,mon,day,dow);
                 Writeln(F,' Date ',day:0,'/',mon:0,'/',year:0,'  ');
                 Writeln(F,' Time ',hour:0,':',min:0,':',sec:0,' ms: ',msec);
                 Writeln(F,' ');
                 Writeln(F,' Commentary:');
                 Writeln(F,' '+Commentary);
                 J:=0;
                 While J<(CountOfBuffer) Do
                    Begin
                      J:=J+1;
                      Writeln(F,' ',J:3,':  ', Buffer^[J]);
                    End;
                 Close(F);
               End;
          End;

{*********************************************************************}
       Begin
         {Инициализируем буфер}
         New(Buffer);
         {Задаем входные значения перменных}
         C:=#0;
         N:=0;
         P:=0;
```

```
        ExitY:=true;
        {Выводим главное меню}
        Menu;
        {Основная программа}
        While ExitY Do
           Begin
             C:=Readkey;
             Case C of
        #49:
            Begin
              {Вывод меню выбора режима задания команды}
              Menu;
              CommEnterMenu;
              Ch:=readkey;
              If (Ch=#49) or (Ch=#50) Then
                   Begin
                     If Ch=#49 Then
                        Begin
                          Ch:=#0;
                          While Ch<>#32 Do
                             Begin
                               Menu;
                               {Инициализация порта}
                               InitCOM(1, 9600, 7);
                               {Введение командной строки}
                               CommandLine(Command);
                               {Преобразуем команду из строки в массив скан-
кодов клавиш}
                               Writeln(' Выход из меню <Пробел> ');
                               StrToByte(Command,CmdByte);
                               {Передаем байты}
                               For I:=1 To Length(Command) Do
                                  Begin
                                    WriteData(CmdByte[i], 3000);
                                  End;
                               {Передаем дополнительный байт конца команды}
                               WriteData($0A,3000);
                               Delay(1000);
                               {Записываем даннные в буфер}
                               ReadFromComMan(N,P);
                               {Запоминаем   значение   счетчика   элементов
буфера}
                               N:=P;
                               Str(P,Watch[P]);
                               Ch:=ReadKey;
                             End;
                        End;
                      If Ch=#50 Then
                        Begin
                     Lab1: Dispose(Buffer);
                           New(Buffer);
                           N:=0;
                           P:=0;
                           {Инициализация порта}
                           InitCOM(1, 9600, 7);
                           Menu;
                           CommMeasMenu;
                           Ch:=Readkey;
                           Command:='SYST:REM';
                           {Преобразуем команду из строки в массив скан-кодов
клавиш}
                           StrToByte(Command,CmdByte);
                           {Передаем байты}
                           For I:=1 To Length(Command) Do
```

```
                    Begin
                      WriteData(CmdByte[i], 3000);
                    End;
                  {Передаем дополнительный байт конца команды}
                  WriteData($0A,3000);
                  If (Ch=#49) Or (Ch=#50) Then
                    Begin

                      If Ch=#49 Then
                        Begin
                          {  Перевод  ИП  в  режим  -  измерение
напряжения}
                          Command:='CONF:VOLT:DC';
                          Menu;
                          Writeln(' ИЗМЕРЕНИЕ НАПРЯЖЕНИЯ');
                          Writeln('');
                        End;
                      If Ch=#50 Then
                        Begin
                         { Перевод ИП в режим - имзмерение тока}
                          Command:='CONF:VOLT:AC';
                          Menu;
                          Writeln(' ИЗМЕРЕНИЕ ТОКА');
                          Writeln('');
                        End;
                    Writeln('Введите время задержки между замерами в
секундах ');
                      {$I-}
                      Write(' ');Readln(DelayTime);
                      {$I+}
                      If IOresult<>0 Then GoTo Lab1;
                      Writeln(' Введите число замеров (не более 512)
');
                      {$I-}
                      Write(' ');Readln(NSam);
                      {$I+}
                      If IOResult<>0 Then GoTo Lab1;
                      {Преобразуем команду из строки в массив скан-
кодов клавиш}
                      StrToByte(Command,CmdByte);
                      {Передаем байты}
                      For I:=1 To Length(Command) Do
                        Begin
                          WriteData(CmdByte[I], 3000);
                        End;
                      {Передаем дополнительный байт конца команды}
                      WriteData($0A,3000);
                      Command:='Read?';
                      StrToByte(Command,CmdByte);
                      Writeln(' Ждите... ');
                      For J:=1 To NSam Do
                        Begin
                          {Передаем байты}
                          For I:=1 To Length(Command) Do
                            Begin
                              WriteData(CmdByte[i],3000);
                            End;
                          {Передаем   дополнительный   байт   конца
команды}
                          WriteData($0A,3000);
                          {Задержка}
                          If DelayTime >= 1 Then
                            Begin
                              DT:=Round(DelayTime);
```

```
                                        For I:=1 To DT Do
                                           Begin
                                             For  TimeOut:=  1  To  125000000
Do;
                                           End
                                      End
                                   Else
                                      Begin
                                        DT:=Round(100*DelayTime/2) Div 100;
                                        For  TimeOut:=1  To  Round(125000000/2)
Do;
                                      End;
                                   End;
                                For J:=1 To NSam Do
                                  Begin
                                    {Записываем даннные в буфер}
                                    ReadFromComAuto(N,P);
                                    {Запоминаем значение счетчика элементов
буфера}
                                    N:=P;
                                    Delay(3000);
                                  End;
                                {Значения таймера }
                                For J:=1 To N Do Str(DT*J,Watch[J]);
                             End;
                       End;
                  End;
            Writeln(' Нажмите клавишу меню...');
            {Чистим буфер клавиатуры, задержка}
            While Keypressed Do Readkey;
          End;

       #50:
          Begin
            {Выводим меню "СПРАВКА"}
            Menu;
            HelpMenu;
          End;

       #51:
          Begin
            {Выводим меню "ПРИНЯТЫЕ ДАННЫЕ"}
            N:=P;
            Q:=0;
            Menu;
            WriteLn(' ПРИНЯТЫЕ ДАННЫЕ ОТ ВОЛЬТМЕТРА: ');
            {Выводим данные с шагом 10 значений }
            If N=0 Then Writeln(' В буфере нет данных! ')
            Else
               For I:=1 to N Do
                  Begin
                    Q:=Q+1;
                    Writeln(' ',I,' : ',Buffer^[I]);
                    If Q=10 Then
                       Begin
                         Q:=0; Writeln(' <Enter>');Readln;
                       End;
                  End;
            Writeln(' Нажмите клавишу меню...')
          End;

       #52:
          Begin
            {Выводим меню "ПОСТРОЕНИЕ ГРАФИКА"}
```

```
            N:=P;
            Menu;
            E:=3;
            CountE:=0;
            {Если число точек в диапозоне допустимых значений, то...}
            If (N>0) And (N<=512) Then
            Begin
            {Преобразуем принятые байты из строчного формата в численный }
               For I:=1 To N Do
                  Begin
                    Val(Buffer^[I],Izmerenie[I],E);
                    If E<>0 Then CountE:=CountE+1;
                  End;
               If CountE>0 Then
                  Begin
                    Writeln(' Ошибка в входных данных!');
                    Writeln(' График может содержать неверные данные!');
                    Writeln(' Подождите...');
                    For I:=1 To 90 Do
                       Begin
                         Delay(5000);
                       End;
                  End;
               {Если нет ошибки при преобразовании в численный формат,
то...}
                  Begin
                    {Выводим график}
                    Graphic(Izmerenie,N,Watch);
                    Readkey;
                    CloseGr;
                    Menu;
                  End;
            End
            Else Writeln(' Число точек построения выходит за предел!');
            Writeln(' Нажмите клавишу меню...');
          End;

        #53:Begin
            {Выводим меню "ОЧИСТИТЬ БУФЕР"}
            N:=P;
            Menu;
            Write(' ОЧИСТИТЬ БУФЕР? ');
            Writeln(' Да[Y]/Нет[Any] ');
            {Если нажата клавиша "Да", то... }
            Begin
              C:=Readkey;
              If (C='Y') Or (C='y') Then
                 Begin
                   {Выходим из меню, очищая буфер }
                   Dispose(Buffer);
                   New(Buffer);
                   N:=0;
                   P:=0;
                   Writeln(' Буфер очищен!');
                   Writeln(' Нажмите клавишу меню...');
                 End
              Else
                 Begin
                   {Выходим из меню, не очищая буфер}
                   Writeln(' Буфер не очишен!' );
                   Writeln(' Нажмите клавишу меню...');
                 End;
            End;
          End;
```

```
 #54:Begin
       {Выводим меню "СОХРАНИТЬ ДАННЫЕ"}
       N:=P;
       Menu;
       Write(' СОХРАНИТЬ ДАННЫЕ НАХОДЯЩИЕСЯ В БУФЕРЕ? ');
       Writeln(' Да[Y]/Нет[Any Key] ');
       Begin
         {Если нажата клавиша "Да", то... }
         C:=Readkey;
         If (C='Y') Or (C='y') Then
            Begin
              {Выходим из меню, очищая буфер }
              SaveFile(N,HaveFolder);
              If HaveFolder Then Writeln(' Данные сохранены!')
              Else Writeln(' Данные не сохраннены! ');
              Writeln(' Нажмите клавишу меню...');
            End
         {Если нажата клавиша "Нет", то... }
         Else
            Begin
              {Выходим из меню,не очищая буфер }
              Writeln(' Данные не сохранены! ');
              Writeln(' Нажмите клавишу меню...');
            End;
       End;
     End;

 #27:Begin
       {Выводим меню "ВЫХОД" }
       N:=P;
       Menu;
       Write(' ВЫХОД?');
       Writeln(' Да[Y]/Нет[Any Key] ');
       {Если нажата клавиша "Да", то... }
       C:=Readkey;
       If (C='Y') Or (C='y') Then
          Begin
            {Очищаем буфер}
            Dispose(Buffer);
            {Закрываем COM-порт}
            CloseCOM;
            {Выполняем условие "ВЫХОД?"}
            ExitY:=False;
          End
       {Если нажата клавиша "Нет", то... }
       Else
          Begin
            {Выводим сообщение}
            Writeln(' Нажмите клавишу меню...');
          End;
     End;
   End;
 End;
End;
End.
```

Приложение Б

```
UNIT INTERFAC;

INTERFACE

Uses CRT,DOS;

     {MaxCommand - максимальная длина команды - 80 символов}
     {BuffSize   - размер буфера}

Const MaxCommand=80;
      BuffSize=512;

     {MasByte    - массив ASCII кодов}
Type MasByte=Array [1..MaxCommand] Of Byte;
     Str80=String[80];
     Str5=String[5];

     {Command    - командая строка}
     {CmdByte  -     хранит  ASCII  коды  клавишь,  нажатых  при  задании
команды}
Var  Command:Str80;
     CmdByte:MasByte;


Procedure CommandLine (Var Command:Str80);
Procedure StrToByte(Command:Str80;Var CmdByte:MasByte);
Procedure Menu;
Procedure HelpMenu;
Procedure CommEnterMenu;
Procedure CommMeasMenu;

                                IMPLEMENTATION
Procedure CommEnterMenu;
Begin
  Writeln(' ВВОД КОМАНД ВОЛЬТМЕТРУ');
  Writeln('');
  Writeln(' Ввод команд оператором             - нажмите <1>');
  Writeln(' Автоматизированное снятие показаний - нажмите <2>');
End;

Procedure CommMeasMenu;
Begin
  Writeln(' АВТОМАТИЗИРОВАННОЕ СНЯТИЕ ПОКАЗАНИЙ');
  Writeln(' ');
  Writeln(' Измерение напряжения - нажмите <1>  ');
  Writeln(' Измерение тока       - нажмите <2>  ');
End;

{Рисуем меню справки}
Procedure HelpMenu;
Begin
  Writeln('                                СПРАВКА        ');
  Writeln('  ');
  Writeln('    * Ввод команд: ' );
  Writeln('  Ввод  команд  осуществляется  оператором  при  помощи  языка
программирования SCPI.');
  Writeln('  Максимальная  длина  команды  80  символов.Ввод  команды
оканчивается нажатием');
  Writeln(' клавиши <Enter>.');
  Writeln('    * Сохранение данных: ' );
```

```
        Writeln(' Сохранение данных происходит по умолчанию в папку B7-78,
которая создается ');
        Writeln(' на указанном оператором диске.');
        Writeln('    * Вывод на экран:');
        Writeln(' При выходе из меню "Построение графика" необходимо нажать
<ALT>+<Enter>.');
        Writeln(' Нажмите клавишу меню...');
      End;

      {Рисуем основное меню}
      Procedure Menu;
      Begin
        Clrscr;

Writeln('*********************************************************************')
;
        Writeln('                    Программа управления вольтметром B7-78');

Writeln('*********************************************************************')
;
        Writeln('                    Ввод команд            -нажмите <1>   ');
        Writeln('                    Справка                -нажмите <2>   ');
        Writeln('                    Принятые данные        -нажмите <3>   ');
        Writeln('                    Построение графика     -нажмите <4>   ');
        Writeln('                    Очистить буфер         -нажмите <5>   ');
        Writeln('                    Сохранить данные       -нажмите <6>   ');
        Writeln('                    Выход                  -нажмите <ESC> ');

Writeln('*********************************************************************')
;
        Writeln(' ');
      End;

      {Задаем команду}
      Procedure CommandLine(Var Command:Str80);
      Begin
        Writeln(' ВВЕДИТЕ КОМАНДУ ');
        Write(' : '); Readln(Command);
      End;

      {Переработка строчной команды в байтовый формат }
      Procedure StrToByte(Command:Str80; Var CmdByte:MasByte);
      Var Ch:Char;
          I:Byte;
      Begin
        For I:=1 To Length(Command) Do
           Begin
             Ch:=Command[I];
             CmdByte[I]:=Ord(Ch);
           End;
      End;
      End.
```

Приложение В

```
{                      Модуль Paint                          }
{Реализует набор функций необходимых для построения графика}
UNIT PAINT;

INTERFACE
{ Использованные модули  }
Uses Graph;

{ BuffSize - размер буфера                    }
{ MaxX - размер поля вывода графика по оси X }
{ MaxY - размер поля вывода графика по оси Y  }
Const BuffSize=512;
      MaxX=580;
      MaxY=430;
{ Mas     - тип определяет массив значений буфера числового формата}
{ Str15   - тип определяет строку в 15 символов}
{ ArStr15 - тип определяет массив значений времени   }
Type  Mas=Array[1..BuffSize] Of Real;
      Str15=String[15];
      ArStr15=array [1..BuffSize] Of Str15;

{ Graphic - процедура выводит график на экран          }
{ Sample - массив значений находящихся в буфере        }
{ NN     - счетчик числа значений находящихся в буфере }
{ TimeX  - массив значений времени }
Procedure Graphic(Sample:Mas;NN:Integer;TimeX:ArStr15);

{ CloseGr - закрывает графический режим вывода на экран}
Procedure CloseGr;

                                IMPLEMENTATION

{Инициализация графического режима}
Procedure InitGr;
Var Error,Driver,Mode:integer;
Begin
  Driver:=VGA;
  Mode:=2;
  InitGraph(Driver,Mode,'');
  Error:=GraphResult;
  If Error<>0 Then
     Begin
       Writeln(GraphErrorMsg(Error));
       CloseGraph;
     End
  Else
     Begin
       Writeln(' GraphDriver - VGA, GraphMode - 2 ');
     End;
End;

{Закрытие графического режима}
Procedure CloseGr;
Begin
  CloseGraph;
End;

{Рисует линию}
Procedure Line(X1,Y1,X2,Y2:Integer);
Begin
  MoveTo(x1,y1);
```

```
  LineTo(x2,y2);
End;

{Рисует координатные линии}
Procedure KrdLine;
Begin
  Line(50,5,50,475);
  Line(5,450,630,450);
End;

{Очищает экран}
Procedure Clear;
Begin
  ClearDevice;
End;


{Рисование графика}
Procedure Graphic(Sample:Mas;NN:Integer;TimeX:ArStr15);
 {Рисует координаты по оси X}
 Procedure PosX;
 Var Px,I,Div2: Integer;
     PointX,PointStartX :Integer;
 Begin
   Div2:=0;
   OutTextXY(610,452,TimeX[NN]);
   If (NN>=0) and (NN<16) Then PointStartX:=1;
   If (NN>=16) and (NN<=512) Then PointStartX:=Round(NN/15);
   If NN>512 Then PointStartX:=Round(NN/15);
   PointX:=PointStartX;
   For I:=1 To NN Do
      Begin
        If NN<16 Then
           Begin
             If PointX=I Then
                Begin
                  Div2:=Div2+1;
                  Px:=50+(I-1)*Round(MaxX Div NN);
                  Line(Px,451,Px,449);
                  If   (Div2   mod   2)   =   0   Then   OutTextXY(Px-
10,460,TimeX[I])
                  Else OutTextXY(Px-5,470,TimeX[I]);
                  PointX:=PointStartX+PointX;
                End
           End
         Else
         If (NN>=16)And(NN<=512)  Then
            Begin
              If I=PointX Then
                 Begin
                   Div2:=Div2+1;
                   Px:=50+(I-1)*(MaxX Div NN);
                   Line(Px,451,Px,449);
                   If (Div2 mod 2) = 0 Then OutTextXY(Px-10,460,TimeX[I])
                   Else OutTextXY(Px-5,470,TimeX[I]);
                   PointX:=PointStartX+PointX;
                 End
            End
        End;
      End;

 {Рисование кординат по оси Y}
 Procedure PosY(W:real;Mx:real);
 Var Py:Integer;
```

```
     S :String[5];
 Begin
   If Mx<0.9 Then
      Begin
        W:=W*1000;
        Mx:=Mx*1000;
        OutTextXY(33,2,'-3');
        OutTextXY(5,10,'* 10');
      End;
   If (W>=0) And (W<0.9) Then Str(W:2:3,S);
   If (W>=0.9) And (W<10) Then Str(W:1:4,S);
   If (W>=10) And (W<100) Then  Str(W:2:3,S);
   If (W>=100) And (W<1000) Then  Str(W:3:2,S);
   If (W>=1000) And (W<10000) Then  Str(W:4:1,S);
   If  W>=10000 Then Str(W:5:0,S);
   Py:=Round(450-(MaxY*W/Mx)+1);
   OutTextXY(2,Py,S);
   Line(48,Py,52,Py);
 End;

 {Основная программа}
Var I,Kx,Ky:Integer;
   Max,U:real;
Begin
  InitGr;
  KrdLine;
  Kx:=0;Ky:=0;
  Max:=Sample[1];
  For I:=2 To NN Do
     Begin
       If Sample[I]>Max Then Max:=Sample[I];
     End;
  PosX;
  For I:=0 To 19 Do
     Begin
       U:=I*(Max/20);
       PosY(U,Max);
     End;
  PosY(Max,Max);
  For I:=1 To NN Do
     Begin
       Kx:=50+(I-1)*Round(MaxX Div NN);
       If Round(MaxY*Sample[I]/Max)<10 Then
          Ky:=450-Round(100*(MaxY*Sample[I]/Max)) Div 100
       Else Ky:=450-Round(MaxY*Sample[I]/Max);
       PutPixel(Kx,Ky,15);
       If I>=2 Then LineTo(Kx,Ky);
       MoveTo(Kx,Ky);
     End;
  End;
End.
```

Приложение Г

```
Unit RS232Dos;

INTERFACE

{$N+}

{Тип для преобразования последовательности байт}
{в число типа Single и обратно                 }
Type UFloat = packed record
  b : Array [0..3] of Byte;
 End;

{Инициализация порта с номером ComIndex                     }
{ ComIndex - номер порта                                    }
{ Speed     - скорость в бод                                }
{ Params   - конфигурация порта, согласно формату битов LCR }
{Возвращает False, если порт не обнаружен                   }
Function  InitCOM(ComIndex : Byte; Speed : Longint; Params : Byte) :
Boolean;

{Чтение байта с порта с тайм-аутом                          }
{ B    - прочитанный байт                                   }
{ Wait - время ожидания байта                               }
{Возвращает True, если байт реально прочитан                }
Function  ReadData(var B : Byte; Wait : LongInt): Boolean;
{Запись байта в порта с тайм-аутом                        }
{ B    - передаваемый байт                                }
{ Wait - время ожидания готовности передатчика            }
{Возвращает True, если байт передан                         }
Function  WriteData(B : Byte; Wait : LongInt): Boolean;


Function  CheckReadData(ChB : Byte; Wait : LongInt) : Boolean;
Function  ReadSingle(var S : Single; WD : LongInt) : Boolean;
Procedure CloseCOM;

IMPLEMENTATION


 Var { Будет хранить базовый адрес порта }
 BaseAdr : Word;

 { Возвращает базовый адрес порта с номером PortIndex }
 Function GetBaseAdr(PortIndex : Byte) : Word;
 Var LowAdr : Word;
 Begin
  { вычисляем младшую часть адреса в таблице }
  LowAdr := (PortIndex-1)*2;
  { получаем базовый адрес порта из таблицы }
  GetBaseAdr:= MemW[$0040:LowAdr];
 End;

 {Инициализация порта}
 Function  InitCOM(ComIndex : Byte; Speed : Longint; Params : Byte) :
Boolean;
 Var Freq : Word; FreqH, FreqL : Byte;
 Begin
  Freq := 115200 div Speed;
  FreqH:= Freq shr 8;
  FreqL:= Freq and $00FF;

  InitCOM:= True;
```

```
 { Вычисляем базовый адрес порта }
 BaseAdr:= GetBaseAdr(ComIndex);
 If BaseAdr = 0 Then Begin
  WriteLn('Порт ', ComIndex,' не обнаружен!');
  InitCOM:= False; {вернем ошибку}
  Exit;
 End;
 {Адресуем делитель порта с помощью установки DLAB=1}
 Port[BaseAdr+3]:= $80;
 {Устанавливаем младшую часть делителя}
 Port[BaseAdr+0]:= FreqL;
 {Устанавливаем старшую часть делителя}
 Port[BaseAdr+1]:= FreqH;
 {Сбрасываем DLAB и прописываем конфигурацию}
 Port[BaseAdr+3]:= Params;
End;

{Определение готовности}
Function GetSR : Boolean; assembler;
Asm
 Mov Dx, BaseAdr
 Add Dx, 5
 In  Al, Dx
 And Al, 20H
End;


{Чтение байта с порта}
Function ReadCOM : Byte; assembler;
Asm
 Mov Dx, BaseAdr
 In  Al, Dx
End;

{Передача байта в порт}
Procedure WriteCOM(B : Byte); assembler;
Asm
 Mov Dx, BaseAdr
 Mov Al, B
 Out Dx, Al
End;

{ Возвращает бит DR порта LSR }
Function GetDR : Boolean; assembler;
Asm
 Mov Dx, BaseAdr
 Add Dx, 5
 In  Al, Dx
 And Al, 1
End;

{Возвращает биты ошибок порта LSR          }
{ Бит 4 - обрыв линии                      }
{ Бит 3 - ошибка кадра (неверный стоп-бит) }
{ Бит 2 - ошибка четности                  }
{ Бит 1 - переполнение (потеря символа)    }
Function GetErr : Byte; assembler;
Asm
 Mov Dx, BaseAdr
 Add Dx, 5
 In  Al, Dx
 And Al, 1EH {0001 1110b}
End;
```

```
{Чтение байта с порта с ожиданием готовности порта}
Function ReadData(var B : Byte; Wait : LongInt): Boolean;
Var Result : Boolean; w : LongInt;
Begin
 w:= 0; Result:= False;
 For w:= 1 to Wait do begin {ожидание...}
  If GetDR then begin {проверяем доступность данных }
   B:= ReadCOM; {читаем байт с порта}
   Result:= (GetErr = 0); {проверяем код ошибки}
   Break;
  End;
 End;
 ReadData:= Result;
End;

{Возвращает true, если байт прочитан и равен ChB }
Function CheckReadData(ChB : Byte; Wait : LongInt) : Boolean;
Var B : Byte;
Begin
 CheckReadData:= False;
 If ReadData(B, Wait) then begin
  CheckReadData:= (B = ChB);
 End;
End;

{Передает байт в порт с ожиданием готовности}
Function WriteData(B : Byte; Wait : LongInt): Boolean;
Var Result : Boolean; w : LongInt;
Begin
 w:= 0; Result:= False;
 For w:= 1 to Wait do begin {ожидание...}
  If GetSR then begin {передатчик готов?}
   WriteCOM(B); {передаем байт}
   Result:= True;
   Break;
  End;
 End;
 WriteData:= Result;
End;

{Чтение числа с плавающей точкой (тип single) }
Function  ReadSingle(var S : Single; WD : LongInt) : Boolean;
Var F : UFloat; R : Single;
Begin
 S:= 0.00; ReadSingle:= False;
 If ReadData(F.b[3], WD) then
  If ReadData(F.b[2], WD) then
   If ReadData(F.b[1], WD) then
    If ReadData(F.b[0], WD) then begin
     Move(F, S, 4);
     ReadSingle:= True;
    End;
End;

{"Закрытие" порта }
Procedure CloseCOM;
Begin
 WriteCOM($0FF);
End;

END.
```

Учебное пособие

Орликов Л.Н.

Измерение параметров вакуумной системы на ЭВМ в реальном режиме
Методические указания к лабораторной работе

Усл. печ. л. _____Препринт
Томский государственный университет
систем управления и радиоэлектроники
634050, г.Томск, пр.Ленина, 40