Федеральное агентство научных организаций

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ НАУКИ
ИНСТИТУТ МОНИТОРИНГА КЛИМАТИЧЕСКИХ И ЭКОЛОГИЧЕСКИХ СИСТЕМ
СИБИРСКОГО ОТДЕЛЕНИЯ РОССИЙСКОЙ АКАДЕМИИ НАУК
(ИМКЭС СО РАН)

УДК 004.78:504.3.064.36
№ госрегистрации 114102740054
Инв. № 2-2015

УТВЕРЖДАЮ
Директор ИМСКЭ СО РАН,
д.ф.-м.н.
_____ В.А. Крутиков
"_27_" декабря 2015 г.

ОТЧЕТ
О ПРИКЛАДНЫХ НАУЧНЫХ ИССЛЕДОВАНИЯХ

Разработка и создание измерительно-вычислительной системы для реализации технологии мезомасштабного мониторинга и прогнозирования состояния атмосферного пограничного слоя

по теме:

ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ ПОСТАВЛЕННЫХ ПЕРЕД ПНИ ЗАДАЧ

(1-Я ОЧЕРЕДЬ)

(промежуточный)

Этап 3

ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014-2020 годы»

Соглашение о предоставлении субсидии от 05.06.2014 г. № 14.607.21.0030

Часть 2

ПРИЛОЖЕНИЯ Ф, Х

Руководитель проекта
профессор, д.т.н.

21.12.2015 А.А. Тихомиров
подпись, дата

СОДЕРЖАНИЕ

ПРИЛОЖЕНИЕ Ф. Описание программы "МЕТЕО-3.0"..... 3
ПРИЛОЖЕНИЕ Х. Текст программы "МЕТЕО-3.0"..... 58

ПРИЛОЖЕНИЕ Ф

Описание программы "МЕТЕО-3.0"

ООО «СИБАНАЛИТПРИБОР»

УТВЕРЖДАЮ:

Директор ООО «Сибаналитприбор»

_____ А.А.Азбукин

«16» 11 2015 г.

Программное обеспечение для ПЭВМ ультразвуковой метеостанции

АМК-03

Программа «**МЕТЕО 3.0**»

Описание программы

ЛИСТ УТВЕРЖДЕНИЯ

АМЯ 4.079.001 13

Листов 54

Исполнитель, сн.с., к.ф.-м.н.,

_____ А.Я. Богушевич

«16» 11 2015 г.

2015

Инв. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

УТВЕРЖДЕН

АМЯ4.079.001 13-ЛУ

Программное обеспечение для ПЭВМ ультразвуковой метеостанции

АМК-03

Программа «**МЕТЕО 3.0**»

Описание программы

АМЯ 4.079.001 13

2015

Подп. и дата	
Инв. № дубл.	
Взам. инв.№	
Подп. и дата	
Инв. № подл.	

АННОТАЦИЯ

В настоящем документе приводится описание программы «**МЕТЕО 3.0**», являющейся стандартным (базовым) программным обеспечением (ПО) для ультразвуковых метеостанций типа **АМК-03**, устанавливаемым на ПЭВМ (компьютерах), использующих операционную систему Windows (любой 32-х или 64-х разрядной версии). Данное ПО используется совместно с указанной метеостанцией при ее подключении к ПЭВМ на таких стадиях эксплуатации как подготовка к работе, использование по назначению, градуировка и ремонт.

Документ содержит сведения о функциональном назначении программы и решаемых с ее помощью функциональных задачах при эксплуатации метеостанций **АМК-03**, описание ее логической структуры, используемых технических средств, информацию об способе вызова и загрузки, о входных и выходных данных.

В дальнейшем наименование «метеостанция **АМК-03**» заменяется для краткости словом «изделие».

Описываемое ПО входит в состав изделия и поставляется совместно с ним на диске CD-ROM.

АННОТАЦИЯ

В настоящем документе приводится описание программы «**МЕТЕО 3.0**», являющейся стандартным (базовым) программным обеспечением (ПО) для ультразвуковых метеостанций типа **АМК-03**, устанавливаемым на ПЭВМ (компьютерах), использующих операционную систему Windows (любой 32-х или 64-х разрядной версии). Данное ПО используется совместно с указанной метеостанцией при ее подключении к ПЭВМ на таких стадиях эксплуатации как подготовка к работе, использование по назначению, градуировка и ремонт.

Документ содержит сведения о функциональном назначении программы и решаемых с ее помощью функциональных задачах при эксплуатации метеостанций **АМК-03**, описание ее логической структуры, используемых технических средств, информацию об способе вызова и загрузки, о входных и выходных данных.

В дальнейшем наименование «метеостанция **АМК-03**» заменяется для краткости словом «изделие».

Описываемое ПО входит в состав изделия и поставляется совместно с ним на диске CD-ROM.

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Решаемые функциональные задачи

Программа «**METEO 3.0**» позволяет выполнять следующие функциональные задачи:

− считывания в компьютер первичных данных измерений метеостанции **АМК-03**, выдаваемых ею в виде цифрового кода в стандарте RS232, через последовательный порт COM (физический или его виртуальный аналог на портах USB или RS485);

− автоматического и непрерывного тестирования и диагностики технического состояния метеостанции и канала связи с ней с выдачей соответствующей информации оператору;

− автоматического вычисления значений метеорологических параметров, а также стандартных статистических и турбулентных характеристик метеорологических полей, с задаваемым временем усреднения из цифрового кода, принимаемого от метеостанции;

− автоматического сохранения (в задаваемые пользователем сроки метеорологических наблюдений) своих выходных данных на жестком диске компьютера в виде файлов разных типов и форматов, а также в базе данных (при необходимости);

− поддержки технологии DDE для передачи своих выходных данных другим Windows-приложениям;

− автоматизации градуировки ультразвукового термоанемометра, применяемого в составе метеостанции; оценивания значений градуировочных параметров для используемой метеостанции, сохраняемых в специальном зашифрованном файле; записи значений градуировочных параметров в энергонезависимую память вычислительного утройства типа пульта ППУ-25 (применяемого для автономных вычислений и индикации метеорологических величин без применения компьютера) и контрольного чтения из него значений этих же параметров.

1.2 Файловый состав

Файловый состав программы «**METEO 3.0**» включает следующие компоненты.

Каждый экземпляр программы имеет основной файл с исполняемым кодом, имя которого соответствует заводскому (серийному) номеру метеостанции, например как «**АМК_XXXXX.EXE**» (цифры заводского номера здесь условно обозначены как «**XXXXX**»). Для хранения значений градуировочных параметров, индивидуальных для каждой метеостанции, и заводского номера используется шифруемый файл данных с тем же именем как у исполняемого файла, но с другим расширением «**.SYS**» (для приводимого примера оно будет «**АМК_XXXXX.SYS**»). При каждом запуске программы (файла «**АМК_XXXXX.EXE**») она считывает из файла «**АМК_XXXXX.SYS**» градуировочные параметры используемой с ней метеостанции.

Каждый экземпляр программы использует динамически загружаемые библиотечные компоненты. К ним относятся файлы:

- «**Meteo.DLL**» - содержит программные реализации математических формул из физики атмосферы и атмосферной акустики, используемых при вычислениях метеорологических параметров из данных измерений метеостанции;

- «**AMKport.DLL**» - содержит программные реализации работы в компьютере с подключаемой к нему через порты COM метеостанцией (интерфейс связи) – они наследуют возможности API-функций Windows из раздела «Comunication functions»;

- «**StanDlgs.DLL**» - содержит обращения к стандартным диалоговым окнам Windows и диалоговые сообщения пользователю;

- «**METEO_RU.HLP**» и «**METEO_RU.CNT**» - файлы справочной системы.

По завершению каждого сеанса работы программа также генерирует файл «**АМК_XXXXX.INI**», в котором она сохраняет последние настройки в ее интерфейсе пользователя, автоматически предлагаемые пользователю при следующем ее запуске.

К метрологически значимой части ПО относятся файлы с программным кодом «**АМК_XXXXX.EXE**», «**Meteo.DLL**» и «**AMKport.DLL**», а также файл данных «**АМК_XXXXX.SYS**» (с индивидуальными значениями градуировочных параметров для используемой метеостанции). Идентификационные данные ПО приведены в таблице 1.

Таблица 1.

Наименование ПО	Идентификационное наименование ПО	Номер версии (идентификационный номер) ПО	Цифровой идентификатор ПО (контрольная сумма исполняемого кода)	Алгоритм вычисления идентификатора ПО
Программа «**METEO 3.0**»	AMK_XXXXX.EXE	3.0.RT	a129df49179cd74f6864c1778d216bba	WinMD5Free v.1.20
Библиотечный файл «**Meteo.DLL**»	meteo.dll	не имеет	5a57ec506a1a8096fea48b85a561c6f3	WinMD5Free v.1.20
Библиотечный файл «**AMKport.DLL**»	amkport.dll	не имеет	f6e75485d59492c4024022bc372e0ea3	WinMD5Free v.1.20
Файл данных с градуировочными параметрами метеостанции	AMK_XXXXX.SYS	не имеет	Индивидуальный для каждой метеостанции, вносимый в свидетельство о ее поверке	WinMD5Free v.1.20

1.3 Используемые технические средства

Для работы программы «**METEO 3.0**» необходимо, чтобы подключаемый к изделию компьютер удовлетворял следующим требованиям:

- процессор с тактовой частотой не менее 600 МГц;
- наличие порта COM (или драйвера для создания его виртуального аналога для порта USB или RS485);
- оперативная память не менее 512 Мб;
- наличие свободного места на жестком диске объемом не менее 50 Мб;
- разрешение экрана не менее 1024 x 768 (при визуальном контроле выходных данных ПО в числовом и графическом виде).

ПО может функционировать в следующих операционных системах: Windows’98, Windows’2000, Windows XP или в другой версии Windows, более поздней по отношению к упомянутым (до Windows 10 включительно).

При разработке данного ПО применялась система объектно-ориентированного программирования «Delphi 6.0» фирмы «Borland Enterprise» и алгоритмический язык программирования «Object Pascal».

1.4 Уровень защиты данных

Уровень защиты программы «**МЕТЕО 3.0**» от непреднамеренных или преднамеренных изменений при использовании цифровых идентификаторов, описанных в таблице 1, по МИ 3286-2010 соответствует категории «С».

ПО ни каким образом не может оказывать влияния на выполнение измерений в изделии и выдаваемые им результаты этих измерений в виде цифрового кода, т.е. на свои входные данные. Визуализация выходных данных ПО на экране монитора обеспечивается программным кодом исполняемого файла «**АМК_XXXXX.EXE**» в режиме «**только для чтения**» и не доступна для редактирования пользователем.

В ПО для защиты процесса вычисления метеорологических параметров из входных данных применяется специальное кодирование (шифрование) файла **АМК_XXXXX.SYS**, в котором содержатся градуировочные параметры изделия. В случае нарушения соответствия имен файлов «**АМК_XXXXX.EXE**» и «**АМК_XXXXX.SYS**», при их несоответствии заводскому номеру изделия, также записанному в файл «**АМК_XXX.SYS**», а также при обнаружении нарушения правил кодирования данных, ПО прерывает свою загрузку с выдачей аварийного сообщения на экран монитора. Доступ к изменению данных в файле «**АМК_XXXXX.SYS**» при градуировке изделия, а также их запись в память пульта ППУ-25, возможны только через интерфейс программы «**МЕТЕО 3.0**» после ввода в ней разрешающего административного пароля.

2 ОПИСАНИЕ ОСНОВНОГО ИНТЕРФЕЙСА

2.1 Система меню и «горячие» клавиши

2.1.1 Программа «**МЕТЕО 3.0**» имеет систему меню, состоящую из четырех субменю: «**Работа**», «**Сервис**», «**Окно**» и «**Справка**», каждое из которых включает команды управления. Латинские символы в конце наименований команд меню, как это принято в Windows, показывают «**горячие**» клавиши для быстрого доступа к ним с клавиатуры. Начальная часть наименования команд до косой линии повторяет наименование субменю, в которых они находятся. Ниже перечисляются имеющиеся в ПО команды с описанием их рабочих функций.

2.1.2 Команда меню «**Работа / Измерения…F9**».

Выполняет запуск измерений метеорологических параметров; предварительно открывает диалоговое окно «**Параметры измерений…**» (см. п. 4.1).

Она также может быть выполнена с помощью кнопки “**Старт**”, расположенной в главном окне программы, если в этом же окне с помощью группы радиокнопок «**Режим»** выбрана опция «**Измерения**».

2.1.3 Команда меню «**Работа / Градуировка…F8**».

Выполняет запуск автоматизированной градуировки ультразвукового термоанемометра после получения доступа через пароль; предварительно открывает диалоговое окно «**Параметры градуировки…**» (см. п. 4.2).

Она также может быть выполнена с помощью кнопки “**Старт**”, расположенной в главном окне программы, если в этом же окне с помощью группы радиокнопок «**Режим** выбрана опция «**Градуировка**».

2.1.4 Команда меню «**Работа / Контроль…F7**».

Выполняет запуск контрольного считывания в компьютер первичных данных изделия и их отображение на экране монитора (см. п.4.3).

Она также может быть выполнена с помощью кнопки “**Старт**”, расположенной в главном окне программы, если в этом же окне с помощью группы радиокнопок «**Режим**» выбрана опция «**Контроль**».

2.1.5 Команда меню «**Работа / Строки данных…F4**».

Открывает окно «**Строки данных…**», в котором пользователю предлагается просмотреть результаты метеорологических измерений в виде массива строковых записей данных с установленным оператором временем усреднения и периодичностью сроков их регистрации во времени (см. п. 4.1 и рис. 15).

Данная команда при загрузке программы для пользователя недоступна. Доступ к ней открывается во время проведения метеорологических измерений по истечению срока первой регистрации данных

2.1.6 Команда меню «**Работа / Отчёт…F5**».

Открывает окно «**Отчет…**», в котором пользователю предлагается просмотреть статистический отчет о результатах последних метеорологических измерений (см. п. 4.1 и рис. 16).

Данная команда при загрузке программы для пользователя недоступна. Доступ к ней открывается только после проведения метеорологических измерений в течение времени, равного задан-

ному пользователем времени усреднения, или по завершению первого сеанса метеорологических измерений.

2.1.7 Команда меню «**Работа / Выход … Alt+X**».

Завершает работу программы.

2.1.8 Команда меню «**Сервис / Настройки…F10**».

Открывает многостраничное диалоговое окно «**Настройки…**», в котором можно выполнить индивидуальную настройку программы под конкретные потребности пользователя и учесть технические параметры используемого изделия (см. п.3.3).

2.1.9 Команда меню «**Сервис / Вид графики…**».

Открывает диалоговое окно «**Вид графики…**» (см. п.3.4), в котором можно изменить стиль и цветовую гамму графического отображения программы на экране монитора по индивидуальному желанию пользователя.

2.1.10 Команда меню «**Сервис / Градуировка пульта…**».

Открывает модальное окно «**Чтение/запись градуировочных параметров во внешний пульт…**» (см. п.4.4), предназначенное для градуировки пульта **ППУ-25**.

Пульт **ППУ-25** содержит микропроцессор с программой, позволяющей производить автономное (без компьютера) вычисление и индикацию значений метеорологических величин с задаваемым временем усреднения.

2.1.11 Команда меню «**Сервис / Тестирование пульта…**».

Открывает модальное окно «**Тестирование пульта…**» (см. п.4.5), предназначенное для выполнения тестирования работы блока **ППУ-25**.

2.1.12 Команда меню «**Сервис / База данных…**».

Загружает Windows-приложение «**METEO DB**”, с помощью которого выполняется оперативный просмотр таблиц **базы данных**.

Данная команда меню доступна пользователю для исполнения только при установке на компьютер дополнительного приложения «**METEO DB»** в одном каталоге (папке) с базовым ПО.

2.1.13 Команда меню «**Сервис / Анализ данных…**».

Загружает Windows-приложение «**METEO DP**”, с помощью которого выполняется научная обработка файлов с выходными данными ПО.

Данная команда меню доступна пользователю для исполнения только при установке на компьютер дополнительного приложения «**METEO DP»** в одном каталоге (папке) с базовым ПО.

2.1.14 Команда меню «**Окно/Минимальное…Alt+0**».

По данной команде главное окно приложения становится невидимым на экране монитора, закрываются все дочерние окна приложения, и на экране в правом верхнем углу появляется минимальное окно приложения (см. рис. 14).

Она также может быть исполнена с помощью кнопки “**Минимально…”** на главном окне приложения.

2.1.15 Команда меню «**Окно/Главное… Ctrl+0**».

По данной команде главное окно приложения (см. рис. 2) становится видимым и выносится на передний фронт на экране монитора по отношению к другим открытым окнам. Если в приложении открыто минимальное окно, то при исполнении данной команды последнее становится невидимым.

2.1.16 Команда меню «**Окно/Метеорологические измерения…Ctrl+1**».

По данной команде окно «**Метеорологические измерения**» (см. рис. 13) выносится на передний фронт на экране монитора по отношению к другим открытым окнам. Если данное окно на момент исполнения команды в программе не открыто, то предварительно выполняется его открытие.

2.1.17 Команда меню «**Окно/Градуировка термоанемометра…Ctrl+2**».

По данной команде окно «**Градуировка термоанемометра**» (см. рис. 21) выносится на передний фронт на экране монитора по отношению к другим открытым окнам. Если данное окно на момент исполнения команды в программе не открыто, то предварительно выполняется его открытие.

2.1.18 Команда меню «**Окно/Контроль данных устройства…Ctrl+3**».

По данной команде окно «**Контроль данных устройства**» (см. рис. 22) выносится на передний фронт на экране монитора по отношению к другим открытым окнам. Если данное окно на момент исполнения команды в программе не открыто, то предварительно выполняется его открытие.

2.1.19 Команда меню «**Справка / Содержание**»

Открывает справочную систему приложения на странице «**Содержание**».

2.1.20 Команда меню «**Справка / О программе…**»

Открывает одноименное информационное модальное окно.

2.1.21 Команда меню «**Справка / О приборе…**»

Открывает одноименное информационное модальное окно.

2.1.22 ПО также поддерживает отработку «**горячих**» клавиш, являющихся стандартными в Windows:

<Alt> – при активном главном окне программы активизирует его меню;

<Alt+F4> – закрывает активное окно (если активно главное окно приложения, то завершает работу программы);

<Alt_Tab> – позволяет переключаться на другие окна и программы, открытые в текущий момент в Windows;

<Tab> – перемещает фокус ввода в активном окне с одного элемента управления на другой;

<F1> – вызывает справочную систему программы и открывает ее на теме, соответствующей выполняемому оператором действию (вызов справки по контексту работы программы);

<Esc> – отмена действия, ранее инициированного оператором (если выполняется работа в модальном или диалоговом окне приложения, то последнее в этом случае закрывается);

<Enter> – подтверждение на выполнение действия, инициированного оператором (если выполняется работа в модальном или диалоговом окне приложения, то последнее также закрывается).

Клавиши управления курсором (со стрелками) могут использоваться для перемещения фокуса ввода внутри многосоставного элемента управления.

2.2 Инструментальные кнопки

Рис. 1.

В главном окне программы ниже строки меню имеется инструментальная панель, показанная на рис. 1, которая содержит кнопки быстрого доступа (одним щелчком левой кнопки мыши) к большинству команд меню (см. табл. 2). Данные кнопки не имеют подписей и различаются только отображенными на них пиктограммами. Если какой-либо команде программа в текущий момент запрещает возможность исполнения, то соответствующая пиктограмма изменяет все цвета рисунка на бледно-серый. При наведении курсора мыши на какую-либо из этих кнопок всплывает малое окно подсказки (желтого цвета) с кратким именем исполняемой ею команды.

Таблица 2.

№	Пиктограмма на кнопке	Исполняемая кнопкой команда
1		Команда меню “**Работа/Измерения…F9**”
2		Команда меню “**Работа/Градуировка…F8**”
3		Команда меню “**Работа/Контроль…F7**”
4		Команда кнопки на главном окне “**Графика…**”
5		Команда меню “**Окно/Минимальное…**”
7		Команда меню “**Работа/Строки данных…F4**”
7		Команда меню “**Работа/Отчет…F5**”
8		Команда меню “**Сервис/Настройки…F10**”
9		Команда меню “**Сервис/Вид графики…**”
10		Команда меню “**Сервис/Градуировка пульта…**”
11		Команда меню “**Сервис/База данных…**”
12		Команда меню “**Сервис/Анализ данных…**”
13		Команда меню “**Справка/Содержание**”
14		Команда меню “**Работа/Выход…Alt+X**”

2.3 Главное окно программы

2.3.1 При загрузке программы **«МЕТЕО 3.0»** в левом верхнем углу экрана монитора появляется ее главное окно с заголовком «**Ультразвуковая метеостанция (МЕТЕО 3.0): АМК-03, № XXXXX**», где **XXXXX** – цифры заводского номера изделия.

Главное окно приложения содержит (см. рис. 2):

− сверху - линейку меню, описанного в подразд. 2.1;

− под ней - линейку с кнопками быстрого доступа к командам меню (см. подразд. 2.2);

− в центре слева - панель «**Работа**» для оперативного управления работой программыс помощью мыши;

− в центре справа - панель «**Значения**» для отображения в числовой и графической форме результатов измерений (выходных данных);

− ниже - панель технического контроля процесса выполнения измерений используемым изделием;

− в самом низу - панель для отображения строки контекстной подсказки оператору.

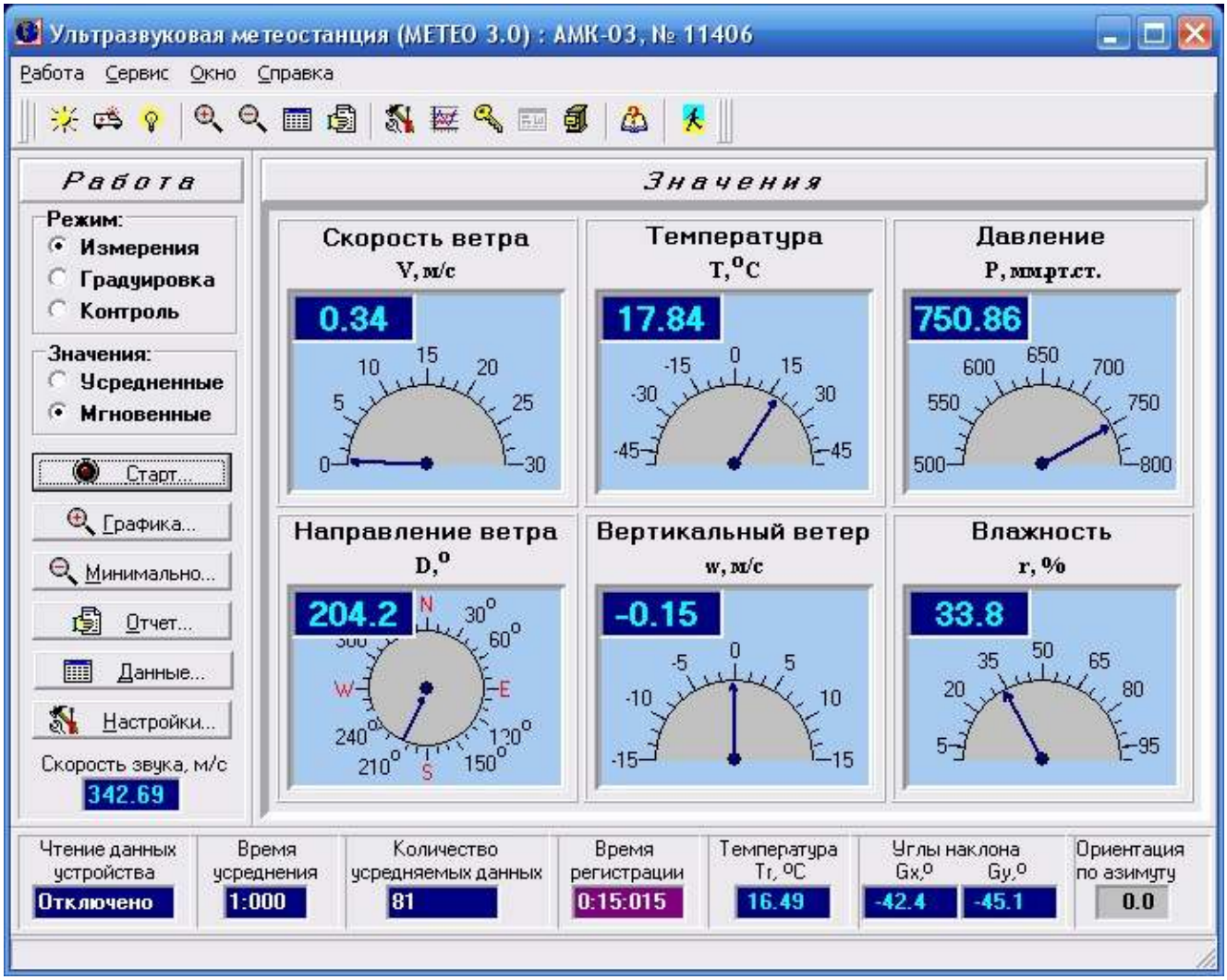

Рис. 2

2.3.2 Вход в меню осуществляется либо щелчком по нему левой кнопкой мыши, либо с клавиатуры нажатием клавиши **<Alt>**. Далее можно перемещаться по командам меню либо также щелчками мыши, либо с помощью клавиш управления курсором (со стрелками). Прямой доступ с

клавиатуры к некоторым командам меню также осуществляется нажатием соответствующей «**горячей**» клавиши (см. подразд. 2.1.). Наиболее важные команды меню продублированы кнопками быстрого доступа с пиктограммами (см. подразд. 2.2.). Программа подсказывает пользователю, какие его команды в текущий момент работы она может выполнить, а какие не следует. Для последних команд все органы управления, через которые они могут быть назначены на исполнение, временно делаются недоступными и отображаются бледным серым цветом.

2.3.3 Управление совместной работой программы с подключенным к ПЭВМ изделием также может выполняться с помощью дополнительных кнопок на панели «**Работа**». Здесь имеется группа радиокнопок «**Режим**», с помощью которой оператор может выбирать один из трёх основных режимов работы программы с изделием (измерений метеорологических величин, градуировки термоанемометра, контроля выходных данных изделия).

Управление непосредственно запуском и принудительным прерыванием процесса измерений осуществляется в любом из трех режимов с помощью кнопки «**Старт**» (после запуска измерений и до их остановки она изменяет свой заголовок на «**Стоп**»).

2.3.4 На панели «**Значения**» отображаются (цифровые значения и графически на стрелочных индикаторах) следующие измеряемые величины:

− во время исполнения команды меню «**Работа / Измерения…F9**» - скорость горизонтального ветра (применяемое сокращенное обозначение – V, м/с), направление горизонтального ветра (D, ^{0}), температура воздуха (T, ^{0}C), скорость вертикального ветра (w, м/с), атмосферное давление (P, мм рт.ст.) и относительная влажность воздуха (r, %);

− во время исполнения команды меню «**Работа / Градуировка…F8**» - расстояния между 4-мя парами ультразвуковых датчиков термоанемометра (S_i, мм), атмосферное давление (P, мм рт.ст.) и относительная влажность воздуха (r, %);

− во время исполнения команды меню «**Работа / Контроль…F7**» - времена распространения акустических сигналов между 4-мя парами ультразвуковых датчиков термоанемометра (X1-X4), выраженные в числе синхроимпульсов кварцевого генератора, первичные показания датчика давления и влажности (X5, ед. АЦП и X6, ед. АЦП).

2.3.5 На панели «**Работа**» имеется группа радиокнопок «**Значения**», с помощью которых оператор всегда может переключить режим отображения перечисленных выше данных. В зависимости от его выбора могут отображаться либо их текущие средние значения, либо их последние мгновенные значения. Кроме того, если оператор выполнит двойной щелчок левой кнопкой мыши на любой из шести субпанелей на панели «**Значения**», то выбранная им субпанель распахнётся практически до размеров главного окна, закрыв другие субпанели и панели. Оператор будет видеть в крупном масштабе только один измеряемый параметр. Повторный двойной щелчок мышью по данной субпанели приведет ее в исходное состояние.

На панели «**Работа**» также отображается вычисляемое во время измерений значение скорости звука в атмосферном воздухе (в м/с).

2.3.6 Панель технического контроля содержит ряд индикаторов, позволяющих оператору непрерывно следить за качеством работы изделия. Первый индикатор «**Чтение данных устройства**» показывает, как считывается цифровой код с изделия и в каком техническом состоянии находится изделие. Если программа не выполняет измерения, то данный индикатор имеет синий цвет и отображает «**Отключено**». Если программа выполняет измерения (в любом из трех режимов) и при этом цифровой код считывается нормально и не содержит ошибок, то индикатор имеет зеленый цвет и отображает «**Нормально**». Если же во время измерений происходит задержка в поступлении кода в компьютер до 2 секунд, то индикатор имеет черный цвет и отображает «**Задержка**».

При задержке поступления кода, превышающей 2 секунды, индикатор меняет цвет на красный и отображает «**Нет данных**». Если программа непрерывно в течение более 2 секунд обнаруживает, что поступающий код содержит ошибки, то на индикаторе (красного цвета) отображается «**Помехи**». В случае обнаружения программой недостоверных данных изделия данный индикатор имеет серо-зеленый цвет и отображает «**Сбой**». Оператор во время измерений всегда имеет возможность получить более детальную диагностику технического состояния изделия (в части, отвечающей за выполнение им измерений), для чего ему следует щелкнуть мышью по данному индикатору. При этом программа откроет окно с развернутым текстовым сообщением о состоянии изделия. Выдаваемое в этом случае аварийное сообщение соответствует перечню возможных неисправностей изделия, приведенному в руководстве по его эксплуатации.

Следующий индикатор на панели технического контроля («**Время усреднения**») во время измерений отображает текущее значение времени усреднения измеряемых величин. В начале измерений текущее значение времени усреднения непрерывно возрастает и совпадает с длительностью измерений. По достижению им величины, заданной пользователем при запуске измерений, его значение практически не меняется.

2.3.7 Индикатор «**Количество усредняемых данных**» показывает, сколько всего было зарегистрировано программой и хранится в данный момент в оперативной памяти ПЭВМ информационных пакетов первичных данных изделия за текущее значение времени усреднения. Данные пакеты поступают в ПЭВМ с частотой, указанной в технических характеристиках изделия (10 или 80 Гц). Часть из них при приеме в ПЭВМ программой могут отбрасываться при их регистрации. Регистрация одного информационного пакета для пользователя эквивалентна получению одного мгновенного значения каждой из измеряемых величин.

2.3.8 Индикатор «**Время регистрации**» отображает время регистрации программой последнего информационного пакета изделия в произведенном сеансе измерений. Показание этого индикатора на момент завершения измерений фиксируется в выходных данных программы (в файлах и в базе данных).

2.3.9 Индикатор «**Температура, Tr, ^{0}C**» отображает значение температуры на корпусе датчика влажности в градусах Цельсия, измеряемого с помощью встроенного в этот датчик терморезистора.

2.3.10 Пара индикаторов «**Углы наклона**» отображают значения углов наклона термоанемометра от вертикали (в градусах) вдоль осей X (угол **Gx**) и Y (угол **Gy**), измеряемых встроенным двухкоординатным акселерометром внутри корпуса термоанемометра.

Последний индикатор «**Ориентация по азимуту**» показывает введенное с клавиатуры значение угла ориентации термоанемометра по азимуту (угла между его осью X и направлением на север).

2.3.11 На панели «**Работа**» также имеются кнопки «**Графика…**», «**Минимально…**», «**Отчет…**», «**Данные…**» и «**Настройки…**». Кнопка «**Графика…**» открывает одно из трех возможных дополнительных окон, предназначенных для графического отображения мгновенных значений измеряемых величин. Тип открываемого окна выбирается программой в зависимости от установленного в ней текущего режима измерений. Последние четыре кнопки обеспечивают с помощью мыши более быстрый и удобный доступ к командам меню «**Окно/Минимальное…Alt+0**», «**Работа/Отчет...F5**», «**Работа/Строки данных...F4**» и “**Сервис/Настройки...F10**» (см. п. 2.1).

3 СИСТЕМНЫЕ НАСТРОЙКИ ПРОГРАММЫ И СРЕДСТВА ИХ ИЗМЕНЕНИЯ

3.1 Первичная настройка программы

Первичная настройка программы выполняется при кабельном соединении изделия с ПЭВМ (через один из ее портов COM, USB или RS485) в следующем порядке:

− выполнить запуск программы **«METEO 3.0»** и дождаться появления ее главного окна;

− используя команду меню “**Сервис/Настройки..F10**», открыть в программе многостраничное диалоговое окно «**Настройки…**» (см. ниже подразд. 3.2) и переключиться в этом окне на страницу «**Связь**»;

− проверить на этой странице введенный номер порта COM, через который подключается изделие (в случае использования порта USB или RS485 номер соответствующего ему виртуального аналога COM), и установленную скорость порта – при необходимости привести их в соответствие с фактическими значениями;

− на странице «**Программа»** окна «**Настройки…**» проверить опции по использованию датчиков давления и влажности и опции записи требуемых выходных данных программы;

− если показания датчиков давления или влажности не предполагается использовать, и они на странице «**Программа»** отмечены как отключенные, то на странице «**Значения**» окна «**Настройки…**» также ввести с клавиатуры реальные значения измеряемых ими величин (для замены их показаний);

− здесь же, на странице «**Значения**», ввести при необходимости значение угла азимутальной ориентации термоанемометра (учесть его ориентацию относительно севера после установки на мачте);

− включить питание изделия и, используя команду меню «**Работа/Контроль...F7»**, выполнить контрольный опрос его выходных данных; если программа настроена правильно, то на экране монитора должны отображаться поступающие с изделия первичные данные измерений; в противном случае программа выдаст сообщение об ошибке.

3.2 Многостраничное диалоговое окно «Настройки…»

3.2.1 Диалоговое окно «**Настройки…**», открываемое командой меню «**Сервис / Настройки…F10**», позволяет изменять наибольшее количество настроек программы **«METEO 3.0»**, создавая тем самым ее конфигурацию под конкретные потребности пользователя. По завершению работы программы последние значения всех измененных параметров ее настройки автоматически фиксируются в вспомогательном файле «**АМК_XXXXX.INI**», данные которого далее используются программой при всех ее последующих запусках.

В данном окне (это также относится ко всем диалоговым окнам программы) нажатие кнопки «**Отмена»** воспринимается программой как отказ от ввода новых значений параметров и от выполнения текущей задачи. При этом окно закрывается без проверки и сохранения произведенных изменений в полях ввода значений параметров. Ввод в программу новых значений параметров с одновременным подтверждением выполнения текущей задачи выполняется только при успешной отработке нажатия кнопки «**ОК**». При этом окно также закрывается. При нажатии кнопки «**ОК**» программа проверяет последовательно все поля ввода параметров на наличие ошибок оператора. При обнаружении первой же ошибки отработка команды кнопки «**ОК**» прекращается с выдачей оператору на экран монитора соответствующего сообщения об его ошибке.

3.2.2 Первая страница окна с заголовком «**Связь**» (см. рис. 3) предназначена для установки параметров связи компьютера с изделием.

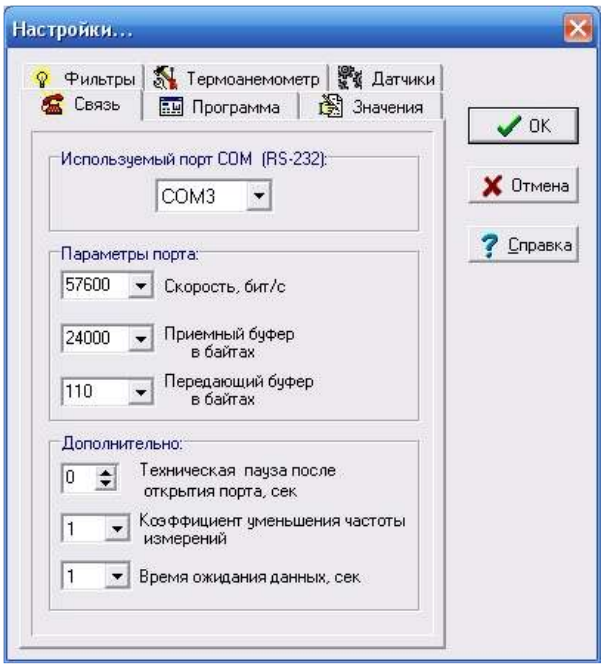

Рис. 3

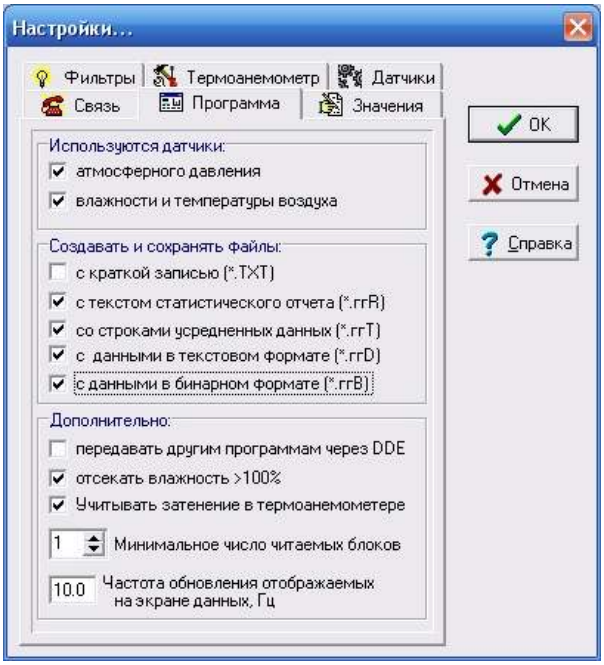

Рис. 4

Пользователь здесь может изменять следующие настройки программы.

Используемый порт COM (RS232) – выбирается из выпадающего списка портов (от COM1 до COM256).

Параметры порта COM (скорость передачи данных, размеры в байтах приемного и передающего буфера). В случае кабельного соединения изделия с ПЭВМ скорость порта всегда должна быть равна скорости выдачи цифрового кода с изделия (применяемое стандартное значение в изделии – 57600 бит/с). Передающий буфер используется только при работе с пультом **ППУ-25**.

Техническая пауза после открытия порта, сек (допускаемые значения – от 0 до 59 секунд) устанавливается не равной нулю только в том случае, если запуск передачи цифрового кода из изделия в последнем производится по сигналу порта ПЭВМ о его открытии (сигналу **DTR**). В типовых модификациях изделия этот запуск происходит при включении его питания.

Коэффициент уменьшения частоты измерений (возможные значения - 1, 2, 4, 8, 16, 32 и 64; при поставке изделия равен 1) устанавливает, во сколько раз программа должна уменьшать частоту регистрации мгновенных значений метеорологических величин в ПЭВМ по сравнению с частотой выдачи изделием первичных данных измерений, установленной на аппаратном уровне в самом изделии. Данный параметр регулирует нагрузку программы на процессор ПЭВМ и на объем требуемого ею места на жестком диске под файлы, сохраняемые программой с результатами измерений мгновенных значений метеорологических величин. Если пользователь обнаруживает, что работа программы совместно с изделием требует от применяемой ПЭВМ больших ресурсов, чем она может обеспечить по своим техническим характеристикам, то следует устанавливать здесь значение больше 1.

Время ожидания данных, сек (допускаемые значения – от 1 до 5 секунд) устанавливает максимальное время отсутствия данных изделия, после которого программа должна выдавать аварийное сообщение.

3.2.3 Вторая страница окна «**Настройки…**» с заголовком «**Программа**» (см. рис. 4) предназначена для задания (с помощью кнопок с независимой фиксацией) командных опций, используемых при функционировании программы.

На данной странице имеются 3 группы опций.

В первой группе «**Используются датчики**» пользователю дается возможность программно отключать опрос показаний датчиков давления или влажности (например, когда они неисправны, или с ними нет связи, или они отсутствуют).

Во второй группе «**Создавать и сохранять файлы**» пользователю дается возможность включать или выключать по необходимости запись программой на жесткий диск файлов разного типа (более детально о типах сохраняемых программой файлов см. в разд. 7).

В третьей группе «**Дополнительно**» опция «**Передавать другим программам через DDE**» позволяет включать или выключать в программе возможность динамической передачи ее выходных данных другим Windows-приложениям по технологии DDE (см. подразд. 7.7). Когда пользователь не использует других программ, связывающихся с «**METEO 3.0**» через DDE, то данную опцию лучше отключить.

Включение опции «**Отсекать влажность >100%**», имеющейся в данной группе параметров, дает указание программе не выдавать значение влажности воздуха, больше 100%, которое физически не может быть. Ее включение полезно, если влажность воздуха близка к 100%, а показания соответствующего датчика в изделии «зашкаливают» за 100% (вследствие погрешности его измерений).

Опция «**Учитывать затенение в термоанемометре**» иногда отключается изготовителем изделия при его ветровых испытаниях в аэродинамической трубе: при стандартной эксплуатации изделия она всегда должна быть включенной.

Параметр «**Минимальное число читаемых блоков**» определяет частоту обращения программы за новыми данными изделия к приемному буферу порта COM. При открытии порта COM для чтения данных изделия программа выдает команду Windows организовать приемный буфер в оперативной памяти ПЭВМ, в который должны заноситься все вновь поступающие данные из порта COM. Программа периодически обращается к этому буферу и если в этом буфере число накопленных блоков данных оказывается больше установленного здесь значения, то оно считывает из него все данные. Данный параметр служит средством регулировки и оптимизации сопряжения системы программа – ПЭВМ – внешнее устройство (изделие) и его оптимальное значение устанавливается самим изготовителем.

Параметр «**Частота обновления отображаемых данных, Гц**» задает частоту обновления числовых и графических выходных данных программы в ее окнах во время работы с изделием. Его значение не имеет смысла устанавливать больше 10 Гц; иначе обновление данных на экране будет чаще, чем его будет способен воспринимать человеческий глаз. Более того, если значение данного параметра установить меньше этой величины, то требования к видеокарте ПЭВМ заметно снижаются за счет уменьшения ее нагрузки. В этом случае программа будет отображать на экране только часть обрабатываемых ею числовых данных; причем неотображаемые числовые данные продолжают обрабатываться в оперативной памяти ПЭВМ и не будут потеряны при их сохранении в выходных файлах программы.

3.2.4 На странице «**Значения**» (см. рис. 5) имеется возможность задать с клавиатуры значения атмосферного давления, относительной влажности и углов наклона термоанемометра от вертикали в двух ортогональных плоскостях. При выполнении измерений введенные здесь значения всегда учитываются при вычислениях метеорологических величин. Когда на странице «**Программа**» какой-либо датчик изделия (давления или влажности) отмечен как отключенный, то вместо его показаний в программе используется значение соответствующего параметра, введенное оператором на данной странице. Если этот же датчик отмечен как включенный, то введенное для него значение на данной странице будет использоваться программой как поправка к показаниям датчика (для исключения его систематической погрешности измерений). Для углов наклона термоанемометра введенные здесь значения используются исключительно только как поправки к показаниям акселерометра.

На этой же странице можно изменять размерности для вычисляемых и отображаемых программой значений атмосферного давления (в гПа или мм.рт.ст.) и направления скорости ветра

(в делениях угломера или градусах), а также ввести имя папки (директории) на жестком диске, в которую программа должна выполнять автоматическое сохранение записываемых ею файлов.

Здесь также вводится тип подстилающей поверхности (выбирается из стандартизированного перечня), который используется программой только в выходных файлах для документирования условий измерений. Логический номер сервера вставляется только в файл с записью результатов измерений в виде информационной строки (см. подразд. 5.5).

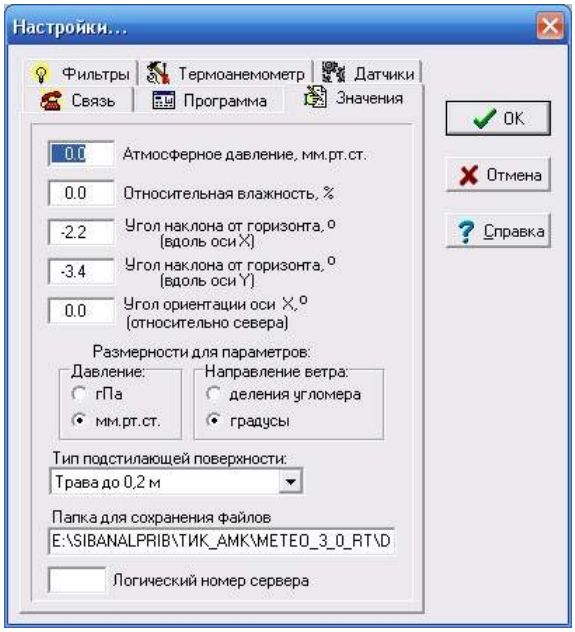

Рис. 5

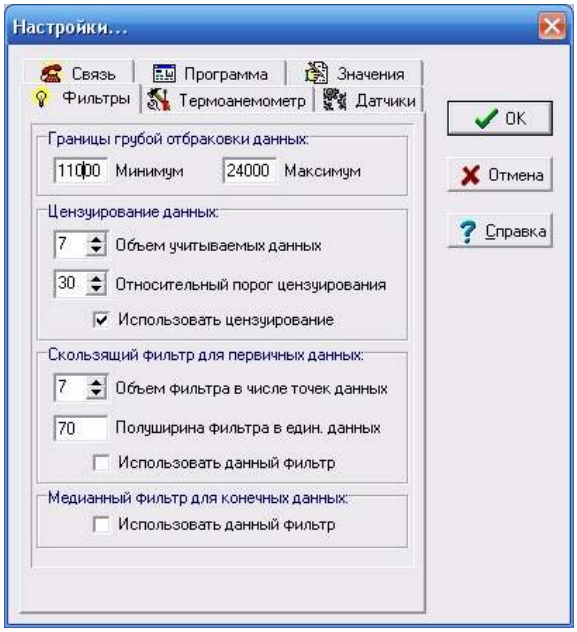

Рис. 6

3.2.5 <u>На странице «**Фильтры**»</u> (рис. 6) задаются параметры контроля и фильтрации первичных данных ультразвукового термоанемометра, применяемого в изделии. Если в работе термоанемометра по какой-либо причине начнут происходить случайные сбои, то они будут приводить к ошибкам в первичных данных измерений и в конечном итоге могут привести к неверным результатам в вычислениях температуры, скорости и направления ветра. Для повышения надежности получения достоверных данных измерений в программе предусмотрены два вида входного контроля данных термоанемометра и применение двух математических фильтров. Входной контроль заключается в проверке данных на соответствие какому-либо условию и отбраковке (цензуированию) тех данных, которые не соответствуют этому условию. Первый вид контроля (грубая отбраковка) заключается в проверке данных термоанемометра на то, что их значения находятся внутри диапазона допускаемых изготовителем величин. Этот вид контроля в программе функционирует всегда. Имеется только одна возможность его регулировки - это изменять на странице «**Фильтры**» нижнюю и верхнюю границы допускаемых данных термоанемометра. Если эти границы раздвинуть максимально, то этот вид контроля фактически перестанет функционировать.

Второй вид контроля (цензуирование данных), который, в отличие от первого, на этой же странице можно отключать, заключается в проверке данных термоанемометра по скорости их изменения во времени. Метеорологические параметры в атмосфере изменяют выходные данные термоанемометра гораздо медленнее во времени, чем аппаратные сбои. При использовании цензуирования программа непрерывно оценивает абсолютную разность однотипных данных термоанемометра в соседних информационных пакетах (по мере их поступления). Эти разности усредняются по заданному количеству последовательных информационных пакетов (объему учитываемых данных). Контроль выполняется по условию, что данные в последнем поступившем пакете отличаются от вычисленной средней разности не более, чем в заданное число раз, которое задается порогом цензуирования. На странице «**Фильтры**» можно изменять объем учитываемых данных (рекомендуемое значение – 11) и порог цензуирования (рекомендуемые значения – от 25

до 50). Этот вид контроля чрезвычайно полезен при наличии во время измерений механических или электромагнитных воздействий на термоанемометр.

Программа дополнительно может выполнять скользящую фильтрацию данных термоанемометра при их считывании и (по завершению измерений) медианную фильтрацию временных рядов мгновенных метеорологических величин. Использование указанной пары фильтров позволяет получать достоверные оценки средних значений метеорологических величин даже при чрезвычайно частых сбоях (до 49 % от общего количества данных) в работе термоанемометра. Однако следует помнить, что при этом, в отличие от случая использования отбраковки данных, искажается информация о других статистических характеристиках метеорологических величин, в том числе о параметрах атмосферной турбулентности. Оператор должен сам принимать решение, когда целесообразно применить тот или иной фильтр. При поставке изделия они всегда отключены.

3.2.6 Страницы «**Термоанемометр**» (см. рис. 7) и «**Датчики**» (см. рис. 8) позволяют просматривать значения градуировочных параметров устройства, установленных при последней градуировке изделия (изготовителем или метрологами после поверки изделия) и записанные в его паспорт. Поскольку эти параметры определяют метрологические характеристики изделия, то доступ к их изменению здесь ограничен (за исключением параметра «**Высота над земной поверхностью, м**», используемого только для документирования условий измерений в файлах с выходными данными).

Для получения доступа к вводу новых значений градуировочных параметров здесь следует нажать кнопку с ключом и ввести административный пароль, записанный в паспорте изделия.

При эксплуатации изделия часть градуировочных параметров на странице «**Термоанемометр**» обновляется самой программой после выполнения каждой автоматической градуировки термоанемометра (см. также подразд. 4.2).

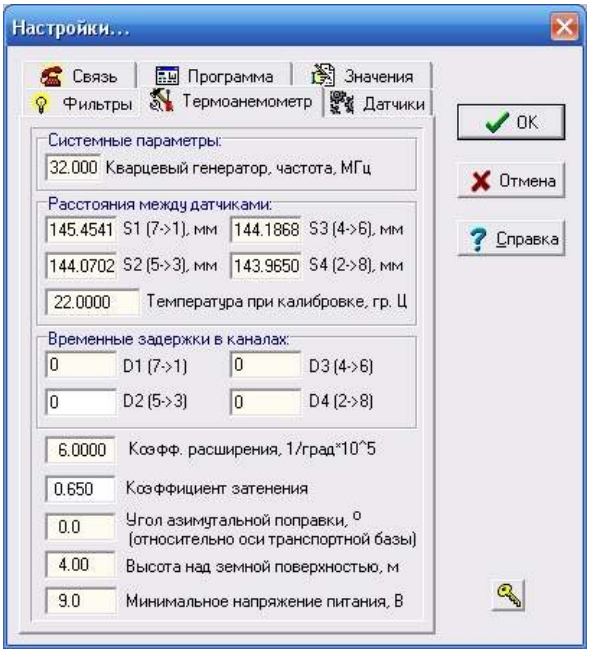

Рис. 7

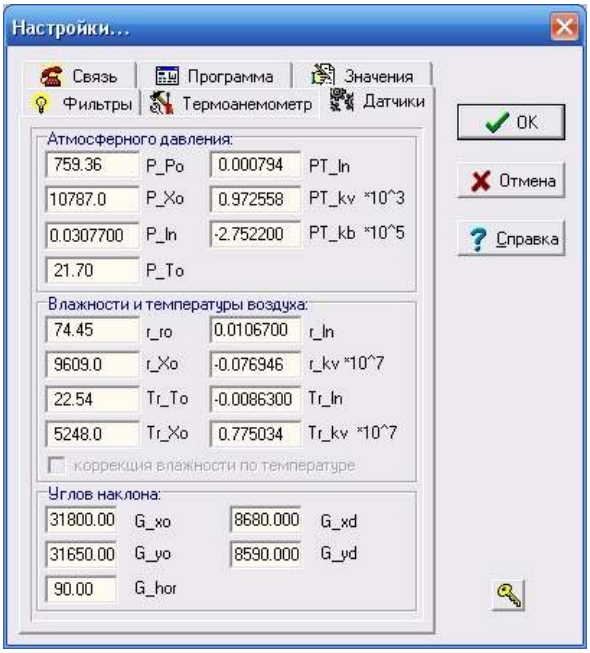

Рис.8

3.3 Диалоговое окно «Вид графики…»

Используя команду меню **«Сервис/Вид графики…»**, в программе всегда можно открыть диалоговое окно, показанное на рис. 9. В данном окне пользователь может изменять по своему усмотрению стиль и цветовую гамму графического отображения рабочих окон программы, а также масштаб отображения измеряемых величин на выводимых программой графиках. Производимые здесь изменения не влияют на числовые результаты работы программы. Для восстановления стиля

графики, первоначально установленной изготовителем, следует воспользоваться дополнительной кнопкой **«По умолчанию»**.

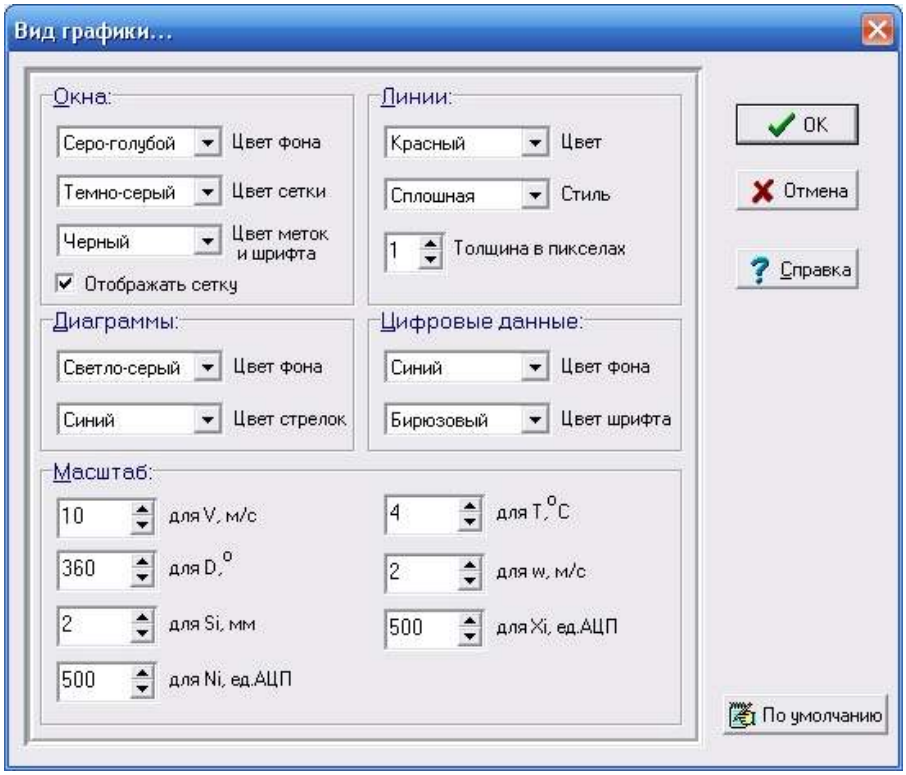

Рис. 9

4 ВЫПОЛНЕНИЕ ОСНОВНЫХ ФУНКЦИЙ

4.1 Измерение метеорологических параметров

4.1.1 Запуск программы на выполнение измерений метеорологических величин осуществляется командой меню «**Работа / Измерения…F9**». Он также может быть выполнен с помощью кнопки “**Старт**”, расположенной в главном окне программы, если в этом же окне в группе радиокнопок “**Режим**“ выбрана опция “**Измерения**”.

По данной команде вначале открывается диалоговое окно **«Параметры измерений…»** (см. рис. 10), в котором пользователь должен установить порядок выполнения запускаемых измерений.

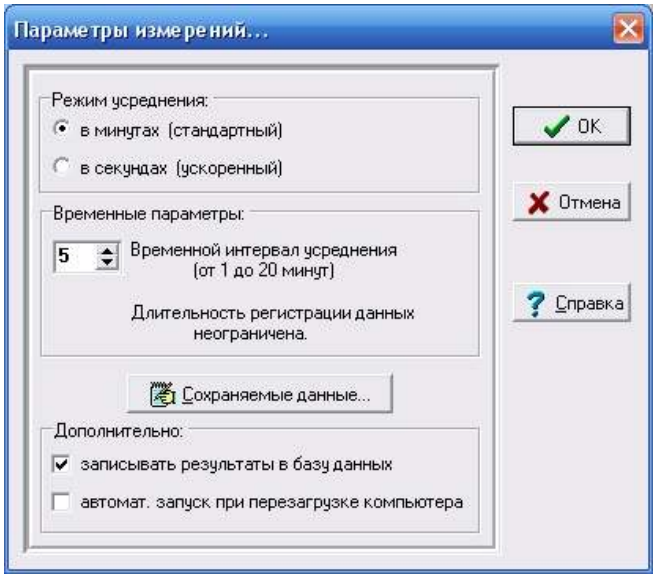

Рис. 10а

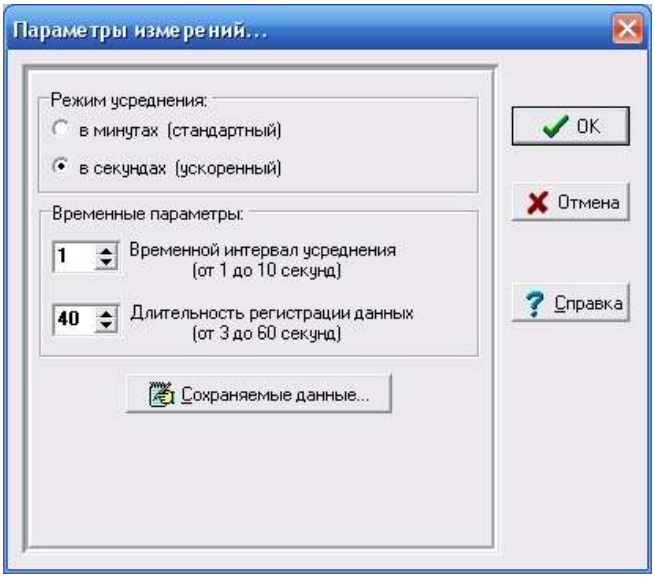

Рис. 10б

4.1.2 В программе «**МЕТЕО 3.0**» предусмотрено два режима выполнения метеорологических измерений, отличающиеся между собой временными параметрами обработки измеряемых данных: стандартный и ускоренный. Стандартный режим (основной) предназначен для организации непрерывных и неограниченных заранее по времени измерений метеорологических величин, что удобно для применения изделия на стационарных метеорологических постах, выполняющих длительные автоматические измерения без участия оператора с накоплением их данных на жестком диске компьютера. Ускоренный режим (дополнительный) предназначен кратковременного контроля метеорологических условий при исследованиях быстропротекающих процессов в атмосфере.

В стандартном режиме временной интервал усреднения при вычислениях усредненных атмосферных параметров задается в минутах (в пределах от 1 до 20), при этом длительность измерений (регистрации данных) не оговаривается (см. рис. 10а). Измерения в этом случае останавливаются только после нажатия оператором кнопки «**Стоп**» в главном окне программы (см. п. 2.3.3) или когда выполняется завершение работы самой программы.

В ускоренном режиме интервал усреднения задается уже в секундах (в пределах от 1 до 10), а длительность измерений устанавливается оператором до начала их выполнения в пределах от 3 до 60 секунд, при этом эта длительность не может быть меньше интервала усреднения (см. рис. 10б). В этом случае программа автоматически завершает измерения по истечению установленной длительности с момента нажатия кнопки «**ОК**» в окне **«Параметры измерений…»**.

4.1.3 Для обоих режимов в окне **«Параметры измерений…»** функционирует кнопка «**Сохраняемые данные…**», нажатие которой открывает дополнительное диалоговое окно с тем же заголовком (см. рис. 11). Данное окно позволяет пользователю оперативно, при каждом запуске измерений, контролировать и/или устанавливать новые параметры программы для сохранения ею

выходных данных. Ввод этих параметров также может быть выполнен в многостраничном окне «**Настройки…**» (см. подразд. 3.3), т.е. здесь он дублируется в более компактном и доступном месте. Детально о сохраняемых программой выходных данных следует смотреть в разделе 7.

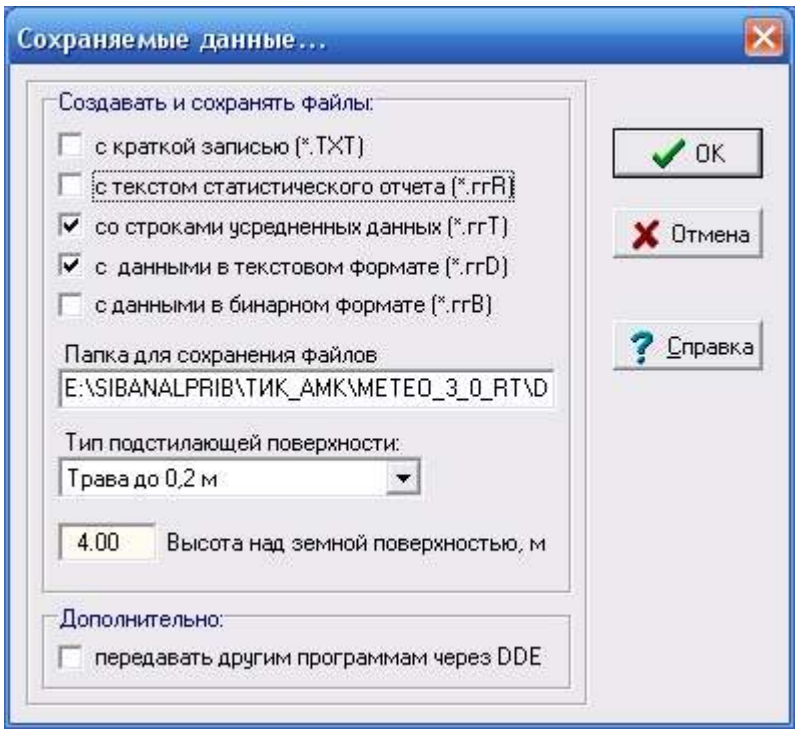

Рис. 11

4.1.4 В диалоговом окне **«Параметры измерений…»** при выборе стандартного режима усреднения (в минутах) также можно включить 2 дополнительные опции работы программы во время измерений (см. рис. 10а).

Опция «**Записывать результаты измерений в базу данных»** функционирует только при установке в одну папку с «**METEO 3.0**» приложения «**METEO DB**» и позволяет включать/отключать запись выходных данных программы в **базу данных** (файлы базы данных поставляются совместно с «**METEO DB**»).

Включение второй опции («**автомат. запуск при перезагрузке компьютера**») разрешает «**METEO 3.0**» временно, на период выполнения ею измерений, зарегистрироваться в реестре Windows как программы, которая должна автоматически загружаться в Windows при каждом запуске этой операционной системы. При спонтанной перезагрузке Windows, например, при случайном сбое электропитания ПЭВМ, программа «**METEO 3.0**» в случае включенной данной опции самостоятельно вновь загрузится в ПЭВМ и автоматически продолжит выполнение ранее прерванной обработки данных изделия как метеорологических измерений с установленными для них параметрами пользователя.

Обратная процедура удаления «**METEO 3.0**» из списка автоматически загружаемых программ в Windows выполняется также самой «**METEO 3.0**» при инициализации лично пользователем завершения работы программы в режиме измерений.

4.1.5 При обработке данных измерений изделия программа постоянно накапливает в оперативной памяти ПЭВМ все вычисляемые ею мгновенные значения метеорологических величин. При этом для каждой вычисляемой метеорологической величины программа формирует отдельную выборку выходных данных, дискретизируемую на временной шкале по моменту регистрации Windows входных данных изделия (на используемом порту ПЭВМ). Как только длительность регистрации входных данных изделия начинает превышать заданное оператором время усреднения, программа начинает удалять в указанных выборках самые ранние числовые данные с добавлением вместо них (в конец выборок) вновь поступающих данных. Соответственно, объем данных в со-

храняемых в оперативной памяти ПЭВМ статистических выборках мгновенных значений метеорологических величин не может превышать объем данных, который можно накопить от применяемого изделия за установленное время усреднения.

4.1.6 Если в программе перед запуском измерений со стандартным режимом усреднения (в минутах) была включена опция записи файла «**с данными в бинарном формате**» (файла с расширением *.ггB, см. рис. 11 и рис. 5, а также разд. 7), то программа через каждые 30 секунд измерений будет записывать вновь полученные мгновенные значения метеорологических величин в указанный файл (примерно по 312 килобайт при частоте измерений 80 Гц или пропорционально другое число байт при иной частоте). Как только время записи этого файла превысит установленное оператором время усреднения, программа, не приостанавливая измерения, создаст новый аналогичный файл, в котором она продолжит запись мгновенных значений метеорологических величин, не пропустив ни одного значения. Данный режим сохранения данных предназначен, главным образом, для записи длинных, в том числе круглосуточных, рядов мгновенных значений метеорологических величин для научных исследований атмосферной турбулентности. Бинарные файлы могут считываться и обрабатываться с помощью Windows-приложения **«METEO DP»**, входящего в состав программного комплекса «**Ультразвуковая метеостанция**» или же собственной программой пользователя, учитывающей формат записи этих файлов. Информация о формате и структуре записи данных файлов содержится в разд. 7.

В случае запуска измерений с ускоренным режимом усреднения (в секундах) значение опции записи бинарных файлов игнорируется, при этом данные файлы никогда не записываются. В этом режиме аналогично не записываются файлы «**с краткой записью**» с расширением *.TXT (также см. рис. 4, рис. 11 и разд. 7).

В данном случае могут быть записываться только файлы с расширениями *.ггR, *.ггT и *.ггD (см.рис. 4).

4.1.7 Если при запуске измерений со стандартным режимом усреднения (в минутах) была включена хотя бы одна из опций автоматической записи файлов с расширениями *.ггR, *.ггT и *.ггD или же опция записи в **базу данных**, то после нажатия кнопки «**OK**» в окне **«Параметры измерений…»** дополнительно откроется диалоговое окно **«Сроки регистрации данных…»** (рис. 12). В этом окне пользователь должен задать временной регламент сроков метеорологических наблюдений, в которые должны выполняться записи результатов измерений в указанные файлы и/или в **базу данных**.

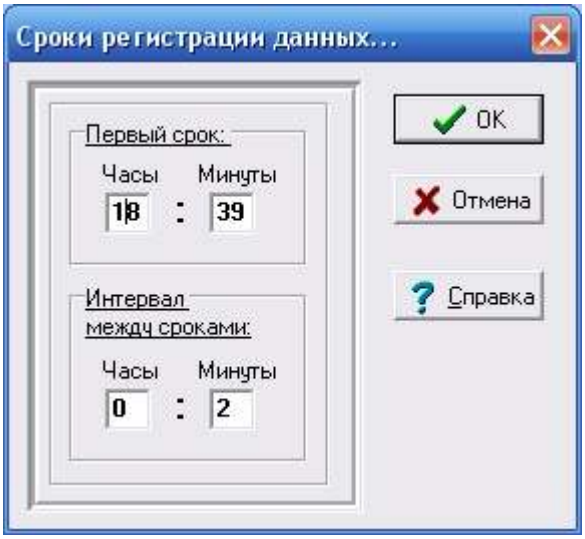

Рис. 12

Временной регламент должен устанавливаться по компьютерным часам. При этом задаются время первого срока (в пределах суток от текущего момента времени), в который программа должна выполнить первую регистрацию результатов измерений, и интервал между сроками (он также не может превышать 24 часа). Время первого срока наблюдений следует задавать больше фактического времени, иначе первая запись результатов измерений произойдёт почти через сутки. Интервал между сроками наблюдений не должен быть меньше времени усреднения, ранее установленного в диалоговом окне **«Параметры измерений…»**. Записываемые в каждый срок результаты вычисляются по данным изделия, считанным в компьютер за интервал времени, равный установленному в программе времени усреднения и заканчивающийся в текущий срок. Регистрация результатов измерений также дополнительно выполняется и в момент завершения измерений при нажатии кнопки **«Стоп»** в главном окне программы. Одним из достоинств непрерывности измерений изделием является возможность регистрировать значения метеорологических параметров не только для установленных сроков наблюдений, но и для интервалов между соседними сроками. Для этих интервалов программа дополнительно регистрирует минимальные и максимальные значения средней температуры и мгновенной (на самом деле с двухсекундным усреднением) скорости ветра.

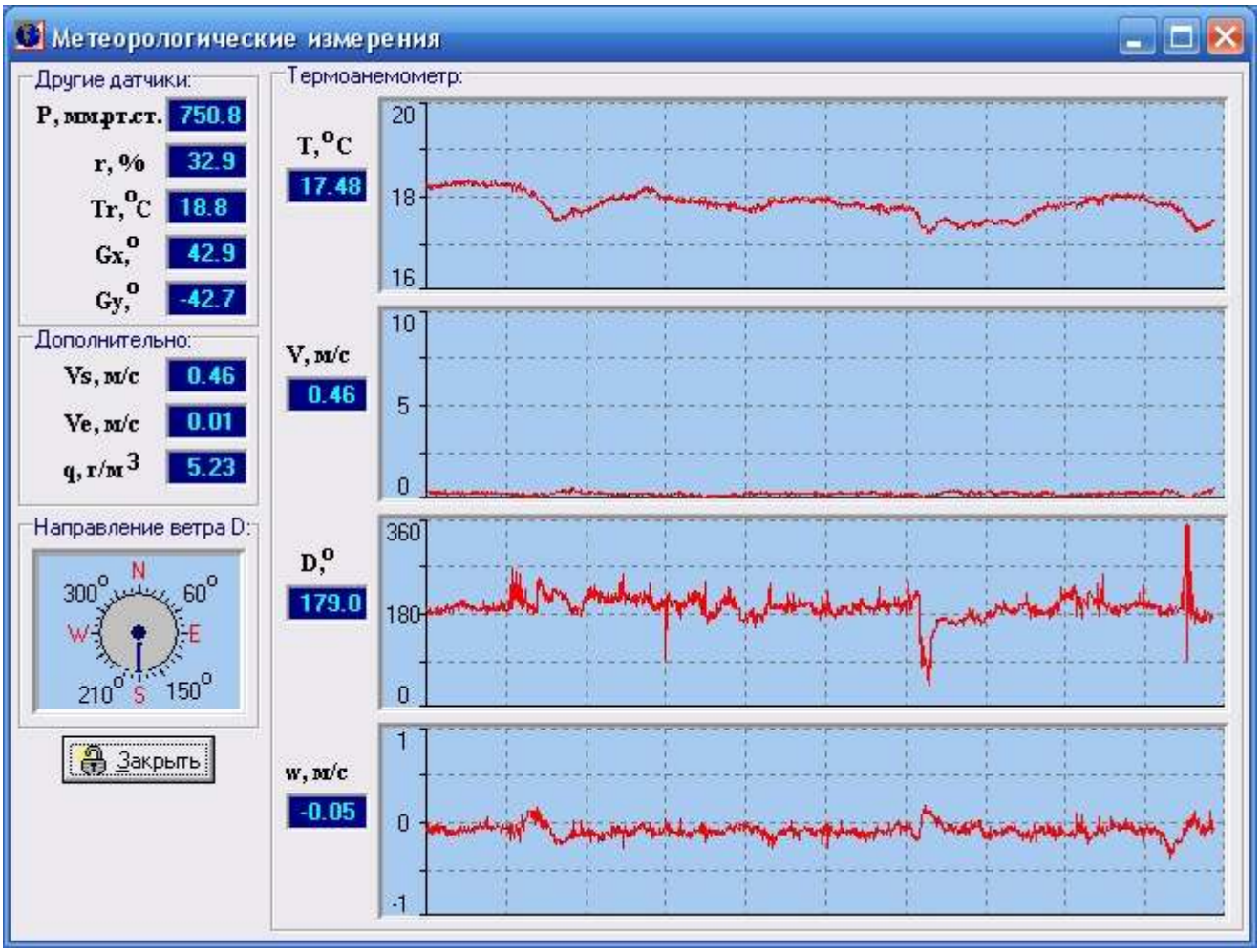

Рис. 13

При запуске измерений с ускоренным режимом усреднения (в секундах) диалоговое окно **«Сроки регистрации данных…»** никогда не появляется. В этом случае время первого срока записи файлов программа принимает равным моменту запуска измерений с добавлением к нему установленного интервала усреднения, а интервал между сроками равным интервалу усреднения. Например, если был введен интервал усреднения в 1 секунду, то файл **«со строками усредненных**

данных» с расширением *.ггТ будет пополняться новой строкой данных через каждую секунду и по завершению измерений число строк данных в нем совпадет с длительностью измерений в секундах.

4.1.8 Если на главном окне программы с помощью радиокнопок выбран режим «**Измерения**», то при нажатии в этом же окне кнопки «**Графика…**» в правом верхнем углу экрана появится окно «**Метеорологические измерения**» (рис. 13). Данное окно предназначено для более детального отображения процесса измерений мгновенных значений метеорологических величин. При выполнении метеорологических измерений здесь в числовой и графической форме выводятся мгновенные значения температуры (в окне она обозначена как T, ^{0}C), скорости горизонтального ветра (V, м/с), направления горизонтального ветра (D, ^{0}), вертикальной компоненты скорости ветра (w, м/с). Кроме того, только в числовой форме также выводятся значения атмосферного давления (P, мм рт.ст.), относительной влажности воздуха (r, %), температуры (Tr, ^{0}C), измеренной терморезистором в датчике влажности, углов наклона термоанемометра от вертикали в продольном (Gx, ^{0}) и поперечном (Gy, ^{0}) направлениях относительно выделенной оси термоанемометра, а также значения абсолютной влажности воздуха (q, г/м^3), южной (Vs, м/с) и восточной (Ve, м/с) компонент скорости горизонтального ветра. Для направления ветра дополнительно отображается круговая графическая диаграмма.

4.1.9 Во время длительных измерений может возникнуть потребность использовать компьютер для работы с другими программами. В этом случае удобно воспользоваться командой меню «**Окно/Минимальное…Alt+0**». По данной команде главное окно приложения становится невидимым на экране монитора, закрываются все дочерние окна программы, и на экране в правом верхнем углу появляется минимальное окно программы (рис. 14). Указанная команда также может быть исполнена с помощью кнопки “**Минимально…”** на главном окне приложения.

Рис. 14

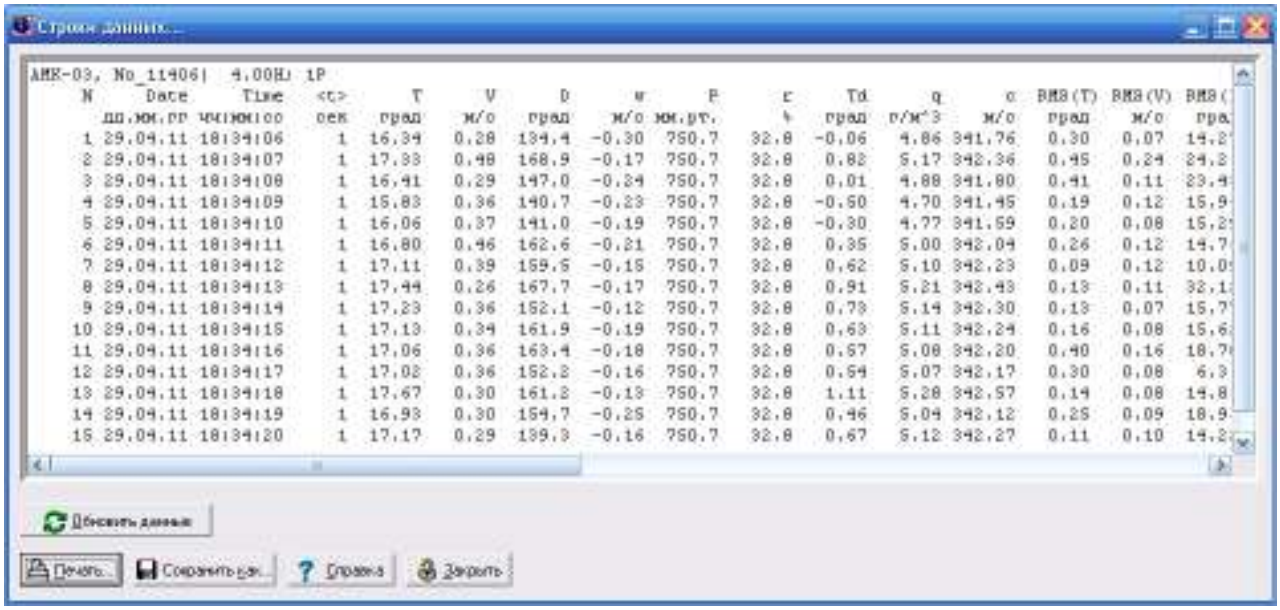

Рис.15

4.1.10 При измерениях по истечению срока первой регистрации данных для пользователя становится доступной команда «**Работа / Строки данных…F4**», дублируемая в главном окне программы дополнительной кнопкой «**Данные…**».

С ее помощью открывается окно «**Строки данных…**» (см. рис. 15), в котором пользователю предлагается просмотреть результаты метеорологических измерений в виде массива строковых записей данных. В каждой строке здесь представляются данные, усредненные за время, ранее установленное оператором. При этом периодичность между записями этих строк соответствует установленному интервалу между сроками регистрации данных. Содержащаяся в окне информация полностью идентична по структуре и содержанию записываемому программой файла **со строками усредненных данных** (с расширением *.ггТ, см. подразд. 7.5).

Окно «**Строки данных…**» содержит кнопки «**Печать**» (для печати информации окна на принтере), «**Сохранить как**» (для сохранения информации окна на жестком диске компьютера в виде текстового файла), «**Справка**» и «**Закрыть**». Имеющаяся здесь дополнительная кнопка «**Обновить данные**» позволяет во время измерений обновлять информационное содержание окна, не используя для этого его повторное открытие.

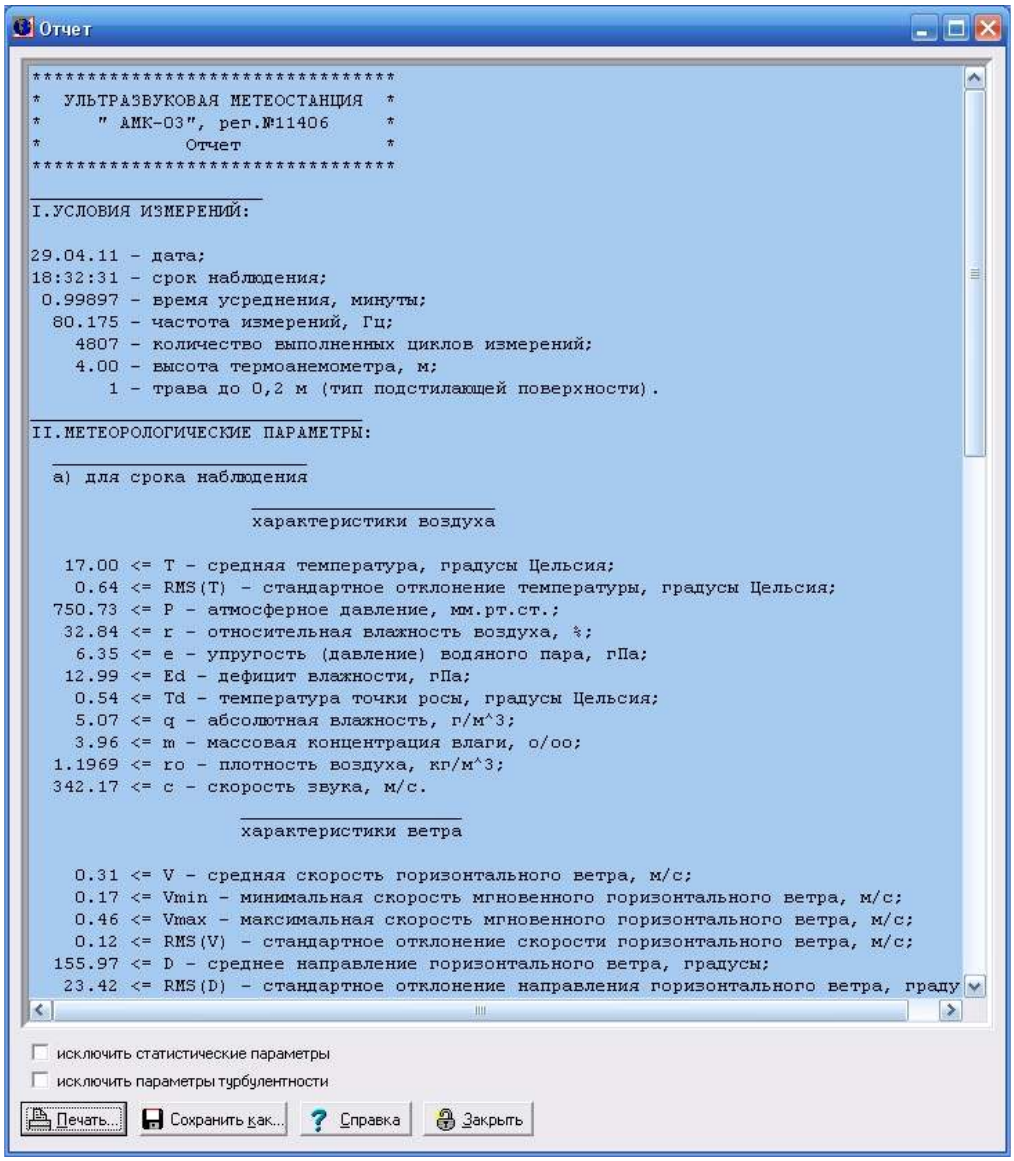

Рис. 16

4.1.11 По завершению установленного времени усреднения с момента запуска измерений или по завершению измерений для пользователя также становится доступной команда «**Работа / Отчёт…F5**», дублируемая в главном окне программы дополнительной кнопкой «**Отчет…**».

С ее помощью открывается окно «**Отчёт**» (см. рис. 16), которое содержит статистический отчет об измерениях, выполненных к текущему моменту времени за интервал времени, равный времени усреднения. Содержащаяся в окне информация полностью идентична по структуре и содержанию формируемого программой файла **с текстом статистического отчета** (с расширением *.ггR, см. подразд. 6.6).

Окно «**Отчет**» содержит кнопки «**Печать**» (для печати информации окна на принтере), «**Сохранить как**» (для сохранения информации окна на жестком диске компьютера в виде текстового файла), «**Справка**» и «**Закрыть**». Имеются также две опции, позволяющие исключать из текста отчета раздел, содержащий статистические характеристики измеренных метеорологических величин, и раздел, содержащий значения параметров атмосферной турбулентности (независимо друг от друга).

4.2 Градуировка ультразвукового термоанемометра

4.2.1 Для минимизации систематических погрешностей ультразвуковых измерений температуры воздуха, скорости и направления ветра в изделии предусмотрена специальная градуировка термоанемометра. С помощью программы **«МЕТЕО 3.0»** она автоматизирована так, что может выполняться даже неквалифицированным пользователем.

4.2.2 Суть автоматизации градуировки термоанемометра, входящего в состав изделия, заключается в измерении с помощью программы расстояний S_i (i =1, 2, 3, 4) между четырьмя парами его ультразвуковых датчиков при температуре воздуха $T_к$, предварительно измеренной эталонным термометром. Измерение этих величин выполняется также акустическим методом, используя поступающие данные изделия и введённое в программу значение температуры воздуха $T_к$ в момент градуировки. Измеренные при градуировке технические параметры термоанемометра S_i (i =1, 2, 3, 4) и значение температуры $T_к$, при которой они получены, после дополнительного подтверждения оператором запоминаются программой и отображаются им для чтения на странице «**Термоанемометр**» окна «**Настройки…**». В дальнейшем они используются программой как градуировочные параметры изделия при вычислении значений метеорологических параметров.

Внимание! *До выполнения запуска градуировки необходимо предварительно поместить термоанемометр и эталонный термометр в закрытый от ветровых потоков контейнер (камеру), снять показание эталонного термометра и включить питание изделия.*

4.2.3 Запуск программы на выполнение градуировки термоанемометра осуществляется командой меню «**Работа / Градуировка…F8**».

Он также может быть выполнен с помощью кнопки “**Старт**”, расположенной в главном окне программы, если в этом же окне с помощью группы радиокнопок “**Режим**“ выбрана опция “**Градуировка**”.

Для исполнения этой команды меню оператор предварительно должен получить в программе доступ к праву изменения градуировочных параметров изделия, для чего ему потребуется ввести административный пароль (см. рис. 17), записанный в паспорте изделия. Это обусловлено необходимостью защиты метрологических характеристик изделия от случайных действий пользователя. Если после загрузки программы указанный доступ оператором уже был получен ранее, например, при работе в диалоговом окне «**Настройки…**» (см. п. 3.2.6), то повторный

запрос пароля программа не делает. При повторной загрузке программы его ввод снова потребуется.

Дополнительно, во всех случаях, когда оператор пытается изменить градуировочные параметры изделия, программа также открывает диалоговое окно с запросом подтверждения у оператора осмысленности его действия (см. рис. 18).

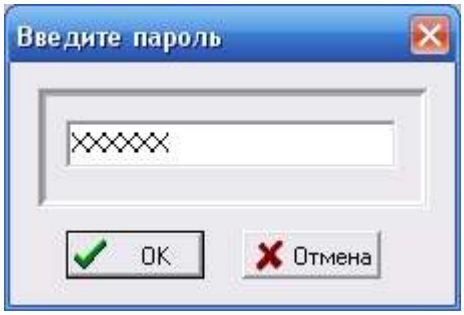

Рис. 17

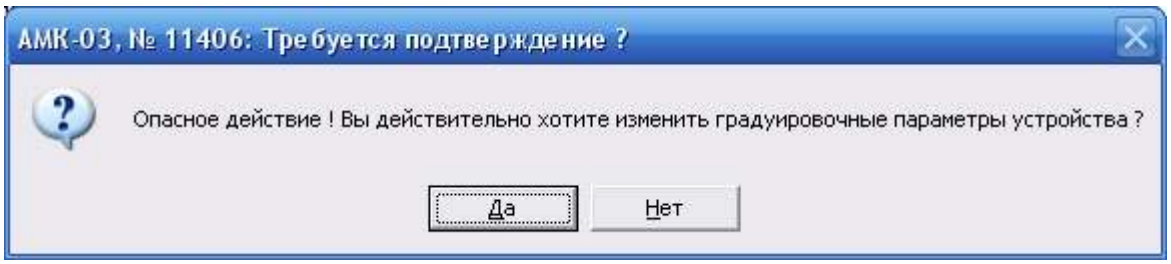

Рис. 18

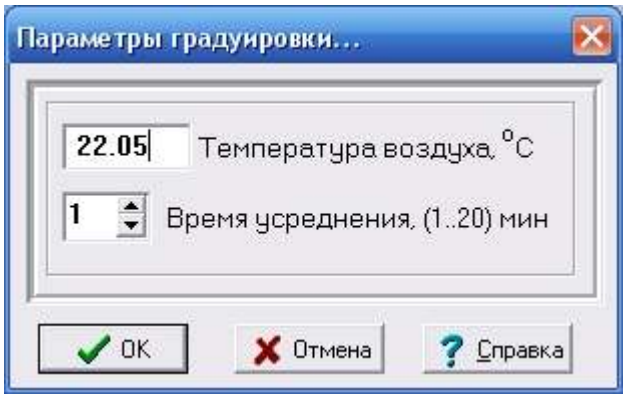

Рис. 19

4.2.4 При исполнении команды меню, указанной в п. 4.2.3, программа вначале открывает диалоговое окно **«Параметры градуировки…»** (рис. 19), в котором оператор должен ввести с клавиатуры значение температуры воздуха в контейнере, используя при этом показание эталонного термометра, и задать время усреднения результатов измерений в минутах (стандартное значение – 1 минута).

После нажатия в этом окне кнопки «**ОК**» программа будет выполнять градуировочные измерения непрерывно и неограниченное время, которые останавливаются только нажатием кнопки «**Стоп**» в главном окне программы. По завершению градуировки программа вычисляет значения градуировочных параметров термоанемометра только по последним данным измерений, полученным за период, равный установленному времени усреднения. При этом программа снова выдает запрос оператору (см. рис. 20) на целесообразность изменения градуировочных

параметров. Если в ответ на него оператор даст подтверждение, то она запомнит вычисленные при градуировке параметры для использования их в последующих метеорологических измерениях. Если оператор сомневается в качестве проведенной им градуировки, то он должен всегда отказываться от сохранения ее результатов.

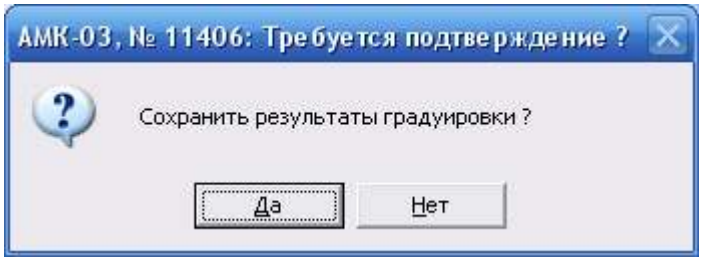

Рис. 20

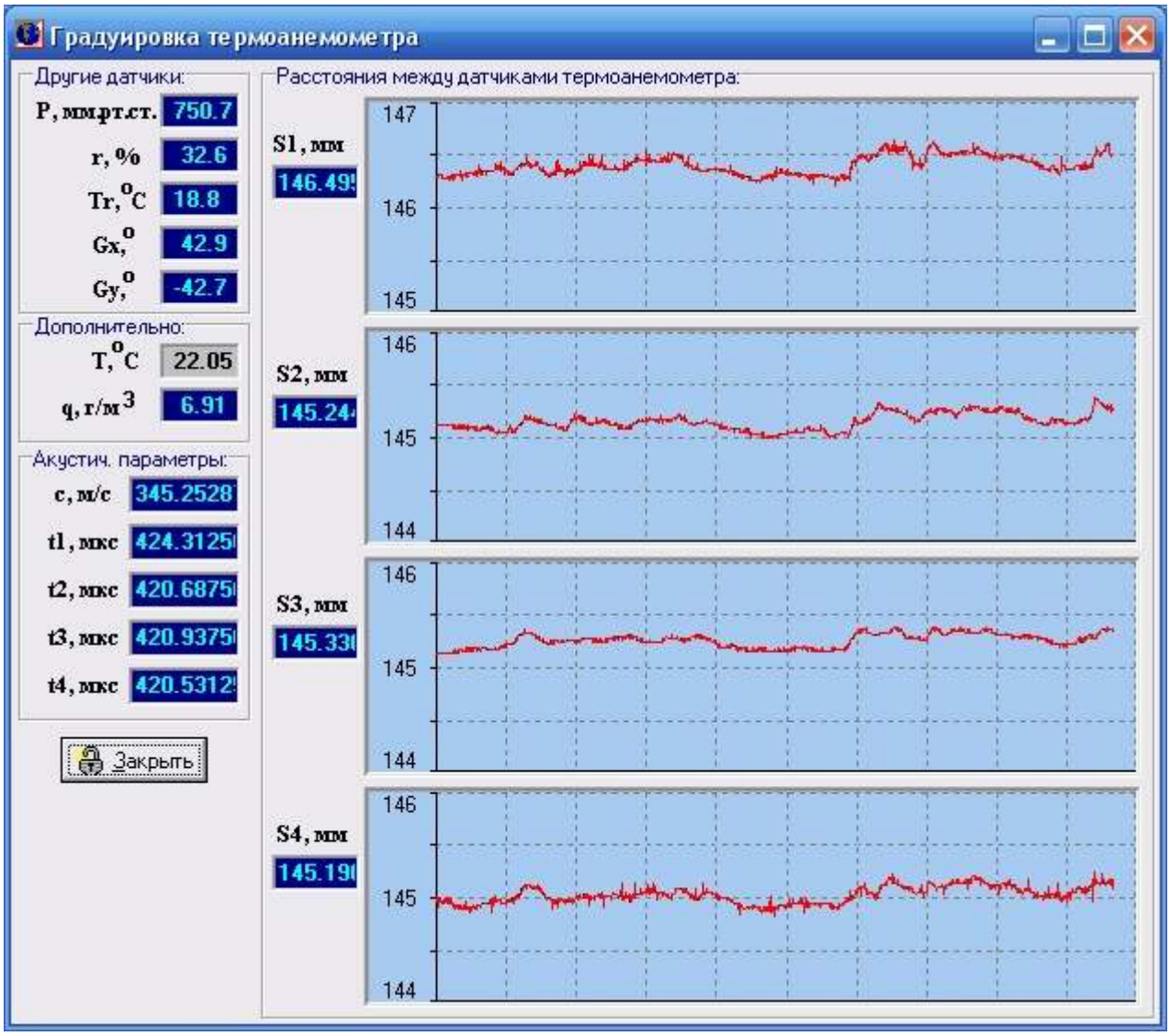

Рис. 21

4.2.5 Если в главном окне приложения радиокнопками выбран режим «**Градуировка**», то при нажатии на панели «**Работа**» кнопки «**Графика…**» на экране появится окно «**Градуировка термоанемометра**» (рис. 21). Данное окно предназначено для отслеживания качества выполнения градуировки ультразвукового термоанемометра. Здесь в числовой и графической форме выводятся

мгновенные значения измеряемых величин (расстояний Si в мм). При качественной градуировке графические линии должны быть горизонтальными прямыми, что указывает на отсутствие ветровых потоков в контейнере и аппаратных сбоев в работе изделия. На рис. 21 показан пример градуировки в комнатных условиях без защитного контейнера (при наличии слабых ветровых потоков около термоанемометра, но без его аппаратных сбоев в работе). Результаты подобной градуировки сохранять не следует.

В этом окне также (только в числовой форме) выводятся значения атмосферного давления P, относительной влажности воздуха r, температуры на корпусе датчика влажности Tr, углов наклона термоанемометра от вертикали Gx и Gy, абсолютной влажности q воздуха, скорости звука *c*, временных интервалов t_i, (в мкс) прохождения звуком расстояний между ультразвуковыми датчиками термоанемометра.

4.3 Контроль первичных данных изделия

4.3.1 Запуск программы на выполнение контроля первичных данных изделия выполняется командой меню «**Работа / Контроль…F7**». Он также может быть выполнен с помощью кнопки “**Старт**”, расположенной в главном окне программы, если в этом же окне с помощью группы радиокнопок “**Режим**“ выбрана опция “**Контроль**”. При этом программа не требует вводить какие-либо параметры исполнения команды.

Данный режим измерений предназначен, главным образом, для изготовителя изделия - для тестирования им технического состояния устройства в целом и его составных частей (отдельных датчиков) на таких стадиях как изготовление термоанемометра и/или поиск его неисправностей при ремонте. При работе программы в этом режиме ею выполняется считывание в ПЭВМ первичных данных изделия и их отображение на экране монитора. Этот процесс также выполняется непрерывно и неограниченное заранее время. Он останавливается только нажатием кнопки «**Стоп**» в главном окне программы.

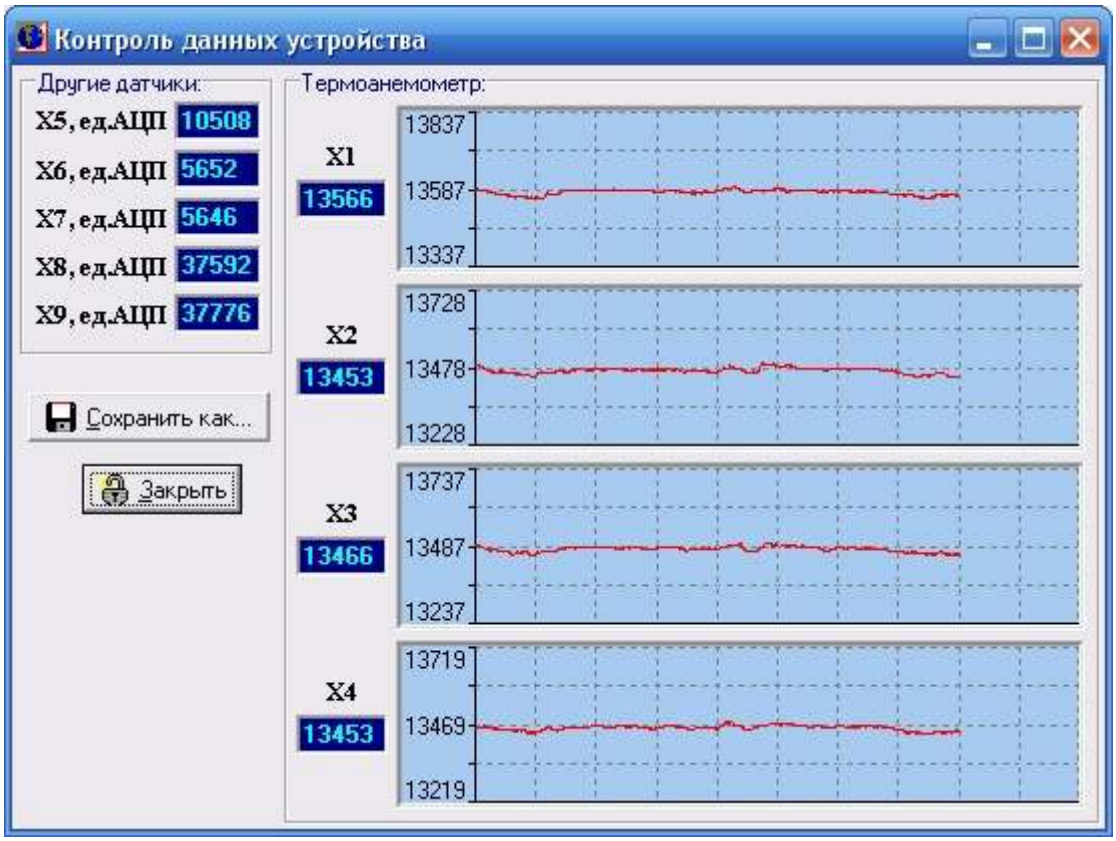

Рис. 22

4.3.2 Если в главном окне приложения радиокнопками выбран режим «**Контроль**», то при нажатии на панели «**Работа**» кнопки «**Графика…**» на экране появится окно «**Контроль данных устройства**» (рис. 22). Здесь в числовой и графической форме выводятся текущие показания термоанемометра X1-X4 (для его четырех каналов измерений), только в цифровой форме – показания датчика давления X5, датчика влажности X6, терморезистора на корпусе датчика влажности X7 и показания датчика углов наклона термоанемометра X8 и X9. Если датчик давления или датчик влажности на странице **«Программа»** в окне «**Настройки…**» (см. рис. 4) отмечен как не подключенный, то его показания здесь не отображаются.

Данное окно имеет кнопку «**Сохранить как…**», которая позволяет записать в текстовый файл массив первичных данных изделия, считанных программой за время усреднения, последний раз установленное в ней в минутах (при запуске метеорологических измерений или градуировки термоанемометра).

4.4 Чтение/запись градуировочных параметров во внешний пульт

4.4.1 В изделиях **АМК-03** дополнительно может применяться пульт управления **ППУ-25**.

Пульт **ППУ-25** производит вычисление и индикацию значений основных метеорологических величин с задаваемым временем усреднения, что позволяет эксплуатировать изделие автономно (без ПЭВМ).

Вычисления выполняются во встроенном микропроцессоре зашитой в нем программой, используя первичные данные изделия в стандарте RS-485 и значения его градуировочных параметров, хранимых им в энергонезависимой памяти. Для правильности этих вычислений необходимо, чтобы значения градуировочных параметров изделия, хранимые в памяти пульта, соответствовали используемым в изделии датчикам.

4.4.2 В программе **«МЕТЕО 3.0»** имеется команда меню «**Сервис/Пульт…**», которая открывает специальное модальное окно (рис. 23), предназначенное для настройки пульта. Используя данное окно, можно считывать значения градуировочных параметров из памяти пульта, сравнить их со значениями в программе и при необходимости записать в пульт новые значения. При использовании этого окна пульт предварительно должен быть соединен через порт COM с ПЭВМ, на которой установлена программа.

В данном окне содержится таблица из 5-и колонок. В первой колонке (с заголовком «**N**») указаны порядковые номера градуировочных параметров; во второй (с заголовком «**Константа**») – соответствующие условные обозначения (идентификаторы) градуировочных параметров; в третьей (с заголовком «**В пульте**») – значения этих параметров, считанные из памяти пульта; в четвертой (с заголовком «**В программе**») – значения этих параметров, используемые в **«МЕТЕО 3.0»**; в пятой (с заголовком «**Новые**») – введенные пользователем новые значения, которые он предполагает записать в память пульта. Первые две колонки таблицы и колонка «**В программе**» заполняются программой автоматически при открытии данного окна.

4.4.3 Имеющаяся в окне кнопка «**Чтение**» предназначена для выдачи команды программе на чтение из пульта значений градуировочных параметров. Каждый из параметров считывается отдельно с одновременной диагностикой качества выполнения этой операции. Результаты заносятся в колонку «**В пульте**» в виде комбинации цифр, описывающих значение параметра и через пробел со звездочкой код ошибки. Возможные значения кода ошибки и их текстовые расшифровки приводятся непосредственно на поле окна (см. рис. 23).

4.4.4 Нажатие кнопки «**Новые**» выдает команду программе скопировать значения всех градуировочных параметров из колонки «**В программе**» в колонку «**Новые**». Пользователь также может, используя клавиатуру и мышь, редактировать или вводить свои значения этих параметров в колонке «**Новые**».

4.4.5 Кнопка «**Запись**» выдает команду на запись в пульт введенных в колонке «**Новые**» значений градуировочных параметров. Выполнение записи новых значений градуировочных параметров в пульт производится только после ввода пароля, указанного в паспорте изделия (аналогично, как в п.4.2.3). Если поле для какого-либо параметра в колонке «**Новые**» будет пустым, то запись параметров в пульт также не производится. Записанные значения параметров с результатами диагностики качества операций отображаются в колонке «**В пульте**». Для окончательной проверки записанных в пульт значений параметров рекомендуется выполнить их повторное чтение из пульта с помощью кнопки «**Чтение**».

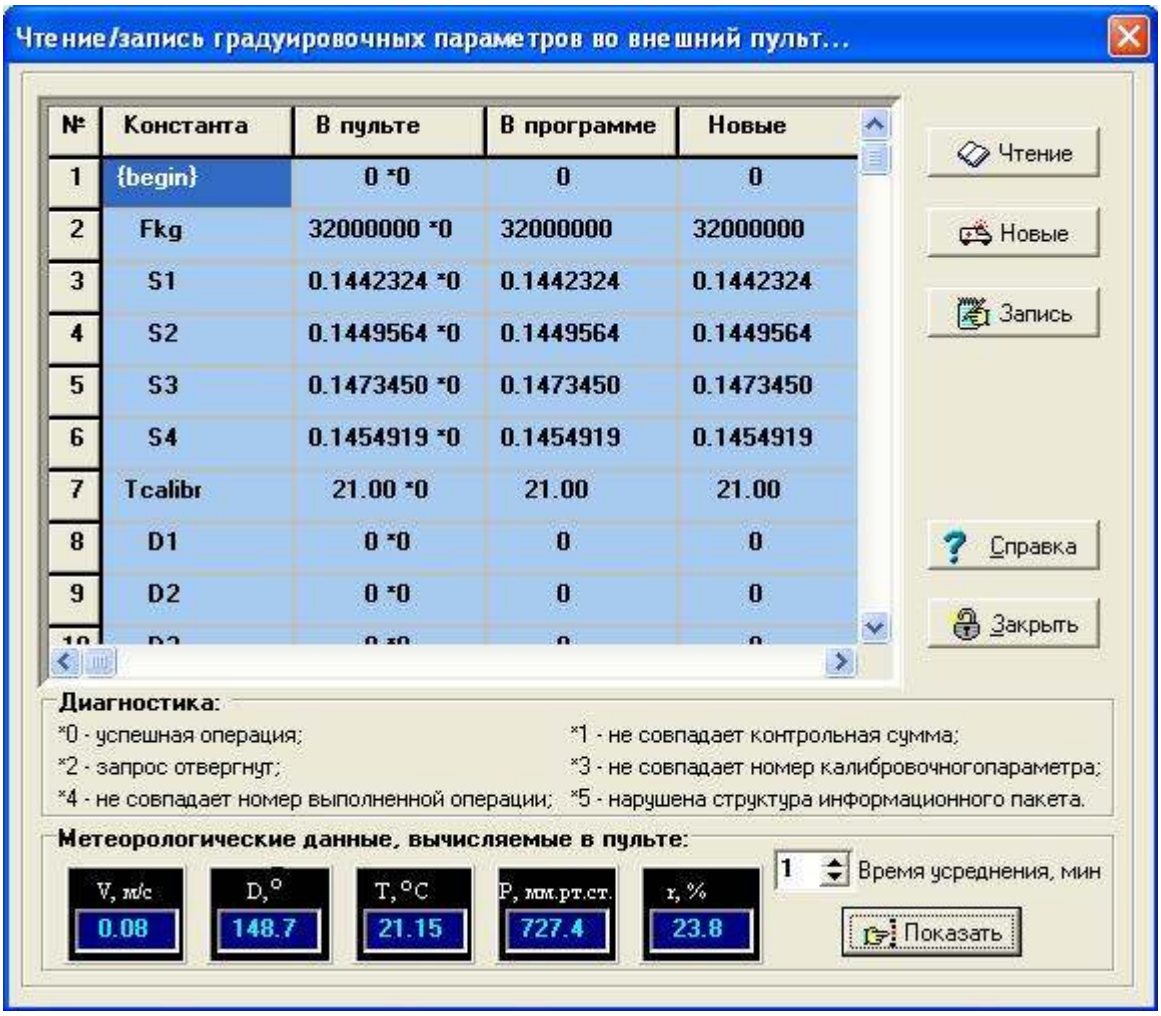

Рис. 23

4.4.6 Внизу окна имеется панель с заголовком «**Метеорологические данные, вычисляемые в пульте**». Она содержит 5 полей для вывода значений скорости (V, м/с) и направления (D, ^{0}) горизонтального ветра, температуры (T, ^{0}C), давления (P, мм рт.ст.) и относительной влажности (r, %), а также поле выбора времени усреднения для вычислений, производимых в пульте, и кнопку «**Показать**». При нажатии этой кнопки программа **«МЕТЕО 3»** выполняет однократный запрос в пульт на выдачу вычисленных им в текущий момент значений перечисленных метеорологических параметров с установленным в окне временем усреднения, а затем отображает их на данной панели. Данную команду можно повторять многократно, наблюдая как изменяются во времени значения вычисляемых в пульте метеорологических параметров.

4.4.7 Кнопка «**Закрыть**» позволяет закрыть данное окно и вернуться к работе в главном окне программы.

4.5 Выполнение тестирования внешнего пульта

4.5.1 Команда меню «**Сервис/Тестирование пульта…**» открывает модальное окно «**Тестирование пульта…**» (см. рис. 24), предназначенное для выполнения тестирования программного обеспечения, зашитого в микропроцессор пульта.

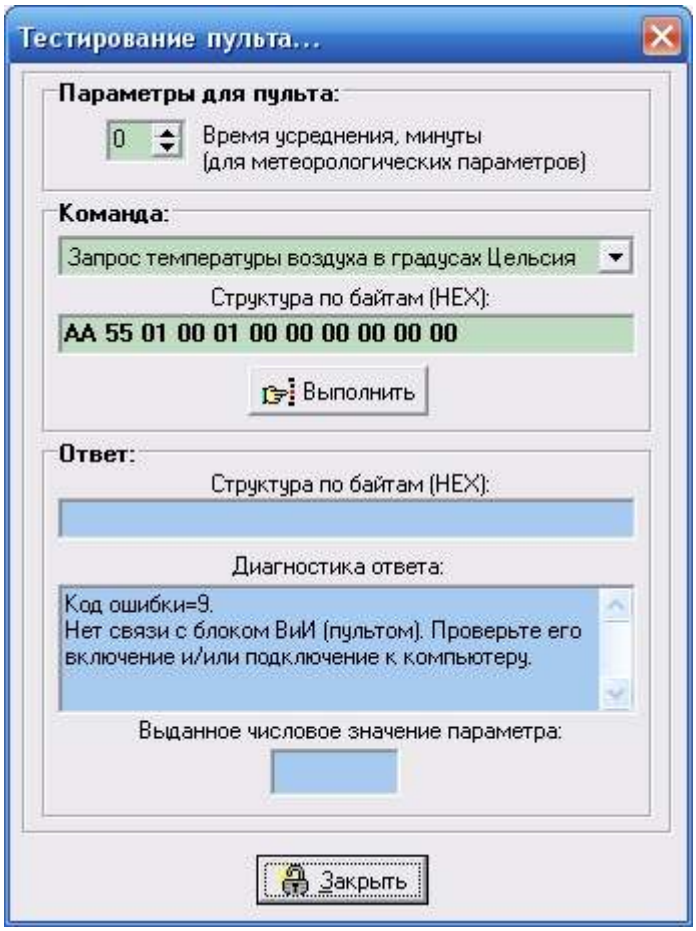

Рис. 24

Тестирование пульта здесь производится посредством передачи из ПЭВМ команд (запросов) установленного формата в микропроцессор пульта и в получении обратно в ПЭВМ его ответов с последующей проверкой ответов на их соответствие установленного для изделия протокола информационного сопряжения с ПЭВМ. Этим протоколом описываются все реализованные в пультах изделия типы команд, форматы их структуры по байтам, а также форматы соответствующих им ответов.

4.5.2 В окне «**Тестирование пульта…**» оператор может выбрать из выпадающего списка одну из стандартных команд для пульта. При этом ниже этого списка программа дополнительно отобразит структуру выбранной команды в виде последовательности 11-и байт (в шестнадцатеричном представлении).

При нажатии кнопки «**Выполнить**» выбранная команда в виде пакета указанных байтов будет отправлена из компьютера в пульт.

Далее программа ждет ответа пульта на посланную ему команду (не более 100 млсек). Полученный ответ также отображается в виде последовательности из 11-и байт. Дополнительно программа выполняет диагностику ответа (анализ на наличие ошибок в работе пульта), результаты которой выдаются в окне в виде текстового сообщения. В случае если выполненная команда является запросом значения какого-либо метеорологического параметра, вычисленного в пульте с заданным временем усреднения, то программа также отобразит возвращенное числовое значение запрошенного параметра.

5 ВЫЗОВ И ЗАГРУЗКА

5.1 Вызов программы

Вызов программы **«МЕТЕО 3.0»** на исполнение может осуществлятся любым стандартным для операционной системы Windows способом. Как правило, для этого необходимо 2 раза щелкнуть левой кнопкой мыши по имени основного исполняемого файла программы «**АМК_XXXXX.EXE**» или по его ярлыку, размещаемому на рабочем столе Windows программой автоматической установки **«Setup»**, поставляемой с дистрибутивом **«МЕТЕО 3.0»** на диске CD-ROM.

Кроме того, если какая либо другая программа пытается установить DDE-соединение с ранее не загруженной программой **«МЕТЕО 3.0»** для получения ее выходных данных по технологии «**Dynamic Data Exchange**», то вызов программой **«МЕТЕО 3.0»** происходит автоматически (более детально об этой технологии см. в подразд. 7.10).

5.2 Загрузка программы

Загрузка программы **«МЕТЕО 3.0»** происходит следующим образом. Вначале в центре экрана монитора на 3 сек появляется окно заставки программы **«МЕТЕО 3.0»**, содержащее наименование программного продукта и информацию о изготовителе изделия. Затем оно автоматически закрывается и в левом верхнем углу экрана монитора появляется главное окно программы (см. рис.2) с заголовком «**Ультразвуковая метеостанция (МЕТЕО 3.0): АМК-03, № XXXXX**», где **XXXXX** – цифры заводского номера изделия. После этого, используя команды меню, оператор может запуститьь в программе любую ее рабочую функцию (см. подразд. 4) или изменять ее настройки (см. подразд. 3).

6 ВХОДНЫЕ ДАННЫЕ

5.1 Основные входные данные

Основными входными данными для программы **«МЕТЕО 3.0»** являются информационные пакеты из 23-х байт с первичными данными измерений в ультразвуковом термоанемометре. Эти пакеты автоматически выдаются термоанемометром с момента его подключения к питанию и повторяются с частотой либо 10 Гц, либо 80 Гц (устанавливается при их изготовлении). Передача данных термоанемометра к ПЭВМ осуществляется по интерфейсу **RS-232**, реализуемым в ПЭВМ ее последовательными портами **COM**. При этом на ПЭВМ должны устанавливаться следующие параметры связи:

- скорость передачи данных – **57600** бит/с;
- формат передаваемых байтов – **8**, **N**, **1** (8 бит данных, контроль четности отсутствует, 1 стоповый бит);
- контроль потока данных отсутствует.

Каждый информационный пакет, поступающий в ПЭВМ от термонемометра, имеет структуру в соответствии с таблицей 3. Прием и обработка одного информационного пакета здесь соответствует получению в программе одного мгновенного значения для каждой из измеряемых метеорологических величин.

Таблица 3

Номера байтов	Назначение	Обозначение	Значения	Число байтов
1-2	Байты синхронизации пакета	-	AA 55 (Hex)	2
3-4	Значение измеренного времени в 1-ом канале термоанемометра (в числе синхроимпульсов кварцевого генератора)	X1	0..65535	2
5-6	Значение измеренного времени во 2-ом канале термоанемометра (в числе синхроимпульсов кварцевого генератора)	X2	0..65535	2
7-8	Значение измеренного времени в 3-ем канале термоанемометра (в числе синхроимпульсов кварцевого генератора)	X3	0..65535	2
9-10	Значение измеренного времени в 4-ом канале термоанемометра (в числе синхроимпульсов кварцевого генератора)	X4	0..65535	2
10-11	Показание датчика давления (в ед. АЦП)	X5	0..65535	2
12-14	Показание датчика влажности (в ед. АЦП)	X6	0..65535	2
15-16	Показание датчика углов наклона термоанемометра (в ед. АЦП)	X7	0..65535	2
17-18	Показание датчика углов наклона термоанемометра (в ед. АЦП)	X8	0..65535	2
19-20	Показание датчика углов наклона термоанемометра (в ед. АЦП)	X9	0..65535	2
21-22	Показание платинового терморезистора в датчике влажности (в ед. АЦП)	X10	0..65535	2
23	Контрольная сумма байтов (с 3 по 22)	-	0…255	1

5.2 Входные данные при использовании пульта ППУ-25

5.2.1 К ультразвуковому термоанемометру может подключаться пульт управления ППУ-25, который функционирует при подключении его к ПЭВМ в 2-х режимах.

В режиме «**Работа**» выполняется передача из ПЭВМ команд (запросов) в ППУ-25 одного из установленных ниже форматов и получение обратно в ПЭВМ ответов его микропроцессора. В режиме «**Калибровка**» ППУ-25 постоянно выдает в ПЭВМ информационные пакеты с первичными показаниями термоанемометра в соответствии с п.5.1 (т.е. в этом режиме ППУ-25 только ретранслирует данные термоанемометра на ПЭВМ).

Каждая команда для микропроцессора ППУ-25 имеет свой отличительный порядковый номер (см. табл.4). При этом должны использоваться параметры связи, указанные в п.5.1.

Значение каждого метеорологического параметра запрашивается отдельно, т.е. все метеорологические данные изделия можно получить только после исполнения поочередно команд с номерами от 01 по 05 Hex (в любом порядке). Запись и чтение значений градуировочных параметров изделия может осуществляться также только по одному, используя при обращении их отличительный порядковый номер (от 1 до 53). Значения всех метеорологических и градуировочных параметров возвращаются числом с плавающей запятой (32 разряда). Контрольная сумма байтов для значений всех градуировочных параметров является целым 2-х байтовым числом. Если в момент запроса метеорологического параметра микропроцессор блока **ВиИ** не готов выдать его значение, например, когда с момента включения изделия прошло времени меньше, чем требуемое время усреднения, то он возвращает минус 999 в формате с плавающей запятой.

Таблица 4

Номер команды (Hex)	Назначение команды
01	Команда запроса значения атмосферного давления P
02	Команда запроса значения температуры воздуха T
03	Команда запроса значения относительной влажности воздуха U
04	Команда запроса значения направления ветра Av
05	Команда запроса значения скорости ветра V
06	Резерв
07	Резерв
08	Команда переключения режима информационного обмена с ППУ-25
10	Резерв
11	Резерв
20	Команда на прерывание исполнения предыдущей команды запроса
40	Команда на запись в энергонезависимую память ППУ-25 значения одного градуировочного параметра изделия
64	Команда на запись в энергонезависимую память ППУ-25 контрольной суммы байтов для значений всех градуировочных параметров
80	Команда на чтение из энергонезависимой памяти ППУ-25 значения одного градуировочного параметра изделия

5.2.2 Для каждой команды и ответа всегда используются информационные пакеты одинакового объема из 11 байтов и с одинаковой логической структурой. При этом если какой-либо байт в конкретном пакете не используется, то он также передается (с нулевым значением 00 Hex). Вычисление контрольной суммы байтов, входящих в пакет, выполняется логической поразрядной операцией **xor** (исключающее ИЛИ).

Структура информационного пакета для команд запроса значений метеорологических параметров (с номерами с 01 по 05 Hex) приведена в таблице 5.

Таблица 5

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3-4	Время усреднения в секундах	1, 10, 60*1,…60*10	2
5	Номер команды (см. табл.4)	01, 02,…05 (Hex)	1
6	Не используется	00 (Hex)	1
7-10	Не используется	00 00 00 00 (Hex)	4
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

5.2.2 На запрос по п.5.2.2 изделие возвращает информационный пакет ответа, который имеет структуру, описанную в таблице 6.

Таблица 6

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3-4	Время усреднения в секундах	1, 10, 60*1,…60*10	2
5	Номер команды (см. табл.4)	01, 02,…05 (Hex)	1
6	Не используется	00 (Hex)	1
7-10	Значение метеорологического параметра	(Float)	4
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

Примечание: *Здесь и ниже во всех ответах на команды в 3-ем и 4-ом байтах может быть возвращено значение FF (Hex), что означает отказ блока ППУ-25 от исполнения команды (например, при несовпадении контрольной суммы байтов или при другом техническом сбое в связи или работе изделия).*

5.2.3 В энергонезависимой памяти ППУ-25 сохраняются значения градуировочных параметров изделия. Данные параметры определяют метрологические характеристики изделия и устанавливаются после его изготовления или ремонта с использованием эталонных средств измерений.

Структура информационного пакета для команды записи в энергонезависимую память ППУ-25 значения одного градуировочного параметра изделия (с 1 по 53) приведена в таблице 7. На эту команду ППУ-25 возвращает информационный пакет ответа (подтверждение исполнения), который имеет полностью идентичную структуру (как в таблице 7).

Таблица 7

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3-4	Резерв	00 00 (Hex)	2
5	Номер команды (см. табл.4)	40 (Hex)	1
6	Номер градуиров. параметра	1…53	1
7-10	Значение градуировочного параметра	(Float)	4
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

5.2.4 Структура информационного пакета для команды записи в энергонезависимую память ППУ-25 контрольной суммы байтов значений всех градуировочных параметров изделия приведена в таблице 8. На эту команду ППУ-25 возвращает информационный пакет ответа (подтверждение исполнения), который имеет полностью идентичную структуру (как в таблице 8).

Таблица 8

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3-4	Резерв	00 00 (Hex)	2
5	Номер команды (см. табл.4)	64 (Hex)	1
6	Номер параметра	54	1
7-8	Значение параметра	0…65535	2
9-10	Не используется	00 00 (Hex)	2
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

5.2.5 Структура информационного пакета для команды чтения в ПЭВМ из энергонезависимой памяти ППУ-25 значения одного градуировочного параметра изделия (с 1 по 53) приведена в таблице 9. На эту команду ППУ-25 возвращает информационный пакет ответа, который имеет структуру, приведенную в таблице 10.

Таблица 9

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3-4	Резерв	00 00 (Hex)	2
5	Номер команды (см. табл.4)	80 (Hex)	1
6	Номер градуиров. параметра	00…FF (Hex)	1
7-10	Не используется	00 00 00 00 (Hex)	4
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

Таблица 10

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3-4	Резерв	00 00 (Hex)	2
5	Номер команды (см. табл.4)	80 (Hex)	1
6	Номер градуиров. параметра	00…FF (Hex)	1
7-10	Значение градуировочного параметра	(Float)	4
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

Каждая запись в энергонезависимую память ППУ-25 новых значений градуировочных параметров изделия должна быть упорядоченной и полной, т.е. она должна производиться строго в том же порядке возрастания номеров параметров, как они записаны в окне программы, показанном на рис. 23, не пропуская ни одного из параметров, и завершаться записью контрольной суммы. Новые значения градуировочных параметров изделия начинают использоваться в ППУ-25 только после выключения и повторного включения его питания.

Чтение значений градуировочных параметров изделия может производиться в любом порядке и частично (аналогично запросам значений метеорологических параметров).

5.2.6 В таблице 11 приведена структура информационного пакета для команды переключения ППУ-25 в режим «**Калибровка**», а в таблице 12 - для команды переключения ППУ-25 в режим «**Работа**».

Таблица 11

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3	Значение команды	2	1
4	Резерв	00 (Hex)	1
5	Номер команды (см. табл.4)	8 (Hex)	1
6-10	Не используются	00 00 00 00 00 (Hex)	5
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

Таблица 12

Номера байтов	Назначение	Значения	Число байтов
1-2	Байты синхронизации пакета	AA 55 (Hex)	2
3	Значение команды	60	1
4	Резерв	00 (Hex)	1
5	Номер команды (см. табл.4)	8 (Hex)	1
6-10	Не используются	00 00 00 00 00 (Hex)	5
11	Контрольная сумма байтов (с 3 по 10)	0…255	1

7 ВЫХОДНЫЕ ДАННЫЕ

7.1 Список измеряемых и вычисляемых метеорологических параметров

Программа **«МЕТЕО 3.0»** вычисляет из первичных данных изделия следующую метеорологическую информацию.

А) для срока наблюдения

характеристики воздуха

*1) ***T*** - средняя температура воздуха, ^{0}С;
2) **σ[*T*]** - стандартное отклонение температуры, ^{0}С;
*3) ***P*** - атмосферное давление, мм рт.ст. / гПа;
*4) ***r*** – относительная влажность воздуха, %;
*5) ***e*** – упругость (давление) водяного пара, гПа;
*6) E_d – дефицит влажности, гПа;
*7) T_d – температура точки росы, ^{0}С;
8) ***q*** – абсолютная влажность воздуха, г/м^3;
9) ***m*** – массовая концентрация влаги, о/оо;
10) ***ρ*** – плотность воздуха, г/м^3;
11) ***c*** – скорость звука в воздухе, м/с;

характеристики ветра

*12) ***V*** – средняя скорость горизонтального ветра, м/с;
13) V_{min} – минимальная скорость мгновенного горизонтального ветра, м/с;
*14) V_{max} – максимальная скорость мгновенного горизонтального ветра, м/с;
15) **σ[*V*]** – стандартное отклонение скорости горизонтального ветра, м/с;
*16) ***D*** – среднее направление горизонтального ветра, градусы;
17) **σ[*D*]** – стандартное отклонение направления ветра, м/с;
18) ***w*** – средняя скорость вертикального ветра, м/с;
19) **σ[*w*]** – стандартное отклонение скорости вертикального ветра, м/с;
20) ***W*** – модуль среднего вектора скорости ветра, м/с;
21) ***α*** – угол наклона к горизонту среднего вектора скорости ветра, градусы;
22) ***Vs*** – среднее значение южной компоненты скорости ветра, м/с;
23) ***Ve*** – среднее значение восточной компоненты скорости ветра, м/с;

Б) для интервала между сроками наблюдения

*24) T_{min} – минимальная средняя температура воздуха, ^{0}С;
*25) T_{max} – максимальная средняя температура воздуха, ^{0}С;
26) V_{min} – минимальная скорость мгновенного горизонтального ветра, м/с;
*27) V_{max} – максимальная скорость мгновенного горизонтального ветра, м/с.

Примечания: 1. *Символом (*) здесь отмечены стандартные параметры, требуемые к измерению Роскомгидрометом.*
2. *Под скоростью мгновенного ветра здесь понимается скорость ветра с текущим 2-х секундным усреднением.*
3. *Параметры группы Б выдаются только при стандартном режиме усреднения (в минутах) и при наличии интервала между сроками наблюдения, большем времени усреднения.*

7.2 Список вычисляемых параметров атмосферной турбулентности

Программа **«МЕТЕО 3.0»** вычисляет из первичных данных изделия следующие параметры атмосферной турбулентности:

1) $E_V = (\sigma_u^2 + \sigma_v^2 + \sigma_w^2)/2$ - полная энергия турбулентных движений ($\sigma_u^2, \sigma_v^2, \sigma_w^2$ - дисперсии турбулентных пульсаций трех компонент скорости ветра $u^/$, $v^/$, $w^/$);

2) $I_V = E_V/(V_m)^2$ - относительная интенсивность флуктуаций скорости ветра;

3) $E_t = \sigma_t^2/2$ - энергия температурных флуктуаций (σ_t^2 - дисперсия турбулентных пульсаций температуры $T^/$);

4) $< u^/ \cdot w^/ >$ - момент потока импульса (касательного напряжения) (<> - символ статистического усреднения);

5) $< T^/ \cdot w^/ >$ - момент потока тепла (потока температуры);

6) $\tau = -\rho < u^/ \cdot w^/ >$ - вертикальный поток импульса (ρ - плотность воздуха);

7) $H = c_p \rho < T^/ \cdot w^/ >$ - вертикальный поток тепла (c_p - удельная теплоемкость воздуха при постоянном давлении);

8) $v^* = \sqrt{- < u^/ \cdot w^/ >}$ - скорость трения (масштаб ветра);

9) $T^* = < T^/ \cdot w^/ > / v^*$ - масштаб температуры;

10) $L^* = < T > (v^*)^2 / (\chi \cdot g \cdot T^*)$ - масштаб Монина-Обухова (χ = 0,4 и g = 9,81 м/с^2);

11) $C_d = (v^* / < V >)^2$ - коэффициент сопротивления потоку;

12) $C_T^2 = < [T^/(t+\Delta t) - T^/(t)]^2 > (< V_m > \Delta t)^{-2/3}$ - структурная постоянная температурных флуктуаций (V_m – модуль среднего вектора скорости ветра; Δt - временной интервал между измерениями мгновенных метеовеличин);

13) $C_V^2 = < [u^/(t+\Delta t) - u^/(t)]^2 > (< V_m > \Delta t)^{-2/3}$ - структурная постоянная ветровых флуктуаций;

14) $C_{na}^2 = C_T^2/(2 < T_k >)^2 + C_V^2 / < c >^2$ - структурная постоянная флуктуаций акустического показателя преломления (T_k – температура воздуха в Кельвинах; *c* - скорость звука);

15) $C_{no}^2 = \left\{8 \cdot 10^{-5} < P > / < T_k >^2\right\}^2 C_T^2$ - структурная постоянная флуктуаций оптического показателя преломления (P - атмосферное давление в гПа).

7.3 Выходные данные на экране монитора

7.3.1 Программа **«МЕТЕО 3.0»** в процессе выполнения метеорологических измерений выдает на экран монитора в цифровом и графическом виде мгновенные и текущие средние значения следующих метеорологических величин:

− температуры воздуха **T**, в ^{0}C;
− южной компоненты скорости ветра **Vs**, в м/с (знак плюс соответствует потоку воздуха с юга на север) (только в цифровом виде);

− восточной компоненты скорости ветра **Ve**, в м/с (знак плюс соответствует потоку воздуха с востока на запад) (только в цифровом виде);
− вертикальной компоненты скорости ветра **w**, в м/с (знак плюс соответствует потоку воздуха вверх);
− скорости горизонтального ветра **V**, в м/с;
− направления горизонтального ветра **D**, в градусах или д.у.;
− атмосферного давления **P**, в мм рт.ст. или гПа;
− относительной влажности воздуха **r**, в %;
− абсолютной влажности воздуха **q**, в г/м^3;
− скорости звука в воздухе **c**, в м/с;
а также значения углов наклона ультразвукового термоанемометра от вертикали в двух ортогональных плоскостях **Gx** и **Gy** в угловых градусах – вдоль его осей X и Y соответственно.

7.3.2. По завершению измерений вместо мгновенных величин выводятся их последние вычисленные средние значения (за время усреднения, установленное в диалоговом окне «**Параметры измерений…**», см. рис. 10а). Дополнительно через окно «**Отчет…**» программа может выдать числовые значения других, кроме средних, статистических характеристик температуры и компонент скорости ветра, а также параметры атмосферной турбулентности в наборе, соответствующем подразд. 7.2.

7.3.3 Если отключены датчики давления или влажности, то вместо их показаний на экране отображаются значения, введенные в программу через окно «**Настройки…**» (на ее странице «**Значения**», см. рис. 5).

7.4 Текстовый файл с информационной строкой

7.4.1 Во время метеорологических измерений программа автоматически записывает файлы данного типа, если в ее окне «**Настройки…**» (на странице «**Программа**», см. рис. 4) или в окне «**Сохраняемые данные…**» (рис. 11) в группе кнопок «**Создавать и сохранять файлы**» включена опция «**с краткой записью (*.TXT)** » (см. п. 3.2.3 и п. 4.1). Запись файлов выполняется в папку (директорию), указанную оператором в этих же окнах.

7.4.2 При стандартном режиме усреднения (в минутах) запись этих файлов производится периодически в сроки метеорологических наблюдений, установленные оператором через диалоговое окно «**Сроки регистрации данных…**» (см. разд. 4.1 и рис. 12), а также дополнительно в момент завершения измерений.

При ускоренном режиме усреднения (в секундах) запись этих файлов <u>никогда не производится</u> (см. п. 4.1.6).

7.4.3 Имя файла данного типа формируется самой программой из цифр заводского номера изделия, даты и времени <u>момента записи данных</u> в формате: АМКХХХХХ_ММДДччммсс.TXT, где «АМК» – общий признак для файлов изделий серии АМК-03, «ХХХХХ» – цифры заводского номера изделия, «ММ» – порядковый номер месяца, «ДД» – дата (день месяца), «чч» – час, «мм» – минута, «сс» – секунда, «TXT» – стандартное расширение имени для файлов описываемого типа. Если какое-либо число из перечисленных параметров даты и времени регистрации данных получается меньше 10, то оно дополняется спереди нулем (всегда записывается в виде двух цифр).

7.4.4 В данном файле сохраняется одна строка символов вида

12|04.11.10 12:40|751.23| 24.36| 4.26|176.54| 87.44| 4.57,

где 12 – логический номер сервера (устанавливается на странице «**Значения**» окна «**Настройки…**»), заканчивающийся разделителем «|»;

04.11.10 – дата из 8 символов в формате ДД.ММ.ГГ (здесь и ниже ГГ – две последние цифры года), заканчивающаяся разделителем пробелом;

12:40 – время регистрации сохраняемых данных в формате чч:мм, заканчивающееся разделителем «|».

Далее в строку записываются вычисленные средние значения 6-ти метеорологических величин, разделяемых между собой «|». Для записи каждой величины здесь отводится 6 символьных позиций (используется формат записи вещественных чисел f6.2). При этом они располагаются в строке в следующем порядке: атмосферное давление | температура воздуха | скорость горизонтального ветра | направление ветра | относительная влажность | абсолютная влажность.

7.4.5 Файлы данного типа включены в выходные данные программы **«МЕТЕО 3.0»** для дополнительной возможности считывать и накапливать средние значения метеорологических параметров, измеряемых изделием, в базах данных, разрабатываемых самим пользователем.

7.5 Текстовый файл с массивом строк усредненных данных

7.5.1 Данный файл программа автоматически записывает во время метеорологических измерений, если в ее окне «**Настройки…**» (на странице «**Программа**») или в окне «**Сохраняемые данные…**» в группе кнопок «**Создавать и сохранять файлы**» включена опция «**со строками усредненных данных (*.ггТ)**». Автоматическая запись файла выполняется в папку (директорию), указанную оператором в этих же окнах.

7.5.2 Имя данного файла формируется программой <u>в момент запуска метеорологических измерений</u> из цифр заводского номера изделия, даты и времени <u>начала измерений</u> в формате: АМКХХХХХ_ММДДччммсс.ГГТ, где «АМК» – признак файла изделия серии АМК-03, «ХХХХХ» – цифры заводского номера изделия, «ММ» – порядковый номер месяца, «ДД» – дата (день месяца), «чч» – час, «мм» – минута, «сс» – секунда, «ГГ» – две последние цифры года, Т (или t) – признак файла данного типа. Если какое-либо число из перечисленных параметров даты и времени запуска измерений получается меньше 10, то оно дополняется спереди нулем (всегда записывается в виде двух цифр).

Файл этого типа также может быть записан в любое время по инициативе оператора под любым именем и в выбранную им папку из окна «**Строки данных…**» (рис. 15), если в этом окне нажать кнопку «**Сохранить как…**» (см. п. 4.1.9).

7.5.3 В отличие от файлов всех других типов, описываемых в данном разделе настоящей инструкции, за время одного сеанса измерений (с момента их начала и до момента их завершения) этот файл формируется в единственном числе. Он впервые создается в памяти программы и сохраняется на жестком диске при наступлении первого срока регистрации результатов. При наступлении каждого нового срока регистрации результатов в образ файла, сохраняемого в памяти программы, добавляется снизу новая строка с усредненными (за установленный интервал статистической обработки) значениями метеорологических параметров. После этого новый образ повторно записывается на жесткий диск под тем же именем, замещая ранее записанный файл с этим именем.

При стандартном режиме усреднения (в минутах) дополнение новой строки данных в этом файле производится периодически в сроки метеорологических наблюдений, установленные оператором через диалоговое окно «**Сроки регистрации данных…**», а также в момент завершения измерений.

При ускоренном режиме усреднения (в секундах) дополнение в файл новой строки данных производится через каждый интервал, равный установленному времени усреднения, начиная с момента запуска измерений.

7.5.4 При первой записи файла в него первой записывается строка с параметрами метеостанции (заводской номер, географические координаты, высота над землей, тип подстилающей поверхности), под ней - строка с буквенными обозначениями сохраняемых в файле метеорологических величин, ниже – строка с указаниями физической размерности записываемых величин (см. рис. 15). После этого, четвертой сверху, записывается первая строка измеренных данных, содержащая значения метеорологических параметров для текущего срока их регистрации, разделяемых между собой пробелами.

В каждую строку измеренных данных включаются (в порядке следования)
- порядковый номер строки данных (в колонке N);
- дата (Date) и время (Time) срока, к которому относятся сохраняемые данные;
- фактическое время усреднения данных (<t>);
- средние значения метеорологических величин T, V, V_{min}, V_{max}, D, w, P, r, T_d, q, c;
- среднеквадратические отклонения флуктуаций температуры и компонент ветра RMS(T), RMS(V), RMS(D), RMS(w), RMS(u), RMS(v);
- параметры атмосферной турбулентности Ct2, Cv2, Cn2o, Cn2a, Tau, H, T*, u*, L*.

При записи даты и времени регистрации данных в файле используются форматы представления: для даты - ДД.ММ.ГГ; для времени – чч:мм:сс. В зависимости от используемого программой режима усреднения значение времени усреднения <t> может приводиться в минутах или в секундах. Использованные обозначения метеорологических величин здесь соответствуют п. 7.1, а обозначения для статистических параметров и параметров турбулентности - п. 7.6 (см. рис. 25).

7.6 Текстовый файл статистического отчета

7.6.1 Файл данного типа записывается программой автоматически, если в ее окне «**Настройки…**» (на странице «**Программа**») или в окне «**Сохраняемые данные…**» в группе кнопок «**Создавать и сохранять файлы**» включена опция «**с текстом статистического отчета (*.ггR)**». Автоматическая запись файлов выполняется в папку (директорию), указанную оператором в этих же окнах. Объем первичных данных изделия, учитываемый при подготовке файла отчета, всегда соответствует значению установленного в программе времени усреднения.

7.6.2 При стандартном режиме усреднения (в минутах) автоматическая запись этих файлов производится периодически в сроки метеорологических наблюдений, установленные оператором через диалоговое окно «**Сроки регистрации данных…**», а также дополнительно в момент завершения измерений.

При ускоренном режиме усреднения (в секундах) запись этих файлов производится через каждый интервал, равный установленному времени усреднения, начиная с момента запуска измерений.

7.6.3 Имя файла отчета формируется самой программой из цифр заводского номера изделия, даты и времени <u>момента записи данных</u> в формате: АМКХХХХХ_ММДДЧЧммсс.ГГR, где «АМК» – признак файла изделия серии АМК-03, «ХХХХХ» – цифры заводского номера изделия, «ММ» – порядковый номер месяца, «ДД» – дата (день месяца), «чч» – час, «мм» – минута, «сс» – секунда, «ГГ» – две последние цифры года, R (или r) – признак файла данного типа. Если какое-либо число из перечисленных параметров даты и времени регистрации данных получается меньше 10, то оно дополняется спереди нулем (всегда записывается в виде двух цифр).

Файл этого типа также может быть записан по инициативе оператора под любым именем и в выбранную им папку из окна «**Отчет**» (рис. 16), если в этом окне нажать кнопку «**Сохранить как…**» (см. п. 4.1.11).

7.6.4 Структура содержания файла отчета показана на рис. 25.

В данном файле независимо друг от друга могут исключаться разделы III (со статистическими параметрами) и IV (с параметрами атмосферной турбулентности), если в окне **«Отчет…»** включены соответствующие командные опции (см. п. 4.1.11). Кроме того, в зависимости от используемого программой режима усреднения в разделе I время усреднения может приводиться в минутах или в секундах.

```
********************************
*  УЛЬТРАЗВУКОВАЯ МЕТЕОСТАНЦИЯ  *
*     " АМК-03", рег.№11406     *
*             Отчет             *
********************************

_______________________
I.УСЛОВИЯ ИЗМЕРЕНИЙ:

29.04.11 - дата;
18:52:00 - срок наблюдения;
 0.99897 - время усреднения, минуты;
  80.183 - частота измерений, Гц;
    4804 - количество выполненных циклов измерений;
    4.00 - высота термоанемометра, м;
       1 - трава до 0,2 м (тип подстилающей поверхности).

_______________________________
II.МЕТЕОРОЛОГИЧЕСКИЕ ПАРАМЕТРЫ:

  _________________________
  а) для срока наблюдения

                     _______________________
                     характеристики воздуха
   17.94 <= T - средняя температура, градусы Цельсия;
    0.32 <= RMS(T) - стандартное отклонение температуры, градусы Цельсия;
  750.67 <= P - атмосферное давление, мм.рт.ст.;
   32.06 <= r - относительная влажность воздуха, %;
    6.58 <= e - упругость (давление) водяного пара, гПа;
   13.94 <= Ed - дефицит влажности, гПа;
    1.03 <= Td - температура точки росы, градусы Цельсия;
    5.25 <= q - абсолютная влажность, г/м^3;
    4.10 <= m - массовая концентрация влаги, о/оо;
  1.1929 <= ro - плотность воздуха, кг/м^3;
  342.73 <= c - скорость звука, м/с.

                    _____________________
                    характеристики ветра
    0.44 <= V - средняя скорость горизонтального ветра, м/с;
    0.32 <= Vmin - минимальная скорость мгновенного горизонтального ветра, м/с;
    0.54 <= Vmax - максимальная скорость мгновенного горизонтального ветра, м/с;
    0.09 <= RMS(V) - стандартное отклонение скорости горизонтального ветра, м/с;
  183.59 <= D - среднее направление горизонтального ветра, градусы;
   11.68 <= RMS(D) - стандартное отклонение направления горизонт. ветра, градусы;
   -0.11 <= w - средняя скорость вертикального ветра, м/с;
    0.05 <= RMS(w) - стандартное отклонение скорости вертикального ветра, м/с;
    0.45 <= Vm - модуль среднего вектора скорости ветра, м/с;
  -14.69 <= Av - угол наклона к горизонту среднего вектора скорости ветра, градусы;
    0.43 <= Vs - среднее значение южной компоненты горизонтального ветра, м/с;
   -0.03 <= Ve - среднее значение восточной компоненты горизонтального ветра, м/с.
  ___________________________________________;
  б) для интервала между сроками наблюдения;
   17.57 <= Tmin - минимальная средняя температура воздуха, градусы Цельсия;
   17.85 <= Tmax - максимальная средняя температура воздуха, градусы Цельсия;
    0.29 <= Vmin - минимальная скорость мгновенного горизонтального ветра, м/с;
    0.55 <= Vmax - максимальная скорость мгновенного горизонтального ветра, м/с.

_______________________________
III.СТАТИСТИЧЕСКИЕ ПАРАМЕТРЫ:
```

```
 а) среднеквадратичные отклонения RMS(x)
для T -   0.32 град Ц;    для Vs -   0.10 м/с;    для Ve -   0.08 м/с;
для w -   0.05 м/с;       для u -    0.10 м/с;    для v -    0.08 м/с;
для V -   0.09 м/с;       для D -   11.68 град;   для Vm -   0.09 м/с;

 б) среднеквадратичные разности значений за интервал между циклами
для T -   0.00 град Ц;   для Vm -   0.09 м/с;

 в) коэффициенты асиметрии Ass(x)
для T -  -0.89;          для Vs -   0.13;        для Ve -  -0.48;
для w -  -0.44;          для u -    0.12;        для v -    0.61;
для V -   0.49;          для D -    0.08;        для Vm -   0.71;

 г) коэффициенты эксцесса Ex(x)
для T -  -0.20;          для Vs -   0.35;        для Ve -   5.11;
для w -   1.44;          для u -    0.37;        для v -    5.03;
для V -   0.41;          для D -    4.40;        для Vm -   0.62;

 д) коэффициенты корреляции R(x,y)
между T и w -   0.08;          между T и u -   0.06;
между T и v -  -0.06;          между T и V -  -0.01;
между T и Vm - -0.03;          между w и V -   0.07;
между u и w -   0.07;          между v и w -  -0.10;
между u и v -  -0.15;
где u и v - продольная и поперечная компоненты флуктуаций скорости ветра
(относительно среднего направления D горизонтального ветра).
_____________________________
IV.ПАРАМЕТРЫ ТУРБУЛЕНТНОСТИ:

 9.300E-0003 <= Ev - полная энергия турбулентных движений, (м^2)/(с^2);
 4.589E-0002 <= Iv - относительная интенсивность флуктуаций скорости ветра;
 5.115E-0002 <= Et - энергия температурных флуктуаций, град^2
-4.636E-0004 <= <u*w> - момент потока импульса, (м^2)/(с^2);
 1.611E+0000 <= <w*T> - момент потока тепла, (м^2)/(с^2);
 5.530E-0004 <= Tau - вертикальный поток импульса;
 1.931E+0003 <= H - вертикальный поток тепла;
 2.445E-0003 <= Cd - коэффициент сопротивления потоку;
 7.482E+0001 <= T* - масштаб температуры, град;
 2.153E-0002 <= u* - масштаб скорости ветра, м/с;
 4.596E-0004 <= L* - масштаб Монина-Обухова, м;
 5.366E-0005 <= Ct2 - структ. постоянная флукт. температуры, град^2/(см^2/3);
 1.079E-0004 <= Cv2 - структ. постоянная флуктуаций ветра, (м/с)^2/(см^2/3);
 2.161E-0008 <= Cn2a - структ. пост.для акустич. показат. преломл., 1/(м^2/3);
 4.791E-0017 <= Cn2o - структ. пост. для оптич. показат. преломл., 1/(см^2/3).
```

Рис. 25

7.7 Текстовый файл с массивом мгновенных данных

7.7.1 Файл данного типа записывается программой автоматически, если в ее окне «**Настройки…**» (на странице «**Программа**») или в окне «**Сохраняемые данные…**» в группе кнопок «**Создавать и сохранять файлы**» включена опция «**с данными в текстовом формате (*.ггD)**»). Автоматическая запись файлов выполняется в папку (директорию), указанную оператором в этих же окнах. Объем первичных данных изделия, учитываемый при подготовке файла, всегда соответствует значению установленного в программе времени усреднения.

7.7.2 При стандартном режиме усреднения (в минутах) автоматическая запись этих файлов производится периодически в сроки метеорологических наблюдений, установленные оператором через диалоговое окно «**Сроки регистрации данных…**», а также дополнительно в момент завершения измерений.

При ускоренном режиме усреднения (в секундах) запись этих файлов производится через каждый интервал, равный установленному времени усреднения, начиная с момента запуска измерений.

7.7.3 Имя файла с мгновенными данными формируется самой программой из цифр заводского номера изделия, даты и времени момента записи данных в формате: АМКХХХХХ_ММДДччммсс.ГГD, где «АМК» – признак файла изделия серии АМК-03, «ХХХХХ» – цифры заводского номера изделия, «ММ» – порядковый номер месяца, «ДД» – дата (день месяца), «чч» – час, «мм» – минута, «сс» – секунда, «ГГ» – две последние цифры года, D (или d) – признак файла данного типа. Если какое-либо число из перечисленных параметров даты и времени регистрации данных получается меньше 10, то оно дополняется спереди нулем (всегда записывается в виде двух цифр).

7.7.4 Содержание начальной части этих файлов повторяет содержание файлов отчета (см. п.7.6.4 и рис.25), в которых исключены разделы III (со статистическими параметрами) и IV (с параметрами атмосферной турбулентности). Единственное здесь отличие только в том, что в шапку заголовка файла с мгновенными данными вместо слова «Отчет» записываются слова «Данные измерений». Далее в этот файл включаются новые разделы (III и IV, см. рис. 26). В раздел III в колонки записываются измеренные за время усреднения мгновенные значения метеорологических величин: T – температуры, Vs, Ve и w – южной, восточной и вертикальной ортогональных компонент вектора скорости ветра, P – давления и r – относительной влажности. В разделе IV приводятся пояснения возможных значений err - признака качества данных для текущих мгновенных измерений.

7.7.5 Текстовые файлы с массивом мгновенных данных (типа *.ГГD) могут загружаться программой «**METEO DP**» для последующего анализа из их данных атмосферной турбулентности.

```
********************************
*  УЛЬТРАЗВУКОВАЯ МЕТЕОСТАНЦИЯ  *
*     " АМК-03", рег.№11406     *
*        Данные измерений       *
********************************
_______________________
I.УСЛОВИЯ ИЗМЕРЕНИЙ:

29.04.11 - дата;
18:52:00 - срок наблюдения;
 0.99897 - время усреднения, минуты;
  80.183 - частота измерений, Гц;
    4804 - количество выполненных циклов измерений;
    4.00 - высота термоанемометра, м;
       1 - трава до 0,2 м (тип подстилающей поверхности).
_______________________________
II.МЕТЕОРОЛОГИЧЕСКИЕ ПАРАМЕТРЫ:
  _______________________
  а) для срока наблюдения
                    _______________________
                    характеристики воздуха
   17.94 <= T - средняя температура, градусы Цельсия;
    0.32 <= RMS(T) - стандартное отклонение температуры, градусы Цельсия;
  750.67 <= P - атмосферное давление, мм.рт.ст.;
   32.06 <= r - относительная влажность воздуха, %;
    6.58 <= e - упругость (давление) водяного пара, гПа;
   13.94 <= Ed - дефицит влажности, гПа;
    1.03 <= Td - температура точки росы, градусы Цельсия;
    5.25 <= q - абсолютная влажность, г/м^3;
```

```
    4.10 <= m - массовая концентрация влаги, о/оо;
  1.1929 <= ro - плотность воздуха, кг/м^3;
  342.73 <= c - скорость звука, м/с.

                    ____________________
                    характеристики ветра
    0.44 <= V - средняя скорость горизонтального ветра, м/с;
    0.32 <= Vmin - минимальная скорость мгновенного горизонтального ветра, м/с;
    0.54 <= Vmax - максимальная скорость мгновенного горизонтального ветра, м/с;
    0.09 <= RMS(V) - стандартное отклонение скорости горизонтального ветра, м/с;
  183.59 <= D - среднее направление горизонтального ветра, градусы;
   11.68 <= RMS(D) - стандартное отклонение направления горизонт. ветра, градусы;
   -0.11 <= w - средняя скорость вертикального ветра, м/с;
    0.05 <= RMS(w) - стандартное отклонение скорости вертикального ветра, м/с;
    0.45 <= Vm - модуль среднего вектора скорости ветра, м/с;
  -14.69 <= Av - угол наклона к горизонту среднего вектора скорости ветра, градусы;
    0.43 <= Vs - среднее значение южной компоненты горизонтального ветра, м/с;
   -0.03 <= Ve - среднее значение восточной компоненты горизонтального ветра, м/с.
  ____________________________________________;
  б) для интервала между сроками наблюдения;
   17.57 <= Tmin - минимальная средняя температура воздуха, градусы Цельсия;
   17.85 <= Tmax - максимальная средняя температура воздуха, градусы Цельсия;
    0.29 <= Vmin - минимальная скорость мгновенного горизонтального ветра, м/с;
    0.55 <= Vmax - максимальная скорость мгновенного горизонтального ветра, м/с.

_____________________________
III.МГНОВЕННЫЕ ЗНАЧЕНИЯ:
     N    T      Vs     Ve     w      P      r    err
     1  17.03   0.40   2.03  -0.85  750.5   32.1   0
     2  17.10   0.30   2.02  -0.85  750.5   32.1   0
     3  17.08   0.27   2.08  -0.86  750.5   32.0   0
     4  17.92   0.32   2.10  -1.03  750.5   32.1   0
.................................................
  4802  17.08  -0.34   0.92   0.48  750.4   32.4   0
  4803  17.07  -0.44   0.93   0.39  750.4   32.4   0
  4804  17.11  -0.38   0.93   0.31  750.4   32.4   0

_____________________________
IV.ПРИЗНАКИ КАЧЕСТВА ДАННЫХ:
err = 0 - идеальные данные;
err = 1 - использовалась входная фильтрация первичных данных устройства;
err = 2 - использовалась медианная фильтрация выходных данных программы;
err = 3 - не совпала контрольная сумма для переданных с устройства байтов
          (т.е.существует вероятность сбоя при передаче данных);
err = 4 - первичные данные термоанемометра были вне диапазона разрешенных значений;
err = 5 - не поступили данные датчика атмосферного давления;
err = 6 - не поступили данные датчика влажности воздуха;
err = 7 - не поступили данные датчика угла наклона метеомачты вдоль оси X;
err = 8 - не поступили данные датчика угла наклона метеомачты вдоль оси Y;
ПРИ СОБЫТИЯХ err=4,5,6,7 и 8 В НАСТОЯЩИЙ ФАЙЛ ВМЕСТО ОТБРАКОВАННЫХ ДАННЫХ
ЗАПИСЫВАЮТСЯ ИХ ПОСЛЕДНИЕ ДОСТОВЕРНЫЕ ЗНАЧЕНИЯ.
```

Рис. 26

7.8 Бинарный файл с массивом мгновенных данных

7.8.1 Данный файл записывается программой автоматически только при стандартном режиме усреднения (в минутах), если в ее окне «**Настройки…**» (на странице «**Программа**») или в окне «**Сохраняемые данные…**» в группе кнопок «**Создавать и сохранять файлы**» включена опция «**с данными в бинарном формате (*.ггВ)**» (см. п.п. 3.2.3 и 4.1.6).

7.8.2 Бинарный файл создается в момент запуска измерений. Далее через каждые 30 секунд этот файл дополняется данными из оперативной памяти ПЭВМ, измеренными изделием за этот

промежуток времени (при этом одновременно уже сохраненные в файле данные удаляются из оперативной памяти). Файл дополняется примерно по 3900 байт при частоте измерений изделия, равной 10 Гц, и пропорционально большим числом байтов при использовании в изделии большей частоты измерений. Как только время записи файла, записываемого в текущий момент времени, превысит установленное в программе время усреднения <t> (от 1 до 20 мин), создается новый бинарный файл для последующей записи данных. При досрочном прерывании измерений пользователем все данные в оперативной памяти, которые не были еще записаны, также сохраняются в файле.

Размер каждого из бинарных файлов (объем данных) при стандартной длительности их записи, равной установленному времени усреднения <t>, составляет 13*f*<t>+31 байт, где f – частота измерений в Гц, а <t> в секундах, т.е. он может достигать 1,25 Мбайт (максимум при f =80 Гц и <t> = 1200 с).

7.8.3 Запись бинарных файлов выполняется только в папку, указанную в строке ввода «**Папка с выходными данными**» на странице «**Значения**» окна «**Настройки…**» (см. п. 3.2.4).

7.8.4 Имя каждого бинарного файла формируется программой из цифр заводского номера изделия, даты и времени регистрации первого отсчета данных изделия, записываемого в файл, в формате: АМКХХХХХ_ММДДччммсс.ГГВ, где «АМК» – признак файла изделия серии АМК-03, «ХХХХХ» – цифры заводского номера изделия, «ММ» – порядковый номер месяца, «ДД» – дата (день месяца), «чч» – час, «мм» – минута, «сс» – секунда, «ГГ» – две последние цифры года, В (или b) – признак файла данного типа. Если какое-либо число из перечисленных параметров даты и времени получается меньше 10, то оно дополняется спереди нулем (всегда записывается в виде двух цифр).

7.8.5 Структура содержания бинарного файла.

В начале файла записываются байты, несущие информацию о дате и времени регистрации первого отсчета данных изделия, записываемого в файл, (ниже они описываются парами: наименование числового параметра – размер его записи в байтах):

год – 2 байта;
месяц – 2 байта;
день – 2 байта;
час – 2 байта;
минута – 2 байта;
секунда – 2 байта;
миллисекунда – 2 байта.
Далее – высота измерений, умноженная на 100 и округленная до целого – 2 байта;
тип подстилающей поверхности – 1 байт.

После этого записываются мгновенные значения метеорологических параметров для каждого цикла измерений:

T, умноженное на 100 и округленное до целого – 2 байта;
Vs, умноженное на 100 и округленное до целого – 2 байта;
Ve, умноженное на 100 и округленное до целого – 2 байта;
w, умноженное на 100 и округленное до целого – 2 байта;
P, умноженное на 10 и округленное до целого – 2 байта;
r, умноженное на 100 и округленное до целого – 2 байта;
признак ошибки err – 1 байт.

По завершению записи каждого бинарного файла в его конец дописываются дата и время регистрации последнего отсчета данных изделия, записываемого в файл:

год – 2 байта;
месяц – 2 байта;

день – 2 байта;
час – 2 байта;
минута – 2 байта;
секунда – 2 байта;
миллисекунда – 2 байта.

Двухбайтовые параметры записываются младшим байтом вперед, т.е. сначала младший байт числа, затем старший байт.

7.8.6 Бинарные файлы с массивом мгновенных данных (типа *.ГГВ) могут загружаться программой «**METEO DP**» для последующего анализа из их данных атмосферной турбулентности.

7.9 Запись в базу данных

7.9.1 Программа **«METEO 3.0»** выполняет автоматические записи результатов измерений в **базе данных**, если (см. также подразд. 4.1.4):

- на ПЭВМ установлена поддерживаемая программой **база данных**;
- на ПЭВМ в одной папке с программой **«METEO 3.0»** дополнительно установлена программа **«METEO DB»**;
- при запуске измерений в диалоговом окне **«Параметры измерений…»** (см. рис. 10а) была включена командная опция **«Записывать результаты в базу данных»**.

7.9.2 Запись новых данных в **базу данных** производится только при стандартном режиме усреднения (в минутах) периодически в сроки, установленные оператором через диалоговое окно «**Сроки регистрации данных…**», а также дополнительно в момент завершения измерений.

В каждой новой записи в **базу данных** объем передаваемой в нее информации идентичен объему данных, содержащихся в файле отчета.

7.10 Динамический обмен данными с другими Windows-приложениями

7.10.1 В программе **«METEO 3.0»** реализована возможность динамически связываться с ней другим приложениям Windows на основе поддерживаемой в этой операционной системе технологии “**Dynamic Data Exchange**” (сокращенно - **DDE**). При этом, если на странице “**Программа**” окна “**Настройки…**” включена опция “**Передавать другим программам через DDE**» (см. п. 3.2.3), то **«METEO 3.0»** передает свои выходные данные другим приложениям Windows, установившим с ней DDE-связь, непосредственно через оперативную память ПЭВМ.

При использовании DDE-связи приложение **«METEO 3.0»** является DDE-сервером (поставщиком данных), а приложения, которые связываются с ним, – DDE-клиентом (потребителем данных). Ниже описывается правила установления DDE-связи с приложением **«METEO 3.0»** и порядок обмена с ним данными. Данная информация предназначена для программистов, разрабатывающих собственные приложения Windows.

7.10.2 При реализации информационного сопряжения по DDE-технологии DDE-клиент должен использовать три символьных идентификатора, установленные изготовителем DDE-сервера:

а) ***M1*** - идентификатор имени DDE-сервера, который совпадает с именем исполняемого exe-файла приложения без расширения (если DDE-сервер и DDE-клиент установлены на компьютере в разных директориях, то имя головного файла следует использовать с добавлением полного пути к нему);

б) ***M2*** - идентификатор темы DDE-связи между программами;

в) ***M3*** - идентификатор источника данных на DDE-сервере (по выбранной теме).

Возможные значения идентификаторов.

Имя DDE-сервера ***M1*** (исполняемого файла приложения **«МЕТЕО 3.0»**) формируется изготовителем как AMK_XXXXX.EXE, где «XXXXX» - символы цифр заводского номера изделия.

Идентификатор темы DDE-связи между программами ***M2*** может иметь следующие значения:

− AMKReportDdeConv – идентификатор темы DDE-связи, при которой передаются данные, описанные в подразд. 7.6 (в виде набора символьных строк);

− AMKDataDdeConv – идентификатор темы DDE-связи, при которой передаются данные, описанные в подразд. 7.7 (в виде набора символьных строк);

− AMKDataStrDdeConv – идентификатор темы DDE-связи, при которой передаются мгновенные значения метеорологических параметров в виде одной строки, описанной ниже в п. 7.10.5.

Идентификатор источника данных на DDE-сервере ***M3*** может иметь следующие значения:

− AMKReportDdeItem – идентификатор источника данных при установлении DDE-связи по теме AMKReportDdeConv;

− AMKDataDdeItem – идентификатор источника данных при установлении DDE-связи по теме AMKDataDdeConv;

− AMKDataStrDdeItem – идентификатор источника данных при установлении DDE-связи по теме AMKDataStrDdeConv.

7.10.3 Порядок загрузки DDE-сервера и установки с ним связи.

DDE-клиент должен выполнять автоматическую загрузку DDE-сервера и устанавливать с ним DDE-связь, используя пару идентификаторов ***M1*** и ***M2***.

Пример программного кода на DELPHI (см. также п. 7.10.2):

```
DdeClientConv1.SetLink('AMK_XXX.EXE',' AMKReportDdeItem');
If not DdeClientConv1.OpenLink then ShowMessage('Нет связи с сервером');
```

При завершении работы DDE-клиент должен обязательно закрывать установленную им DDE-связь (оператором DdeClientConv1.CloseLink).

При автоматической загрузке DDE-сервера перед закрытием DDE-связи рекомендуется дополнительно выдать DDE-серверу макрокоманду автоматического завершения работы приложения «**МЕТЕО 3.0**» (см. п. 7.10.6).

7.10.4 Порядок запроса и получения данных DDE-сервера

DDE-клиент получает данные DDE-сервера только по запросу. Запрос данных должен выполняться DDE-клиентом, используя имя идентификатора источника данных на DDE-сервере (***M3***), соответствующее теме ранее установленной DDE-связи.

Пример программного кода на DELPHI:

а) для случая запроса по теме AMKDataDdeConv (или AMKReportDdeConv)

```
Data1.SetText(DdeClientConv1.RequestData(AMKDataDdeItem))
(или Data1.SetText(DdeClientConv1.RequestData(AMKReportDdeItem)));
(здесь Data1 – объект типа TStringList, в который загружается массив строк);
```

б) для случая запроса по теме AMKDataStrDdeConv

```
Info:=DdeClientConv1.RequestData('AMKDataStrDdeItem');
(здесь DDE-клиент получает от DDE-сервера строку Info типа PChar).
```

Если DDE-клиент выполняет запрос данных DDE-сервера по теме AMKReportDdeConv через идентификатор AMKReportDdeItem, то в ответ он получает от DDE-сервера статистический отчет о результатах измерения метеорологических величин. Здесь данные с DDE-сервера передаются

как набор символьных строк в виде, полностью идентичном по структуре и содержанию файлу отчета, описанному в подразд. 7.6.

Если DDE-клиент выполняет запрос данных DDE-сервера по теме AMKDataDdeConv через идентификатор AMKDataDdeItem, то в ответ он получает от DDE-сервера мгновенные значения метеорологических величин, накопленные DDE-сервером в оперативной памяти ПЭВМ за время измерений. Данные с DDE-сервера в этом случае передаются как набор символьных строк в виде, полностью идентичном по структуре и содержанию файлу, описанному в подразд. 7.7.

При выполнении метеорологических измерений данные, передаваемые DDE-сервером по темам AMKDataDdeConv и AMKReportDdeConv, автоматически обновляются им через каждую минуту с момента начала измерений, а также после их завершения. При этом DDE-сервер не выдает никаких данных до окончания первой минуты первого сеанса метеорологических измерений после загрузки приложения. После завершения очередного сеанса измерений DDE-сервер сохраняет накопленные им данные до начала следующего сеанса измерений.

Если DDE-клиент выполняет запрос данных DDE-сервера по теме AMKDataStrDdeConv через идентификатор AMKDataStrDdeItem, то в ответ он получает строку с мгновенными значениями метеорологических параметров (см. п. 7.10.5). В этом случае передаваемые данные обновляются DDE-сервером также часто, как производятся измерения изделием (например, 80 раз за секунду при частоте измерений, равной 80 Гц). В данном случае рекомендуется устанавливать на DDE-клиенте частоту запросов новых данных, равной или больше частоты обновления данных DDE-сервера; в противном случае часть данных DDE-клиентом будет утеряна.

7.10.5 Формат строки с мгновенными значениями метеорологических параметров

Данная строка записывается в виде последовательности символов вида:

09.03.04 12:40:45:354|-1836| 1426|27650| -14|74930| 8640|0

Здесь 09.03.04 – дата измерений передаваемых данных, записанная в 8 символах в формате ДД.ММ.ГГ (ДД – день месяца, ММ – месяц, ГГ – две последние цифры года), заканчивающихся разделителем в виде пробела;

12:40:45:354 – время измерений передаваемых данных, записанное в 12 символах в формате чч:мм:ss:nnn (чч – час, мм – минута, ss – секунда; nnn – три символа для записи миллисекунды), заканчивающихся разделителем «|».

Далее идут мгновенные значения 6-ти метеорологических величин, разделяемые «|», для записи каждого из которых отводится 5 символьных позиций. Метеорологические величины располагаются в порядке: температура воздуха | скорость горизонтального ветра | направление горизонтального ветра | скорость вертикального ветра (знак плюс соответствует потоку воздуха вверх, минус – вниз)| атмосферное давление | относительная влажность воздуха. Значение каждой метеорологической величины умножается на 100 и округляется до целого числа. Если для записи значения какой-либо метеорологической величины требуется меньше 5 символов, то она дополняется спереди необходимым количеством пробелов.

Строка завершается одним символом, указывающим достоверность измерения передаваемых в ней данных. При выполнении измерений мгновенных значений метеорологических величин приложение **«МЕТЕО 3.0»** всегда выполняет контроль их достоверности, чтобы исключить при их математической обработке последствия возможных случайных сбоев в работе устройства. Если в описываемой строке последний символ соответствует 0 (нулю), то передаваемые в ней данные достоверны. Если же этот символ соответствует любому числу, не равному нулю, то для указанного в строке момента времени из устройства были получены недостоверные результаты измерений.

В последнем случае значения метеорологических величин не обновляются в строке, т.е. соответствуют их значениям, переданным в предыдущий момент времени.

7.10.6 Возможности управления DDE-сервером со стороны DDE-клиента

DDE-сервер может отрабатывать 2 макрокоманды от DDE-клиента, описываемые ниже. Каждая макрокоманда представляет собой строку макроса (параметр Cmd типа PChar), последовательность символов в которой определяет передаваемые в DDE-сервер тип и параметры команды.

Пример программного кода на DELPHI:

```
StrPCopy(Cmd,'строка макрокоманды');

DdeClientConv1.ExecuteMacro(Cmd,false);
```

(Здесь Cmd:array[0..n-1] of Char, где n – число используемых символов в строке макрокоманды).

Макрокоманда автоматического завершения работы DDE-сервера выполняется в виде строки символов

Close_METEO_3_0

Данная макрокоманда завершает работу приложения **«METEO 3.0»**. Она исполняется DDE-сервером через 5 сек после ее получения от DDE-клиента. За это время DDE-клиент должен обязательно закрыть DDE-связь.

DDE-сервер также поддерживает возможность автоматической передачи ему от DDE-клиента сообщения о значениях четырех параметров, указанных ниже. При этом прием данного сообщения для DDE-сервера одновременно является командой DDE-клиента на организацию им нового сеанса метеорологических измерений.

Если в момент получения данного сообщения-команды DDE-сервер не выполняет никаких измерений (чтения данных устройства), то он считывает значения этих параметров и автоматически исполняет свою команду меню «**Работа/Измерения…F9**». Если же при получении данного сообщения-команды DDE-сервер выполняет какие-либо измерения, включая градуировку термоанемометра или контрольный опрос первичных данных изделия, то он игнорирует поступившее сообщение-команду.

Указанное сообщение-команду DDE-клиент должен выполнять в виде строки символов *TTFFFFFZZZZZP*.

Здесь *TT* - два символа, используемые для передачи (задания) времени усреднения <*t*> в очередном сеансе метеорологических измерений в пределах от 1 до 20 мин с дискретностью в 1 мин (целое число). Если <*t*> меньше 10, то первый символ должен быть равен '0'.

FFFFF – пять символов, используемые для передачи вещественного значения угла *f* азимутальной ориентации термоанемометра относительно севера в пределах 0 <= *f* < 360 град. Предварительно значение *f* должно быть умножено на 100 и округлено до целого числа. Если получающееся целое число содержит менее 5-и символов, то субстрока *FFFFF* дополняется спереди символами '0' (нуля).

ZZZZZ - пять символов, используемые для передачи вещественного значения высоты *z* измерений над поверхностью земли в пределах 0 < *z* <= 500 м. Значение *z* кодируется в *ZZZZZ* аналогично как *f* в *FFFFF*.

P – символ для передачи номера типа подстилающей поверхности ($0 \le P \le 9$).

Последние три сообщения (*FFFFF*, *ZZZZZ* и *P*) не являются обязательными, т.е. разрешается передавать описанное сообщение-команду как *TTFFFFFZZZZZ* или *TTFFFFF* или *TT*.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводит. докум. и дата	Подпись	Дата
	Изменен-ных	Заменен-ных	Новых	Изъятых					

Подп. и дата	Инв. № дубл.	Взам. инв.№	Подп. и дата	Инв. № подл.

ПРИЛОЖЕНИЕ X

Текст программы "МЕТЕО-3.0"

ООО «СИБАНАЛИТПРИБОР»

УТВЕРЖДАЮ:

Директор ООО «Сибаналитприбор»

_____ А.А.Азбукин

«16» ____11____ 2015 г.

Программное обеспечение для ПЭВМ ультразвуковой метеостанции

АМК-03

Программа «**МЕТЕО 3.0**»

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

АМЯ 4.079.001 12

Листов 242

Исполнитель, сн.с., к.ф.-м.н.,

_____ А.Я. Богушевич

«16» ____ноября____ 2015 г.

2015

Инв. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

УТВЕРЖДЕН

АМЯ4.079.001 12-ЛУ

Программное обеспечение для ПЭВМ ультразвуковой метеостанции

АМК-03

Программа «**МЕТЕО 3.0**»

Текст программы

АМЯ 4.079.001 12

2015

Подп. и дата	
Инв. № дубл.	
Взам. инв.№	
Подп. и дата	
Инв. № подл.	

АННОТАЦИЯ

В настоящем документе приводится текст программы «МЕТЕО 3.0», являющейся стандартным (базовым) программным обеспечением (ПО) для ультразвуковых метеостанций типа АМК-03, устанавливаемым на ПЭВМ, использующих операционную систему Windows (любой 32-х или 64-х разрядной версии).

При разработке данного программного обеспечения применялась система объектно-ориентированного программирования «Delphi 6.0» фирмы «Borland Enterprise»; используемый алгоритмический язык программирования – «Object Pascal».

```
PROGRAM AMK_15401;
uses
  Forms,
  Main in 'Main.pas' {MainForm},
  UnAMKini in 'UnAMKini.pas',
  ApllDlgs in 'ApllDlgs.pas',
  UnAMKsys in 'UnAMKsys.pas',
  UnAMKport in 'UnAMKport.pas',
  UnData in 'UnData.pas',
  UnMeteo in 'UnMeteo.pas',
  UnDevice in 'UnDevice.pas',
  GraphLib in 'GraphLib.pas',
  Izmeren in 'Izmeren.pas' {IzmerenForm},
  Calibrat in 'Calibrat.pas' {CalibUGIForm},
  Control in 'Control.pas' {ControlUGIForm},
  Report in 'Report.pas' {ReportForm},
  UnMinFrm in 'UnMinFrm.pas' {MinForm},
  PPU25 in 'PPU25.pas' {PPU25Dlg},
  Options in 'Options.pas' {OptionsDlg},
  RunIzmer in 'RunIzmer.pas' {RunIzmerenDlg},
  AutSave in 'AutSave.pas' {AutSaveDlg},
  RunCalib in 'RunCalib.pas' {RunCalibratingDlg},
  TestPPU in 'TestPPU.pas' {TestPPUdlg},
  StyleGr in 'StyleGr.pas' {StyleGraphDlg},
  Phone in 'Phone.pas' {PhoneForm},
  SetPhone in 'SetPhone.pas' {OptionPhoneDlg},
  PrnText in 'PrnText.pas' {PrintTextDlg},
  Splash in 'Splash.pas' {SplashForm},
  About in 'About.pas' {AboutBox},
  AboutDevice in 'AboutDevice.pas' {DeviceBox};

{$R *.RES}
BEGIN
  Application.Initialize;
  if GetDataAMKSys then begin
  SplashForm:=TSplashForm.Create(Application);
  SplashForm.Show;
  SplashForm.Update;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TIzmerenForm, IzmerenForm);
  Application.CreateForm(TCalibUGIForm, CalibUGIForm);
  Application.CreateForm(TControlUGIForm, ControlUGIForm);
  Application.CreateForm(TMinForm, MinForm);
  Application.CreateForm(TReportForm, ReportForm);
  SplashForm.Hide;
  SplashForm.Free;
  Application.Run;
  UpDateAMKsys;
  end;
  SysData.Free;
END.

{888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
UNIT Main;
interface
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, Buttons, Menus, ToolWin, ImgList, ActnList,
  ShellApi, DdeMan, DB, DBTables, Grids, DBGrids, BDE, Registry, UnData;

type
  TMainForm = class(TForm)
    Timer1: TTimer;
    MainMenu1: TMainMenu;
    WorkMenu: TMenuItem;
    HelpMenu: TMenuItem;
    SpedPanel: TToolBar;
    Sepan2: TPanel;
    Sepan3: TPanel;
    IzmerenBtn: TToolButton;
    CalibrateBtn: TToolButton;
    ControlBtn: TToolButton;
    ReportBtn: TToolButton;
    OptionBtn: TToolButton;
    BazaDataBtn: TToolButton;
    HelpContentBtn: TToolButton;
    Sep7Btn: TToolButton;
    ExitBtn: TToolButton;
    Sepan7: TPanel;
    Sepan4: TPanel;
    ActionList1: TActionList;
    ImageList1: TImageList;
    IzmerenItem: TMenuItem;
    CalibrateItem: TMenuItem;
    ControlItem: TMenuItem;
    ReportItem: TMenuItem;
    N5: TMenuItem;
    OptionItem: TMenuItem;
    N7: TMenuItem;
    ExitProgrItem: TMenuItem;
    BazaDataItem: TMenuItem;
    N1: TMenuItem;
    HelpContentItem: TMenuItem;
    AboutPrItem: TMenuItem;
    ToolButton1: TToolButton;
    SetupConnectItem: TMenuItem;
    N3: TMenuItem;
    SetPhoneBtn: TToolButton;
    Panel3: TPanel;
    PanelInform: TPanel;
    EditTr: TEdit;
    StatusBar0: TStatusBar;
    EditTime: TEdit;
    EditUsredn: TEdit;
    EditCount: TEdit;
    EditDevice: TEdit;
    Bevel2: TBevel;
    Bevel3: TBevel;
    Bevel4: TBevel;
    Bevel5: TBevel;
    Bevel6: TBevel;
    Bevel1: TBevel;
    EditGamX: TEdit;
    LabGx: TLabel;
```

```
EditGamY: TEdit;
LabGy: TLabel;
LabGxdop: TLabel;
LabGydop: TLabel;
BevRight: TBevel;
EditOrient: TEdit;
AMKReportDdeConv: TDdeServerConv;
AMKReportDdeItem: TDdeServerItem;
AMKDataDdeConv: TDdeServerConv;
AMKDataDdeItem: TDdeServerItem;
ServisMenu: TMenuItem;
GraphicsItem: TMenuItem;
ToolButton3: TToolButton;
GraphicsBtn: TToolButton;
WindowMenu: TMenuItem;
ViewPhoneItem: TMenuItem;
ViewIzmerItem: TMenuItem;
ViewGraduirItem: TMenuItem;
ViewControlItem: TMenuItem;
ViewMainItem: TMenuItem;
SaveDialog1: TSaveDialog;
PanelWork: TPanel;
RegimVabor: TRadioGroup;
StartBtn: TBitBtn;
StopBtn: TBitBtn;
ViewBtn: TBitBtn;
TitleWorkPanel: TPanel;
DopOptionBtn: TBitBtn;
BitBtn1: TBitBtn;
RegOutData: TRadioGroup;
PanelParam: TPanel;
TitleDataPanel: TPanel;
PanelSredn: TPanel;
Panel1a: TPanel;
Label1a: TLabel;
Label1b: TLabel;
Panel1b: TPanel;
PaintBox1: TPaintBox;
Edit1: TEdit;
Panel2a: TPanel;
Label2a: TLabel;
Label2b: TLabel;
Label2c: TLabel;
Panel2b: TPanel;
PaintBox2: TPaintBox;
Edit2: TEdit;
Panel3a: TPanel;
Label3a: TLabel;
Label3b: TLabel;
Label3c: TLabel;
Panel3b: TPanel;
PaintBox3: TPaintBox;
Edit3: TEdit;
Panel4a: TPanel;
Label4a: TLabel;
Label4b: TLabel;
Panel4b: TPanel;
PaintBox4: TPaintBox;
```

```
Edit4: TEdit;
Panel5a: TPanel;
Label5a: TLabel;
Label5b: TLabel;
Panel5b: TPanel;
PaintBox5: TPaintBox;
Edit5: TEdit;
Panel6a: TPanel;
Label6a: TLabel;
Label6b: TLabel;
Panel6b: TPanel;
PaintBox6: TPaintBox;
Edit6: TEdit;
Lab_UGI1: TLabel;
Lab_UGI2: TLabel;
LabelAver1: TLabel;
Label14: TLabel;
LabelTime: TLabel;
LabGammas: TLabel;
LabHor1: TLabel;
LabHor2: TLabel;
Izmeren: TAction;
Calibrate: TAction;
Control: TAction;
Options: TAction;
SetupConnect: TAction;
ExitProgr: TAction;
Report: TAction;
BazaData: TAction;
AboutPr: TAction;
HelpContent: TAction;
VidGraphics: TAction;
ViewMinWinItem: TMenuItem;
MinFormBtn: TToolButton;
MinimumBtn: TBitBtn;
ToolButton4: TToolButton;
AddFormBtn: TToolButton;
ToolButton5: TToolButton;
N4: TMenuItem;
DataSource1: TDataSource;
DBGrid1: TDBGrid;
DataTable: TTable;
DataTableNum: TAutoIncField;
DataTableDateRec: TDateField;
DataTableTimeRec: TTimeField;
DataTableTimeAver: TFloatField;
DataTableTemperat: TFloatField;
DataTableHorWind: TFloatField;
DataTableDirWind: TFloatField;
DataTablePressureRec: TFloatField;
DataTableRelHuRec: TFloatField;
DataTableVertWind: TFloatField;
DataTableT_Ct2: TFloatField;
DataTableT_Cv2: TFloatField;
DataTableSko_T: TFloatField;
DataTableSko_V: TFloatField;
DataTableSko_F: TFloatField;
DataTableSko_w: TFloatField;
```

```
DataTableSko_ux: TFloatField;
DataTableSko_vy: TFloatField;
DataTableR_Tw: TFloatField;
DataTableR_TV: TFloatField;
DataTableR_Tux: TFloatField;
DataTableR_Tvy: TFloatField;
DataTableR_Vw: TFloatField;
DataTableR_uxw: TFloatField;
DataTableR_vyw: TFloatField;
DataTableR_uxvy: TFloatField;
DataTableAss_T: TFloatField;
DataTableAss_V: TFloatField;
DataTableAss_F: TFloatField;
DataTableAss_w: TFloatField;
DataTableAss_ux: TFloatField;
DataTableAss_vy: TFloatField;
DataTableExc_T: TFloatField;
DataTableExc_V: TFloatField;
DataTableExc_F: TFloatField;
DataTableExc_w: TFloatField;
DataTableExc_ux: TFloatField;
DataTableExc_vy: TFloatField;
Label15: TLabel;
LabelAver2: TLabel;
GetConstants: TAction;
ConsolesItem: TMenuItem;
ToolButton2: TToolButton;
AnalysData: TAction;
AnalysDataItem: TMenuItem;
ToolButton6: TToolButton;
DataTableVmin: TFloatField;
DataTableVmax: TFloatField;
AMKDataStrDdeConv: TDdeServerConv;
AMKDataStrDdeItem: TDdeServerItem;
LabelTemper: TLabel;
LabTr: TLabel;
LabTrdop: TLabel;
GetTestPPU: TAction;
N2: TMenuItem;
GetTestPPUItem: TMenuItem;
procedure ShowHint(Sender: TObject);
procedure ParadoxPassword(Sender: TObject; var Continue: Boolean);
procedure UpdateStartControl(Sender:TObject);
procedure MakeCikleControl(Sender:TObject);
procedure UpdateStopControl(Sender: TObject);
procedure UpdateStartCalibr(Sender:TObject);
procedure MakeCikleCalibr(Sender: TObject);
procedure UpdateStopCalibr(Sender: TObject);
procedure UpdateStartIzmeren(Sender:TObject);
procedure MakeCikleIzmeren(Sender: TObject);
procedure UpdateStopIzmeren(Sender: TObject);
procedure ControlAutomatic(Sender: TObject);
procedure RestoreModems(Sender: TObject);
procedure UpdateAllGraph(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure StartBtnClick(Sender: TObject);
procedure StopBtnClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ExitProgrExecute(Sender: TObject);
procedure IzmerenExecute(Sender: TObject);
procedure CalibrateExecute(Sender: TObject);
procedure ControlExecute(Sender: TObject);
procedure ReportExecute(Sender: TObject);
procedure OptionsExecute(Sender: TObject);
procedure BazaDataExecute(Sender: TObject);
procedure AboutPrExecute(Sender: TObject);
procedure HelpContentExecute(Sender: TObject);
procedure RegimVaborClick(Sender: TObject);
procedure FormDestroy(Sender: TObject);
procedure ViewBtnClick(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure SetupConnectExecute(Sender: TObject);
procedure EditDeviceClick(Sender: TObject);
procedure FormResize(Sender: TObject);
procedure FormPaint(Sender: TObject);
procedure PaintBox2Paint(Sender: TObject);
procedure PaintBox1Paint(Sender: TObject);
procedure PaintBox3Paint(Sender: TObject);
procedure PaintBox4Paint(Sender: TObject);
procedure PaintBox5Paint(Sender: TObject);
procedure PaintBox6Paint(Sender: TObject);
procedure RegOutDataClick(Sender: TObject);
procedure VidGraphBtnClick(Sender: TObject);
procedure VidGraphicsExecute(Sender: TObject);
procedure ViewMainItemClick(Sender: TObject);
procedure ViewIzmerItemClick(Sender: TObject);
procedure ViewGraduirItemClick(Sender: TObject);
procedure ViewControlItemClick(Sender: TObject);
procedure ViewPhoneItemClick(Sender: TObject);
procedure CloseGirlForms(Sender: TObject);
procedure DdeExecuteMacro(Sender: TObject; Msg: TStrings);
procedure AMKReportDdeConvExecuteMacro(Sender: TObject; Msg: TStrings);
procedure AMKDataDdeConvExecuteMacro(Sender: TObject; Msg: TStrings);
procedure AMKDataStrDdeConvExecuteMacro(Sender: TObject; Msg: TStrings);
procedure ViewMinWinItemClick(Sender: TObject);
procedure MinFormBtnClick(Sender: TObject);
procedure MinimumBtnClick(Sender: TObject);
procedure AddFormBtnClick(Sender: TObject);
procedure AbouteDeviceItemClick(Sender: TObject);
procedure Panel1aDblClick(Sender: TObject);
procedure PaintBox1DblClick(Sender: TObject);
procedure Edit1DblClick(Sender: TObject);
procedure Label1aDblClick(Sender: TObject);
procedure Label1bDblClick(Sender: TObject);
procedure Panel2aDblClick(Sender: TObject);
procedure PaintBox2DblClick(Sender: TObject);
procedure Edit2DblClick(Sender: TObject);
procedure Label2aDblClick(Sender: TObject);
procedure Label2bDblClick(Sender: TObject);
procedure Panel3aDblClick(Sender: TObject);
procedure PaintBox3DblClick(Sender: TObject);
procedure Edit3DblClick(Sender: TObject);
procedure Label3aDblClick(Sender: TObject);
procedure Label3bDblClick(Sender: TObject);
```

```
    procedure Panel4aDblClick(Sender: TObject);
    procedure PaintBox4DblClick(Sender: TObject);
    procedure Edit4DblClick(Sender: TObject);
    procedure Label4aDblClick(Sender: TObject);
    procedure Label4bDblClick(Sender: TObject);
    procedure Panel5aDblClick(Sender: TObject);
    procedure PaintBox5DblClick(Sender: TObject);
    procedure Edit5DblClick(Sender: TObject);
    procedure Label5aDblClick(Sender: TObject);
    procedure Label5bDblClick(Sender: TObject);
    procedure Panel6aDblClick(Sender: TObject);
    procedure PaintBox6DblClick(Sender: TObject);
    procedure Edit6DblClick(Sender: TObject);
    procedure Label6aDblClick(Sender: TObject);
    procedure Label6bDblClick(Sender: TObject);
    procedure DataTableBeforeClose(DataSet: TDataSet);
    procedure DataTableAfterEdit(DataSet: TDataSet);
    procedure DataTableAfterInsert(DataSet: TDataSet);
    procedure DataTableAfterDelete(DataSet: TDataSet);
    procedure DataTableAfterPost(DataSet: TDataSet);
    procedure GetConstantsExecute(Sender: TObject);
    procedure AnalysDataExecute(Sender: TObject);
    procedure GetTestPPUExecute(Sender: TObject);
  private
    AmplitGr:array[0..8] of Real;
    ViewAmplitGr:Byte;
    CurStartSave,StartTime,StopTime,CurStartRing:tDateTime;
    StartTimeWorkReg,TimeCloseProgram,CheckTimeDDE:tDateTime;
    isNewReStartProgam,ModifDataBase:Boolean;
    TimeStartAvtoIzmer:Cardinal;
    FileNameArrStr:String;
    function  OpenMyDataBase:Boolean;
    procedure ChangeRegistrWindows;
    procedure RecalcStartSave;
    procedure CloseAnalysis;
    procedure OpenMenu;
    procedure CloseMenu;
    procedure UpdateViewBtnEnabled;
    procedure MakeMsgToDataDDE(const Msg:String);
    procedure MakeMsgToDataStrDDE(const Msg:String);
    procedure MakeMsgToReportDDE(const Msg:String);
    procedure MakePanel1;
    procedure MakePanel2;
    procedure MakePanel3;
    procedure MakePanel4;
    procedure MakePanel5;
    procedure MakePanel6;
    procedure MakeAllPaneles;
    procedure UpdateColorEditFont;
    procedure UpdatePanel;
    procedure ClearAmplitGr;
    procedure UpdateAmplitGr;
    procedure ClearEdit;
    procedure UpdateEdit;
    procedure UpdateGraphics;
    procedure DrawAmplitGr_1(MyColor:TColor);
    procedure DrawAmplitGr_2(MyColor:TColor);
    procedure DrawAmplitGr_3(MyColor:TColor);
```

```
    procedure DrawAmplitGr_4(MyColor:TColor);
    procedure DrawAmplitGr_5(MyColor:TColor);
    procedure DrawAmplitGr_6(MyColor:TColor);
    procedure DrawAmplitGrAll(MyColor:TColor);
    procedure MyDelay(Durat1:Real);
    procedure GetStartStopTime;
    function  UpdateTimeCikle(var Data1:TGlobData):Real;
    procedure SetFirstCurStartSave;
    procedure SetNewCurStartSave;
    procedure SaveDataFile;
    procedure SaveReportFile;
    procedure SaveInfoFile;
    procedure SaveDataBase;
    procedure ControlSaveFiles;
    procedure UpdateDataStrDDE(Tim1:TDateTime; err1:Byte);
    procedure SaveDataForDDE;
    procedure SaveReportForDDE;
    procedure SaveArrStrFile;
    procedure ControlUpdateDDE;
    procedure OutAllDataToMainForm;
    procedure ControlDataMainForm;
    procedure UpdateStatusWork;
    procedure ControlWorkDevice;
    procedure ControlWorkModems;
    procedure StartReadDataUGI;
    procedure StopReadData;
    procedure ControlReadDataUGI;
    procedure CheckErrorsDevice;
    procedure GetDataUGI;
  public
    { Public declarations }
    ZapretReadCOMbuf,BreakReadDataUGI:Bool;
    CheckTimeInQue,ChekTimeOutSred,CheckTimeRestModems:TDateTime;
    MinFormCaption:String;
    IsRunAutomatic:Boolean;
    procedure OpenDevice;
    procedure CloseDevice;
  end;

var
  MainForm: TMainForm;

implementation
{$R *.DFM}
uses UnAMKini,UnAMKport,ApllDlgs,UnAMKsys,UnMeteo,Options,UnDevice,GraphLib,
     Izmeren,Calibrat,Control,RunIzmer,RunCalib,Report,StyleGr,
     About,AboutDevice,UnMinFrm,PPU25,TestPPU,AutSave, Phone;

const
  Capt_labUS = 'Термоанемометр';
  Capt_lab1a = 'Скорость ветра';
  Capt_lab1b = 'V, м/с';
  Capt_lab2a = 'Направление ветра';
  Capt_lab2b = 'D,';
  Capt_lab2b2 = 'D, д.у.';
  Capt_lab3a = 'Температура';
  Capt_lab4a = 'Вертикальный ветер';
  Capt_lab4b = 'w, м/с';
```

```
  Capt_lab5a = 'Давление';
  Capt_lab5b = 'P, мм.рт.ст.';
  Capt_lab5b2 = ' P, гПа';
  Capt_lab6a = 'Влажность';
  Capt_labS1 = 'S1, мм';
  Capt_labS2 = 'S2, мм';
  Capt_labS3 = 'S3, мм';
  Capt_labS4 = 'S4, мм';

  TC_a0 = 0;   Vh_a0 = 15;  Vz_a0 = 0;   P_a0 = 650;   P_a02 = 860; Hu_a0 = 50;
  TC_ad = 15;  Vh_ad = 5;   Vz_ad = 5;   P_ad = 50;    P_ad2 = 75;  Hu_ad = 15;
  TC_xd = 50;  Vh_xd = 15;  Vz_xd = 15;  P_xd = 150;   P_xd2 = 225; Hu_xd = 50;

  ACP_a0 = 12000;   Si_a0 = 150;
  ACP_ad = 4000;    Si_ad = 50;
  ACP_xd = 12000;   Si_xd = 150;

  ti_a0 = 20000;   ACPr_a0 = 12000;
  ti_ad = 5000;    ACPr_ad = 4000;
  ti_xd = 10000;   ACPr_xd = 12000;

 Msg_ComentNorm = 'Чтение данных устройства происходит нормально.';
 Msg_ComentOtkl = 'Чтение данных устройства отключено.';
 Msg_NoData =   'Нет данных';
 Msg_Zaderqka = ' Задержка';
 Msg_FormatData='  Помехи';
 Msg_Normal =   'Нормально';
 Msg_Invalid =  ' Неверно';
 Msg_Sboj =     '  Сбой';
 Msg_OtklStatus = 'Отключено';
 Msg_Chtenie = 'Чтение данных';
 Msg_Chtenie1 = 'устройства';
 Msg_DB_er0 = 'Не могу открыть базу данных ';
 Msg_DB_er1 = 'Не могу модифицировать базу данных, открытую в режиме "Только для чтения".';
 MsgForDDE_0 = 'Данные отсутствуют, поскольку измерения не производились';
 MsgForDDE_r = 'Выполняется запуск измерений';
 MsgForDDE_n = 'Выполняется первичный набор данных';
 MsgForDDE_e = 'Не поступают данные из устройства';
 MsgForDDE_s = 'Систематические сбои в данных устройства';
 Msg_AvtoZapusk = 'Автомат. запуск';
 Msg_IntTime = 'через время';

//*****************************************
procedure TMainForm.SetupConnectExecute(Sender: TObject);
begin
  OpenPhoneForm;
end;

//*****************************************
procedure TMainForm.ParadoxPassword(Sender: TObject; var Continue: Boolean);
begin
  Continue := true;
  Session.AddPassword('798');
end;

{*********************************}
function TMainForm.OpenMyDataBase:Boolean;
begin
```

```
  try
  Result:=true;
  DataTable.Open;
  ModifDataBase:=false;
  except
  Result:=false;
  end;
end;

{*********************************}
procedure TMainForm.ChangeRegistrWindows;
var
  Registr:TRegistry;
begin
  try
  Registr:=TRegistry.Create;
  Registr.RootKey:=HKEY_LOCAL_MACHINE;
  Registr.OpenKey('Software\Microsoft\Windows\'+'CurrentVersion\Run',false);
  if IsAvtoLoadProgram then Registr.WriteString(Application.Title,ParamStr(0))
                  else Registr.DeleteValue(Application.Title);
  Registr.Free;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.RecalcStartSave;
label 1,2;
var
  Hour0,Min0,Sec0,MSec0,dt:Word;
  m0,m1,m2,m3:Integer;
begin
  DecodeTime(SysUtils.Time,Hour0,Min0,Sec0,MSec0);
  dt:=TimeUsredn div 60;
  if Min0-dt<60 then Min0:=Min0+dt
             else begin
             Hour0:=Hour0+1;
             Min0:=Min0+dt-60;
             end;
  m0:=60*Hour0+Min0;
  m1:=60*BegHourAutSave+BegMinAutSave;
  m2:=60*InterHourAutSave+InterMinAutSave;
  if m0=m1 then m3:=m1
    else begin
    if m0>m1 then begin
      m3:=m1;
    1:m3:=m3+m2;
      if m3<m0 then GOTO 1;
      if m3>1440 then m3:=m3-1440;
      end else begin
      m3:=m1;
    2:m3:=m3-m2;
      if m3>m0 then GOTO 1;
      m3:=m3+m2;
      end;
    end;
  BegHourAutSave:=m3 div 60;
```

```
  BegMinAutSave:=m3-60*BegHourAutSave;
end;

{*********************************}
procedure TMainForm.IzmerenExecute(Sender: TObject);
label 1;
var
  F1,F2,F3:Boolean;
begin
  try
  FileNameArrStr:=GetNameFileDef(Now);
  FileNameArrStr:=Format('%s%s%s%s',[DataDir,'\',FileNameArrStr,'T']);
  if not IsAvtoLoadProgram then begin
  1:F1:=GetParamIzmeren;
    F3:=IsSaveDataFile or IsSaveReportFile or
       IsSaveInfoFile or IsSaveBazaData or IsSaveArrStrFile;
    F2:=true;
    if F1 and (F3 or (RegimIzmer=1)) then F2:=GetParamAutSave;
    if not F2 then AvtoZapuskIzm:=false;
    if not (F1 and F2) then Exit;
    if AvtoZapuskIzm then begin
      TimeStartAvtoIzmer:=GetTickCount+50000000;
      IsAvtoLoadProgram:=true;
      WriteIniFile;
      ChangeRegistrWindows;
      end;
    end else RecalcStartSave;//if not IsAvtoLoadProgram
  if IsSaveBazaData then begin
    F1:=true;
    if not DataTable.Active then F1:=OpenMyDataBase;
    if not F1 then begin
      GetMyErrorDlg(Msg_DB_er0+'M_DB'+regNumber);
      IsSaveBazaData:=F1;
      if not IsAvtoLoadProgram then GOTO 1;
      end;
    if DataTable.Active then begin
      DataTable.Close;
      end;
    end;
  ControlReadDataUGI;
  SetFirstCurStartSave;
  case RegimIzmer of
  0:begin
      UpdateStartIzmeren(Sender);
      StartReadDataUGI;
    end;
  1:begin
      RegimVabor.ItemIndex:=0;
      ClearAmplitGr;
      IsSbojPhone:=false;
      CloseMenu;
      IsRunAutomatic:=true;
    end;
  end;
  except
  raise
  end;
end;
```

```
{*********************************}
procedure TMainForm.CalibrateExecute(Sender: TObject);
var
  W:Integer;
  Flag1:Boolean;
begin
  W:=GetMyConfirm2Dlg(Msg_Warning+Msg_ChangeGraduir);
  if W <> IdYes then Exit;
  Flag1:=(ParolCurrent=Par_System) or (ParolCurrent=Par_Standart);
  if not Flag1 then Flag1:=GetAdminParol(Par_System,Par_Standart,ParolCurrent);
  if not Flag1 then Exit;
  try
  TkalibrNew:=Tcalibr;
  if GetParamSonicCalib(TkalibrNew,TimeUsredn) then Begin
    ControlReadDataUGI;
    if IsOpenCOMport then begin
    UpdateStartCalibr(Sender);
    StartReadDataUGI;
    end;
    End;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.ControlExecute(Sender: TObject);
begin
  try
  ControlReadDataUGI;
  if IsOpenCOMport then begin
    UpdateStartControl(Sender);
    StartReadDataUGI;
    end;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.ReportExecute(Sender: TObject);
begin
  if Met_Data=nil then Exit;
  if Length(Met_Data)<3 then Exit;
  try
  ReportForm.ViewReportForm(Capt_Viewer);
  except
  raise;
  end;
end;

{*********************************}
procedure TMainForm.ExitProgrExecute(Sender: TObject);
begin
  Close;
end;
```

```
{*********************************}
procedure TMainForm.OptionsExecute(Sender: TObject);
begin
  CloseGirlForms(Sender);
  GetOptionsDlg;
  AddDefault;
  FormPaint(Sender);
  UpdateAllGraph(Sender);
  Timer1.Interval:=TimeTimerSpeed;
  if IsModem then GetConstants.Enabled:=false
          else GetConstants.Enabled:=true;
end;

{*******************************************************}
procedure TMainForm.VidGraphicsExecute(Sender: TObject);
begin
  GetStyleGraphDlg;
  UpdateAllGraph(Sender);
end;

{*********************************}
procedure TMainForm.BazaDataExecute(Sender: TObject);
var
  f1:array[0..255] of Char;
  msg1:String;
begin
  if FileExists(BasaFileName) then begin
  StrPCopy(f1,BasaFileName);
  ShellExecute(Handle,nil,f1,nil,nil,SW_RESTORE);
  end else begin
  msg1:=Format('%s%s%s',[Msg_File_0,BasaFileName,Msg_File_NoFind]);
  GetMyErrorDlg(Msg1);
  end;
end;

{*****************************************}
procedure TMainForm.AnalysDataExecute(Sender: TObject);
var
  f1:array[0..255] of Char;
  msg1:String;
begin
  if FileExists(AnalysisFileName) then begin
  StrPCopy(f1,AnalysisFileName);
  ShellExecute(Handle,nil,f1,nil,nil,SW_RESTORE);
  end else begin
  msg1:=Format('%s%s%s',[Msg_File_0,AnalysisFileName,Msg_File_NoFind]);
  GetMyErrorDlg(Msg1);
  end;
end;

{*********************************************}
procedure TMainForm.GetConstantsExecute(Sender: TObject);
begin
  GetPPU25Dlg;
end;

//***********************************************
procedure TMainForm.GetTestPPUExecute(Sender: TObject);
```

```
begin
  GetTestPPUdlg;
end;

{*********************************}
procedure TMainForm.AboutPrExecute(Sender: TObject);
begin
  GetAboutDlg;
end;

{**********************************************}
procedure TMainForm.AbouteDeviceItemClick(Sender: TObject);
begin
  GetDeviceDlg;
end;

{*********************************}
procedure TMainForm.HelpContentExecute(Sender: TObject);
var
  f1:array[0..255] of Char;
  msg1:String;
begin
  if FileExists(Application.HelpFile) then begin
  StrPCopy(f1,Application.HelpFile);
  ShellExecute(Handle,nil,f1,nil,nil,SW_RESTORE);
  end else begin
  msg1:=Format('%s%s%s',[Msg_File_0,Application.HelpFile,Msg_File_NoFind]);
  GetMyErrorDlg(Msg1);
  end;
end;

{*****************************************************}
procedure TMainForm.DdeExecuteMacro(Sender: TObject; Msg: TStrings);
begin
  try
  if Msg.Strings[0]='Close_METEO_3_0' then
     TimeCloseProgram:=Now+EncodeTime(0,0,3,0)
  else begin
  if Izmeren.Enabled then Begin
  TimeUsredn:=60*StrToInt(Copy(Msg.Strings[0],1,2));
  if Length(Msg.Strings[0])>=7 then
  DeviceOrient:=StrToInt(Copy(Msg.Strings[0],3,5))/100;
  if Length(Msg.Strings[0])>=12 then
  UGI_Z:=StrToInt(Copy(Msg.Strings[0],8,5))/100;
  if Length(Msg.Strings[0])>=13 then
  TypeSurface:=StrToInt(Copy(Msg.Strings[0],13,1));
  MainForm.WindowState:=wsNormal;
  MainForm.Show;
  Invalidate;
  Update;
  IzmerenExecute(Sender);
  End;
  end;
  except
  raise
  end;
end;
```

```
{*****************************************************}
procedure TMainForm.AMKReportDdeConvExecuteMacro(Sender: TObject; Msg: TStrings);
begin
  try
  if Msg.Count<=0 then Exit;
  if Length(Msg.Strings[0])<2 then Exit;
  DdeExecuteMacro(Sender,Msg);
  except
  end;
end;

{*****************************************************}
procedure TMainForm.AMKDataDdeConvExecuteMacro(Sender: TObject; Msg: TStrings);
begin
  try
  if Msg.Count<=0 then Exit;
  if Length(Msg.Strings[0])<2 then Exit;
  DdeExecuteMacro(Sender,Msg);
  except
  end;
end;

//*****************************************************
procedure TMainForm.AMKDataStrDdeConvExecuteMacro(Sender: TObject; Msg: TStrings);
begin
  try
  if Msg.Count<=0 then Exit;
  if Length(Msg.Strings[0])<2 then Exit;
  DdeExecuteMacro(Sender,Msg);
  except
  end;
end;

//*******************************************
procedure TMainForm.FormCreate(Sender: TObject);
const
  Zaderhka=4;
var
  StartTime1,CurTime1:Longint;
begin
  try
  Timer1.Interval:=TimeTimerSpeed;
  Screen.Cursor:=crHourGlass;
  StartTime1:=GetTickCount div 1000;
  ZapretReadCOMbuf:=true;
  BreakReadDataUGI:=true;
  DecimalSeparator:='.';
  TimeCloseProgram:=Date+2000;
  ReadIniFile;
  InitDefault;
  Application.Title:=Format('%s%s%s',[NameDevice,', № ',regNumber]);
  Application.HelpFile:=Format('%s%s%s',[ProgramDir,'\','METEO_RU.HLP']);
  Caption:=Format('%s%s%s%s%s',[st_FullNamePr,' ',st_BriefNamePr,' : ',Application.Title]);
  MinFormCaption:=Format('%s%s%s',[st_BriefNamePr,' : ',Application.Title]);
  LineCaption_R:=Format('%s%s%s',[LineCaption_R,regNumber,'     *']);
  Top:=0;
  Left:=0;
  Height:=520;
```

```
  if Height > Screen.Height-32 then Height:=Screen.Height-32;
  Width:=690;
  if Width>Screen.Width then Width:=Screen.Width;
  ViewAmplitGr:=0;
  StartTimeWorkReg:=Now;
  IsRunAutomatic:=false;
  Report.Enabled:=false;
  IsPhoneForm:=false;
  IsConnectPhone:=false;
  IsSbojPhone:=false;
  SetupConnect.Enabled:=isModem;
  ViewPhoneItem.Enabled:=SetupConnect.Enabled;
  if IsTehnologyDDE then begin
    MakeMsgToDataDDE(MsgForDDE_0);
    MakeMsgToDataStrDDE(MsgForDDE_0);
    MakeMsgToReportDDE(MsgForDDE_0);
    end;
  if IsModem then CloseDevice else OpenDevice;
  if IsModem then GetConstants.Enabled:=false
          else GetConstants.Enabled:=true;
  UpdateGraphics;
  Application.OnHint := ShowHint;
  DataTable.DatabaseName:='M_DB'+regNumber;
  DataTable.TableName:='M_DB'+regNumber+'.db';
  Session.OnPassword := ParadoxPassword;
  BasaFileName := 'M_DB'+regNumber+'.EXE';
  if FileExists(BasaFileName) then BazaData.Enabled:=true else begin
     IsSaveBazaData:=false;
     BazaData.Enabled:=false;
     end;
  AnalysisFileName := 'M_DP'+regNumber+'.EXE';
  if FileExists(AnalysisFileName) then AnalysData.Enabled:=true
                        else AnalysData.Enabled:=false;
  CurTime1:=GetTickCount div 1000;
  if CurTime1 < (StartTime1+Zaderhka) then
  while CurTime1 < (StartTime1+Zaderhka) do
  CurTime1:=GetTickCount div 1000;
  if IsAvtoLoadProgram then begin
    isNewReStartProgam:=true;
    TimeStartAvtoIzmer:=GetTickCount+1000;
    end else isNewReStartProgam:=false;
  finally
  Screen.Cursor:=crDefault;
  end;
end;

{*********************************************}
procedure TMainForm.FormResize(Sender: TObject);
var
  w1:Integer;
begin
  w1:=PanelInform.Width-BevRight.Left-3;
  if w1>1 then BevRight.Width:=w1;
  UpdatePanel;
end;

{*************************************************}
procedure TMainForm.ShowHint(Sender: TObject);
```

```
begin
  StatusBar0.SimpleText := Application.Hint;
end;

{*****************************************************}
procedure TMainForm.DataTableAfterEdit(DataSet: TDataSet);
begin
  ModifDataBase:=true;
end;
procedure TMainForm.DataTableAfterInsert(DataSet: TDataSet);
begin
  ModifDataBase:=true;
end;
procedure TMainForm.DataTableAfterDelete(DataSet: TDataSet);
begin
  ModifDataBase:=true;
end;

{*************************************************}
procedure TMainForm.DataTableAfterPost(DataSet: TDataSet);
begin
  if  DataTable.CachedUpdates and ModifDataBase then begin
    DataTable.ApplyUpdates;
    DataTable.CommitUpdates;
  end;
  DbiSaveChanges(DataTable.Handle);
  ModifDataBase:=false;
end;

{*************************************************}
procedure TMainForm.DataTableBeforeClose(DataSet: TDataSet);
begin
  if  DataTable.CachedUpdates and ModifDataBase then
     DataTableAfterPost(DataSet);
end;

{*********************************}
procedure TMainForm.FormDestroy(Sender: TObject);
begin
  try
  if DataTable.Active then DataTable.Close;
  ClearGlobData(Con_Data);
  ClearGlobData(Cal_Data);
  ClearGlobData(Met_Data);
  IniFile.Free;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  try
  if not BreakReadDataUGI then begin
  BreakReadDataUGI:=true;
  Timer1Timer(Sender);
  end;
```

```
  SetPPU_inWork(Handle_AMK,OverLap_AMK);;
  Close_AMKport(Handle_AMK,IsOpenCOMport);
  if IsAvtoLoadProgram then begin
    IsAvtoLoadProgram:=false;
    ChangeRegistrWindows;
    end;
  WriteIniFile;
  Action:=caFree;
  except
  end;
end;

{*******************************************}
procedure TMainForm.CloseGirlForms(Sender: TObject);
begin
  try
  if IzmerenForm.Visible then IzmerenForm.CloseBtnClick(Sender);
  if CalibUGIForm.Visible then CalibUGIForm.CloseBtnClick(Sender);
  if ControlUGIForm.Visible then ControlUGIForm.CloseBtnClick(Sender);
  if ReportForm.Visible then ReportForm.CloseBtnClick(Sender);
  except
  raise
  end;
end;

{*******************************************}
procedure TMainForm.ViewMinWinItemClick(Sender: TObject);
begin
  try
  CloseGirlForms(Sender);
  if IsPhoneForm then
     if PhoneForm.Visible then PhoneForm.Hide;
  MinForm.Show;
  MinForm.BringToFront;
  MainForm.Hide;
  except
  raise
  end;
end;
procedure TMainForm.MinFormBtnClick(Sender: TObject);
begin
  ViewMinWinItemClick(Sender);
end;
procedure TMainForm.MinimumBtnClick(Sender: TObject);
begin
  ViewMinWinItemClick(Sender);
end;

{*************************************}
procedure TMainForm.ViewMainItemClick(Sender: TObject);
begin
  MainForm.BringToFront;
end;

{*************************************}
procedure TMainForm.ViewIzmerItemClick(Sender: TObject);
begin
  if not IzmerenForm.Visible then IzmerenForm.Show;
```

```
  IzmerenForm.BringToFront;
end;

{*************************************}
procedure TMainForm.ViewGraduirItemClick(Sender: TObject);
begin
  if not CalibUgiForm.Visible then CalibUgiForm.Show;
  CalibUgiForm.BringToFront;
end;

{*************************************}
procedure TMainForm.ViewControlItemClick(Sender: TObject);
begin
  if not ControlUgiForm.Visible then ControlUgiForm.Show;
  ControlUgiForm.BringToFront;
end;

{*************************************}
procedure TMainForm.ViewPhoneItemClick(Sender: TObject);
begin
  if not IsPhoneForm then SetupConnectExecute(Sender)
  else begin
    if not PhoneForm.Visible then PhoneForm.Show;
    PhoneForm.BringToFront;
    end;
end;

{*********************************}
procedure TMainForm.UpdateViewBtnEnabled;
begin
  ViewBtn.Enabled:=false;
  case RegimVabor.ItemIndex of
   0:if IzmerenForm<>nil then begin
       if IzmerenForm.Visible then ViewBtn.Enabled:=false
                          else ViewBtn.Enabled:=true;
       end;
   1:if CalibUgiForm<>nil then begin
       if CalibUgiForm.Visible then ViewBtn.Enabled:=false
                           else ViewBtn.Enabled:=true;
       end;
   2:if ControlUgiForm<>nil then begin
       if ControlUgiForm.Visible then ViewBtn.Enabled:=false
                            else ViewBtn.Enabled:=true;
       end;
   end; {case}
end;

{*********************************}
procedure TMainForm.ViewBtnClick(Sender: TObject);
begin
  CloseGirlForms(Sender);
  case RegimVabor.ItemIndex of
   0:ViewIzmerItemClick(Sender);
   1:ViewGraduirItemClick(Sender);
   2:ViewControlItemClick(Sender);
  end;
  UpdateViewBtnEnabled;
end;
```

```
procedure TMainForm.AddFormBtnClick(Sender: TObject);
begin
  ViewBtnClick(Sender);
end;

{*****************************************}
procedure TMainForm.EditDeviceClick(Sender: TObject);
begin
  if not ZapretReadCOMbuf then begin
  case Data_Ugi[0] of
    4:GetMyInformDlg(MsgErr4);
    5:GetMyInformDlg(MsgErr5);
    6:GetMyInformDlg(MsgErr6);
    7:GetMyInformDlg(MsgErr7);
    8:GetMyInformDlg(MsgErr8);
    9:GetMyInformDlg(MsgErr9_0+MsgErr9_1);
   10:GetMyInformDlg(MsgErr101+MsgErr10_1);
   11:GetMyInformDlg(MsgErr11);
    else GetMyInformDlg(Msg_ComentNorm);
  end;
  end else GetMyInformDlg(Msg_ComentOtkl);
end;

{*********************************}
procedure TMainForm.RegimVaborClick(Sender: TObject);
begin
  FormPaint(Sender);
end;

{*****************************************}
procedure TMainForm.RegOutDataClick(Sender: TObject);
begin
  FormPaint(Sender);
end;

{*****************************************}
procedure TMainForm.VidGraphBtnClick(Sender: TObject);
begin
  VidGraphicsExecute(Sender);
end;

{*****************************************}
procedure TMainForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if StopBtn.Enabled then
  if Key = VK_ESCAPE then StopBtnClick(Sender);
end;

{*****************************************}
procedure TMainForm.Panel1aDblClick(Sender: TObject);
begin
  if ViewAmplitGr=0 then ViewAmplitGr:=1
                 else ViewAmplitGr:=0;
  FormPaint(Sender);
end;
procedure TMainForm.PaintBox1DblClick(Sender: TObject);
begin
```

```
  Panel1aDblClick(Sender);
end;
procedure TMainForm.Edit1DblClick(Sender: TObject);
begin
  Panel1aDblClick(Sender);
end;
procedure TMainForm.Label1aDblClick(Sender: TObject);
begin
  Panel1aDblClick(Sender);
end;
procedure TMainForm.Label1bDblClick(Sender: TObject);
begin
  Panel1aDblClick(Sender);
end;

{*********************************************}
procedure TMainForm.Panel2aDblClick(Sender: TObject);
begin
  if ViewAmplitGr=0 then ViewAmplitGr:=2
                else ViewAmplitGr:=0;
  FormPaint(Sender);
end;
procedure TMainForm.PaintBox2DblClick(Sender: TObject);
begin
  Panel2aDblClick(Sender);
end;
procedure TMainForm.Edit2DblClick(Sender: TObject);
begin
  Panel2aDblClick(Sender);
end;
procedure TMainForm.Label2aDblClick(Sender: TObject);
begin
  Panel2aDblClick(Sender);
end;
procedure TMainForm.Label2bDblClick(Sender: TObject);
begin
  Panel2aDblClick(Sender);
end;

{*********************************************}
procedure TMainForm.Panel3aDblClick(Sender: TObject);
begin
  if ViewAmplitGr=0 then ViewAmplitGr:=3
                else ViewAmplitGr:=0;
  FormPaint(Sender);
end;
procedure TMainForm.PaintBox3DblClick(Sender: TObject);
begin
  Panel3aDblClick(Sender);
end;
procedure TMainForm.Edit3DblClick(Sender: TObject);
begin
  Panel3aDblClick(Sender);
end;
procedure TMainForm.Label3aDblClick(Sender: TObject);
begin
  Panel3aDblClick(Sender);
end;
```

```
procedure TMainForm.Label3bDblClick(Sender: TObject);
begin
  Panel3aDblClick(Sender);
end;

{****************************************}
procedure TMainForm.Panel4aDblClick(Sender: TObject);
begin
  if ViewAmplitGr=0 then ViewAmplitGr:=4
                else ViewAmplitGr:=0;
  FormPaint(Sender);
end;
procedure TMainForm.PaintBox4DblClick(Sender: TObject);
begin
  Panel4aDblClick(Sender);
end;
procedure TMainForm.Edit4DblClick(Sender: TObject);
begin
  Panel4aDblClick(Sender);
end;
procedure TMainForm.Label4aDblClick(Sender: TObject);
begin
  Panel4aDblClick(Sender);
end;
procedure TMainForm.Label4bDblClick(Sender: TObject);
begin
  Panel4aDblClick(Sender);
end;

{*************************************}
procedure TMainForm.Panel5aDblClick(Sender: TObject);
begin
  if ViewAmplitGr=0 then ViewAmplitGr:=5
                else ViewAmplitGr:=0;
  FormPaint(Sender);
end;
procedure TMainForm.PaintBox5DblClick(Sender: TObject);
begin
  Panel5aDblClick(Sender);
end;
procedure TMainForm.Edit5DblClick(Sender: TObject);
begin
  Panel5aDblClick(Sender);
end;
procedure TMainForm.Label5aDblClick(Sender: TObject);
begin
  Panel5aDblClick(Sender);
end;
procedure TMainForm.Label5bDblClick(Sender: TObject);
begin
  Panel5aDblClick(Sender);
end;

{****************************************}
procedure TMainForm.Panel6aDblClick(Sender: TObject);
begin
  if ViewAmplitGr=0 then ViewAmplitGr:=6
                else ViewAmplitGr:=0;
```

```
  FormPaint(Sender);
end;
procedure TMainForm.PaintBox6DblClick(Sender: TObject);
begin
  Panel6aDblClick(Sender);
end;
procedure TMainForm.Edit6DblClick(Sender: TObject);
begin
  Panel6aDblClick(Sender);
end;
procedure TMainForm.Label6aDblClick(Sender: TObject);
begin
  Panel6aDblClick(Sender);
end;
procedure TMainForm.Label6bDblClick(Sender: TObject);
begin
  Panel6aDblClick(Sender);
end;

{*************************************}
procedure TMainForm.MakePanel1;
var
  k:Integer;
begin
  if (ViewAmplitGr=1) or (ViewAmplitGr=2) then Begin
  Panel1a.Width:=PanelSredn.Width-16;
  Panel1a.Height:=PanelSredn.Height-16;
  Panel1a.Left:=8;
  Panel1a.Top:=8;
  Label1a.Font.Size:=18;
  Edit1.Font.Size:=24;
  Edit1.Width:=112;
  Edit1.Height:=40;
  End else Begin
  Panel1a.Left:=8;
  Panel1a.Top:=8;
  Panel1a.Width:=(PanelSredn.Width-22) div 3;
  Panel1a.Height:=(PanelSredn.Height-19) div 2;
  Label1a.Font.Size:=10;
  Edit1.Font.Size:=14;
  Edit1.Width:=70;
  Edit1.Height:=28;
  End;
  k:=(Panel1a.Width-Label1a.Width) div 2;
  if k<2 then k:=2;
  Label1a.Left:=k;
  Label1b.Top:=6+Label1a.Height;
  k:=(Panel1a.Width-Label1b.Width) div 2;
  if k<2 then k:=2;
  Label1b.Left:=k;
  Panel1b.Top:=9+Label1a.Height+Label1b.Height;
  Panel1b.Height:=Panel1a.Height-Panel1b.Top-6;
  Panel1b.Left:=6;
  Panel1b.Width:=Panel1a.Width-12;
  Edit1.Left:=Panel1b.Left+3;
  Edit1.Top:=Panel1b.Top+3;
end;
```

```
{*************************************}
procedure TMainForm.MakePanel2;
var
  k,h1:Integer;
begin
  if (ViewAmplitGr=1) or (ViewAmplitGr=2) then Begin
  Panel2a.Top:=8;
  Panel2a.Height:=PanelSredn.Height-16;
  Label2a.Font.Size:=18;
  Edit2.Font.Size:=24;
  Edit2.Width:=112;
  Edit2.Height:=40;
  Panel2a.Left:=8;
  Panel2a.Width:=PanelSredn.Width-16;
  End else Begin
  h1:=(PanelSredn.Height-19) div 2;
  Panel2a.Left:=8;
  Panel2a.Top:=11+h1;
  Panel2a.Width:=(PanelSredn.Width-22) div 3;
  Panel2a.Height:=h1;
  Label2a.Font.Size:=10;
  Edit2.Font.Size:=14;
  Edit2.Width:=70;
  Edit2.Height:=28;
  End;
  k:=(Panel2a.Width-Label2a.Width) div 2;
  if k<2 then k:=2;
  Label2a.Left:=k;
  Label2b.Top:=6+Label2a.Height;
  k:=(Panel2a.Width-Label2b.Width) div 2;
  if k<2 then k:=2;
  Label2b.Left:=k;
  Panel2b.Top:=9+Label2a.Height+Label2b.Height;
  Panel2b.Height:=Panel2a.Height-Panel2b.Top-6;
  Panel2b.Left:=6;
  Panel2b.Width:=Panel2a.Width-12;
  Edit2.Left:=Panel2b.Left+3;
  Edit2.Top:=Panel2b.Top+3;
  Label2c.Left:=Label2b.Left+12;
  Label2c.Top:=Label2b.Top-5;
end;

{*************************************}
procedure TMainForm.MakePanel3;
var
  k,w1:Integer;
begin
  if ViewAmplitGr=3 then Begin
  Panel3a.Left:=8;
  Panel3a.Top:=8;
  Panel3a.Width:=PanelSredn.Width-16;
  Panel3a.Height:=PanelSredn.Height-16;
  Label3a.Font.Size:=18;
  Edit3.Font.Size:=24;
  Edit3.Width:=112;
  Edit3.Height:=40;
  End else Begin
  w1:=(PanelSredn.Width-22) div 3;
```

```
  Panel3a.Left:=11+w1;
  Panel3a.Top:=8;
  Panel3a.Width:=w1;
  Panel3a.Height:=(PanelSredn.Height-19) div 2;
  Label3a.Font.Size:=10;
  Edit3.Font.Size:=14;
  Edit3.Width:=70;
  Edit3.Height:=28;
  End;
  k:=(Panel3a.Width-Label3a.Width) div 2;
  if k<2 then k:=2;
  Label3a.Left:=k;
  Label3b.Top:=6+Label3a.Height;
  k:=(Panel3a.Width-Label3b.Width) div 2;
  if k<2 then k:=2;
  Label3b.Left:=k;
  Panel3b.Top:=9+Label3a.Height+Label3b.Height;
  Panel3b.Height:=Panel3a.Height-Panel3b.Top-6;
  Panel3b.Left:=6;
  Panel3b.Width:=Panel3a.Width-12;
  Edit3.Left:=Panel3b.Left+3;
  Edit3.Top:=Panel3b.Top+3;
  Label3c.Left:=Label3b.Left+12;
  Label3c.Top:=Label3b.Top-5;
end;

{*************************************}
procedure TMainForm.MakePanel4;
var
  k,h1,w1:Integer;
begin
  if ViewAmplitGr=4 then Begin
  Panel4a.Left:=8;
  Panel4a.Top:=8;
  Panel4a.Width:=PanelSredn.Width-16;
  Panel4a.Height:=PanelSredn.Height-16;
  Label4a.Font.Size:=18;
  Edit4.Font.Size:=24;
  Edit4.Width:=112;
  Edit4.Height:=40;
  End else Begin
  h1:=(PanelSredn.Height-19) div 2;
  w1:=(PanelSredn.Width-22) div 3;
  Panel4a.Left:=11+w1;
  Panel4a.Top:=11+h1;
  Panel4a.Width:=w1;
  Panel4a.Height:=h1;
  Label4a.Font.Size:=10;
  Edit4.Font.Size:=14;
  Edit4.Width:=70;
  Edit4.Height:=28;
  End;
  k:=(Panel4a.Width-Label4a.Width) div 2;
  if k<2 then k:=2;
  Label4a.Left:=k;
  Label4b.Top:=6+Label4a.Height;
  k:=(Panel4a.Width-Label4b.Width) div 2;
  if k<2 then k:=2;
```

```
  Label4b.Left:=k;
  Panel4b.Top:=9+Label4a.Height+Label4b.Height;
  Panel4b.Height:=Panel4a.Height-Panel4b.Top-6;
  Panel4b.Left:=6;
  Panel4b.Width:=Panel4a.Width-12;
  Edit4.Left:=Panel4b.Left+3;
  Edit4.Top:=Panel4b.Top+3;
end;

{*************************************}
procedure TMainForm.MakePanel5;
var
  k,h1,w1:Integer;
begin
  if ViewAmplitGr=5 then Begin
  Panel5a.Left:=8;
  Panel5a.Top:=8;
  Panel5a.Width:=PanelSredn.Width-16;
  Panel5a.Height:=PanelSredn.Height-16;
  Label5a.Font.Size:=18;
  Edit5.Font.Size:=24;
  Edit5.Width:=112;
  Edit5.Height:=40;
  End else Begin
  h1:=(PanelSredn.Height-19) div 2;
  w1:=(PanelSredn.Width-22) div 3;
  Panel5a.Left:=14+2*w1;
  Panel5a.Top:=8;
  Panel5a.Width:=w1;
  Panel5a.Height:=h1;
  Label5a.Font.Size:=10;
  Edit5.Font.Size:=14;
  Edit5.Width:=70;
  Edit5.Height:=28;
  End;
  k:=(Panel5a.Width-Label5a.Width) div 2;
  if k<2 then k:=2;
  Label5a.Left:=k;
  Label5b.Top:=6+Label5a.Height;
  k:=(Panel5a.Width-Label5b.Width) div 2;
  if k<2 then k:=2;
  Label5b.Left:=k;
  Panel5b.Top:=9+Label5a.Height+Label5b.Height;
  Panel5b.Height:=Panel5a.Height-Panel5b.Top-6;
  Panel5b.Left:=6;
  Panel5b.Width:=Panel5a.Width-12;
  Edit5.Left:=Panel5b.Left+3;
  Edit5.Top:=Panel5b.Top+3;
end;

{*************************************}
procedure TMainForm.MakePanel6;
var
  k,h1,w1:Integer;
begin
  if ViewAmplitGr=6 then Begin
  Panel6a.Left:=8;
  Panel6a.Top:=8;
```

```
    Panel6a.Width:=PanelSredn.Width-16;
    Panel6a.Height:=PanelSredn.Height-16;
    Label6a.Font.Size:=18;
    Edit6.Font.Size:=24;
    Edit6.Width:=112;
    Edit6.Height:=40;
    End else Begin
    h1:=(PanelSredn.Height-19) div 2;
    w1:=(PanelSredn.Width-22) div 3;
    Panel6a.Left:=14+2*w1;
    Panel6a.Top:=11+h1;
    Panel6a.Width:=w1;
    Panel6a.Height:=h1;
    Label6a.Font.Size:=10;
    Edit6.Font.Size:=14;
    Edit6.Width:=70;
    Edit6.Height:=28;
    End;
    k:=(Panel6a.Width-Label6a.Width) div 2;
    if k<2 then k:=2;
    Label6a.Left:=k;
    Label6b.Top:=6+Label6a.Height;
    k:=(Panel6a.Width-Label6b.Width) div 2;
    if k<2 then k:=2;
    Label6b.Left:=k;
    Panel6b.Top:=9+Label6a.Height+Label6b.Height;
    Panel6b.Height:=Panel6a.Height-Panel6b.Top-6;
    Panel6b.Left:=6;
    Panel6b.Width:=Panel6a.Width-12;
    Edit6.Left:=Panel6b.Left+3;
    Edit6.Top:=Panel6b.Top+3;
  end;

  {*************************************}
  procedure TMainForm.MakeAllPaneles;
  begin
    if ViewAmplitGr<>0 then begin
      PanelInform.Visible:=false;
      PanelWork.Visible:=false;
      end else begin
      PanelInform.Visible:=true;
      PanelWork.Visible:=true;
      end;
    case ViewAmplitGr of
    0:begin
      Panel1a.Visible:=true;
      Panel2a.Visible:=true;
      Panel3a.Visible:=true;
      Panel4a.Visible:=true;
      Panel5a.Visible:=true;
      Panel6a.Visible:=true;
      MakePanel1;
      MakePanel2;
      MakePanel3;
      MakePanel4;
      MakePanel5;
      MakePanel6;
      end;
```

```
    1:begin
     Panel1a.Visible:=true;
     Panel2a.Visible:=false;
     Panel3a.Visible:=false;
     Panel4a.Visible:=false;
     Panel5a.Visible:=false;
     Panel6a.Visible:=false;
     MakePanel1;
     if RegimVabor.ItemIndex=0 then MakePanel2;
     end;
    2:begin
     Panel1a.Visible:=false;
     Panel2a.Visible:=true;
     Panel3a.Visible:=false;
     Panel4a.Visible:=false;
     Panel5a.Visible:=false;
     Panel6a.Visible:=false;
     if RegimVabor.ItemIndex=0 then MakePanel1;
     MakePanel2;
     end;
    3:begin
     Panel1a.Visible:=false;
     Panel2a.Visible:=false;
     Panel3a.Visible:=true;
     Panel4a.Visible:=false;
     Panel5a.Visible:=false;
     Panel6a.Visible:=false;
     MakePanel3;
     end;
    4:begin
     Panel1a.Visible:=false;
     Panel2a.Visible:=false;
     Panel3a.Visible:=false;
     Panel4a.Visible:=true;
     Panel5a.Visible:=false;
     Panel6a.Visible:=false;
     MakePanel4;
     end;
    5:begin
     Panel1a.Visible:=false;
     Panel2a.Visible:=false;
     Panel3a.Visible:=false;
     Panel4a.Visible:=false;
     Panel5a.Visible:=true;
     Panel6a.Visible:=false;
     MakePanel5;
     end;
    6:begin
     Panel1a.Visible:=false;
     Panel2a.Visible:=false;
     Panel3a.Visible:=false;
     Panel4a.Visible:=false;
     Panel5a.Visible:=false;
     Panel6a.Visible:=true;
     MakePanel6;
     end;
   end;
  end;
```

```
{*********************************}
procedure TMainForm.UpdateColorEditFont;
var
  Col1,Col2:TColor;
begin
  Col1:=GetColorFromIndex(EditColor);
  Col2:=GetColorFromIndex(DigitColor);
  Edit1.Color:=Col1;
  Edit2.Color:=Col1;
  Edit3.Color:=Col1;
  Edit1.Font.Color:=Col2;
  Edit2.Font.Color:=Col2;
  Edit3.Font.Color:=Col2;
  if RegimVabor.ItemIndex<>3 then begin
    Edit4.Visible:=true;
    Edit4.Color:=Col1;
    Edit4.Font.Color:=Col2;
    EditOrient.Visible:=true;
    end else begin
    Edit4.Visible:=false;
    EditOrient.Visible:=false;
    end;
  if (not IsDeviceP) and (RegimVabor.ItemIndex<>2) then begin
  Edit5.Color:=clSilver;
  Edit5.Font.Color:=clBlack;
  end else begin
  Edit5.Color:=Col1;
  Edit5.Font.Color:=Col2;
  end;
  if (not IsDeviceHu) and (RegimVabor.ItemIndex<>2) then begin
  Edit6.Color:=clSilver;
  Edit6.Font.Color:=clBlack;
  end else begin
  Edit6.Color:=Col1;
  Edit6.Font.Color:=Col2;
  end;
  if RegimVabor.ItemIndex<>3 then begin
  EditTr.Visible:=true;
  EditTr.Visible:=true;
  if (not IsDeviceHu) and (RegimVabor.ItemIndex<>2) then begin
  EditTr.Color:=clSilver;
  EditTr.Font.Color:=clBlack;
  end else begin
  EditTr.Color:=Col1;
  EditTr.Font.Color:=Col2;
  end;
  end else begin
  EditTr.Visible:=false;
  EditTr.Visible:=false;
  end;
  if RegimVabor.ItemIndex<>3 then begin
  EditGamX.Visible:=true;
  EditGamY.Visible:=true;
  EditGamX.Color:=Col1;
  EditGamx.Font.Color:=Col2;
  EditGamY.Color:=Col1;
  EditGamy.Font.Color:=Col2;
```

```
  end else begin
  EditGamX.Visible:=false;
  EditGamY.Visible:=false;
  end;
end;

{*********************************}
procedure TMainForm.UpdatePanel;
begin
  label4a.Visible:=true;
  label4b.Visible:=true;
  LabGammas.Visible:=true;
  labGx.Visible:=true;
  labGy.Visible:=true;
  labTr.Visible:=true;
  LabHor1.Visible:=true;
  LabHor2.Visible:=true;
case RegimVabor.ItemIndex of
  0:begin
     label1a.Caption:=Capt_lab1a;
     label1b.Caption:=Capt_lab1b;
     label2a.Caption:=Capt_lab2a;
     if SizeDirWind=1 then begin
       label2b.Caption:=Capt_lab2b;
       label2c.Visible:=true;
       end else begin
       label2b.Caption:=Capt_lab2b2;
       label2c.Visible:=false;
       end;
     label3a.Caption:=Capt_lab3a;
     label3b.Caption:='T,   C';
     label3c.Visible:=true;
     label4a.Caption:=Capt_lab4a;
     label4b.Caption:=Capt_lab4b;
     label5a.Caption:=Capt_lab5a;
     if SizeDavlen=1 then label5b.Caption:=Capt_lab5b
                 else label5b.Caption:=Capt_lab5b2;
     label6a.Caption:=Capt_lab6a;
     label6b.Caption:='r, %';
     labGx.Caption:='Gx,';
     labGxdop.Visible:=true;
     labGy.Caption:='Gy,';
     labGydop.Visible:=true;
     labTr.Caption:='Tr,   C';
     labTrdop.Visible:=true;
    end;
  1:begin
     label1b.Caption:=Capt_labS1;
     label2b.Caption:=Capt_labS2;
     label3b.Caption:=Capt_labS3;
     label4b.Caption:=Capt_labS4;
     label1a.Caption:=Capt_labUS;
     label2a.Caption:=Capt_labUS;
     label3a.Caption:=Capt_labUS;
     label4a.Caption:=Capt_labUS;
     label2c.Visible:=false;
     label3c.Visible:=false;
     label6a.Caption:=Capt_lab6a;
```

```
      label6b.Caption:='r, %';
      label5a.Caption:=Capt_lab5a;
      if SizeDavlen=1 then label5b.Caption:=Capt_lab5b
                 else label5b.Caption:=Capt_lab5b2;
      labGx.Caption:='Gx,';
      labGxdop.Visible:=true;
      labGy.Caption:='Gy,';
      labGydop.Visible:=true;
      labTr.Caption:='Tr,   C';
      labTrdop.Visible:=true;
    end;
  2:begin
      label1a.Caption:=Capt_labUS;
      label1b.Caption:='X1';
      label2a.Caption:=Capt_labUS;
      label2b.Caption:='X2';
      label2c.Visible:=false;
      label3a.Caption:=Capt_labUS;
      label3b.Caption:='X3';
      label3c.Visible:=false;
      label4a.Caption:=Capt_labUS;
      label4b.Caption:='X4';
      label5a.Caption:=Capt_lab5a;
      label5b.Caption:='X5';
      label6a.Caption:=Capt_lab6a;
      label6b.Caption:='X6';
      labGx.Caption:='X7/X8';
      labGxdop.Visible:=false;
      labGy.Caption:='X9/X8';
      labGydop.Visible:=false;
      labTr.Caption:=' X10';
      labTrdop.Visible:=false;
    end;
    3:begin
      label1a.Caption:=Capt_lab1a;
      label1b.Caption:=Capt_lab1b;
      label2a.Caption:=Capt_lab2a;
      if SizeDirWind=1 then begin
        label2b.Caption:=Capt_lab2b;
        label2c.Visible:=true;
        end else begin
        label2b.Caption:=Capt_lab2b2;
        label2c.Visible:=false;
        end;
      label3a.Caption:=Capt_lab3a;
      label3b.Caption:='T,   C';
      label3c.Visible:=true;
      label4a.Caption:=Capt_lab4a;
      label4b.Caption:=Capt_lab4b;
      label4a.Visible:=false;
      label4b.Visible:=false;
      label5a.Caption:=Capt_lab5a;
      if SizeDavlen=1 then label5b.Caption:=Capt_lab5b
                 else label5b.Caption:=Capt_lab5b2;
      label6a.Caption:=Capt_lab6a;
      label6b.Caption:='r, %';
      LabGammas.Visible:=false;
      labGx.Visible:=false;
```

```
      labGxdop.Visible:=false;
      labGydop.Visible:=true;
      labGy.Visible:=false;
      labGydop.Visible:=false;
      labTr.Visible:=false;
      labTrdop.Visible:=false;
      LabHor1.Visible:=false;
      LabHor2.Visible:=false;
    end;
  end;
  MakeAllPaneles;
  UpdateColorEditFont;
end;

{*********************************}
procedure TMainForm.UpdateGraphics;
var
  Col1:TColor;
begin
  Col1:=GetColorFromIndex(WindowColor);
  Panel1b.Color:=Col1;
  PaintBox1.Color:=Col1;
  Panel2b.Color:=Col1;
  PaintBox2.Color:=Col1;
  Panel3b.Color:=Col1;
  PaintBox3.Color:=Col1;
  if RegimVabor.ItemIndex<>3 then begin
    Panel4b.Color:=Col1;
    PaintBox4.Color:=Col1;
    end else begin
    Panel4b.Color:=ClSilver;
    PaintBox4.Color:=ClSilver;
    end;
  Panel5b.Color:=Col1;
  PaintBox5.Color:=Col1;
  Panel6b.Color:=Col1;
  PaintBox6.Color:=Col1;
  UpdateColorEditFont;
end;

//*******************************************
procedure TMainForm.ClearAmplitGr;
var
  i:Byte;
begin
  for i:=0 to 8 do
    AmplitGr[i]:=-1.1e6;
end;

{*********************************}
procedure TMainForm.UpdateAmplitGr;
var
  i:Byte;
begin
 case RegimVabor.ItemIndex of
 0:begin
     if Met_Data<>nil then begin
     if BreakReadDataUGI then begin
```

```
      with MeteoData do begin
      AmplitGr[0]:=Sred.Vh;
      AmplitGr[1]:=Sred.Fh;
      AmplitGr[2]:=Sred.T;
      AmplitGr[3]:=Sred.w;
      AmplitGr[4]:=Sred.MMd;
      AmplitGr[5]:=Sred.Otvl;
      AmplitGr[6]:=Sred.Gx;
      AmplitGr[7]:=Sred.Gy;
      AmplitGr[8]:=Sred.Tr;
      end;
      end else begin
      if RegOutData.ItemIndex=0 then begin
      CalcHorWind(Met_SrVar[1],Met_SrVar[2],AmplitGr[0],AmplitGr[1]);
      AmplitGr[2]:=Met_SrVar[0];
      for i:=3 to 8 do AmplitGr[i]:=Met_SrVar[i];
      end else begin
      AmplitGr[0]:=Vhor;
      AmplitGr[1]:=Fhor;
      AmplitGr[2]:=TC;
      AmplitGr[3]:=Vz;
      AmplitGr[4]:=MMdavl;
      AmplitGr[5]:=Otvlag;
      AmplitGr[6]:=GammaX;
      AmplitGr[7]:=GammaY;
      AmplitGr[8]:=DatTC;
      end; {if RegOutData}
      end; {if BreakReadDataUGI}
      end else ClearAmplitGr;
      end;
    1:begin
       if Cal_Data<>nil then begin
      if BreakReadDataUGI then begin
        for i:=0 to 8 do AmplitGr[i]:=Cal_Sred[i];
        end else begin
        if RegOutData.ItemIndex=0 then begin
         for i:=0 to 8 do AmplitGr[i]:=Cal_SrVar[i];
         end else for i:=0 to 8 do AmplitGr[i]:=Cal_Data[NumPosGlobData].mas[i];
      end; {if BreakReadDataUGI}
      end else ClearAmplitGr;
     end;
    2:begin
      if Con_Data<>nil then begin
      if BreakReadDataUGI then begin
        for i:=0 to 8 do AmplitGr[i]:=Con_Sred[i];
        end else begin
        if RegOutData.ItemIndex=0 then begin
         for i:=0 to 8 do AmplitGr[i]:=Con_SrVar[i];
         end else for i:=0 to 8 do AmplitGr[i]:=Con_Data[NumPosGlobData].mas[i];
      end; {if not IsReadDataPPU}
      end else ClearAmplitGr;
     end;
    end;
   end;

   {****************************}
   procedure TMainForm.ClearEdit;
   begin
```

```
  EditCount.Text:='';
  Edit1.Text:='';
  Edit2.Text:='';
  Edit3.Text:='';
  Edit4.Text:='';
  Edit5.Text:='';
  Edit6.Text:='';
  EditGamX.Text:='';
  EditGamY.Text:='';
  EditTr.Text:='';
end;

{****************************}
procedure TMainForm.UpdateEdit;
begin
  if not (AmplitGr[0]>-1e5) then ClearEdit else
case RegimVabor.ItemIndex of
  0:begin
     Edit1.Text:=Format('%6.2f',[AmplitGr[0]]);
     if SizeDirWind=1 then Edit2.Text:=Format('%6.1f',[AmplitGr[1]])
                 else Edit2.Text:=GetStrDirWindDYg(AmplitGr[1]);
     Edit3.Text:=Format('%6.2f',[AmplitGr[2]]);
     Edit4.Text:=Format('%6.2f',[AmplitGr[3]]);
     if SizeDavlen=1 then Edit5.Text:=Format('%6.2f',[AmplitGr[4]])
                else Edit5.Text:=Format('%6.1f',[AmplitGr[4]/0.75]);
     Edit6.Text:=Format('%6.1f',[AmplitGr[5]]);
     EditGamX.Text:=Format('%6.1f',[AmplitGr[6]]);
     EditGamY.Text:=Format('%6.1f',[AmplitGr[7]]);
     EditTr.Text:=Format('%6.2f',[AmplitGr[8]]);
    end;
  1:begin
     Edit1.Text:=Format('%6.2f',[AmplitGr[0]]);
     Edit2.Text:=Format('%6.2f',[AmplitGr[1]]);
     Edit3.Text:=Format('%6.2f',[AmplitGr[2]]);
     Edit4.Text:=Format('%6.2f',[AmplitGr[3]]);
     if SizeDavlen=1 then Edit5.Text:=Format('%6.1f',[AmplitGr[4]])
                else Edit5.Text:=Format('%6.1f',[AmplitGr[4]/0.75]);
     Edit5.Text:=Format('%6.1f',[AmplitGr[4]]);
     Edit6.Text:=Format('%6.1f',[AmplitGr[5]]);
     EditGamX.Text:=Format('%6.1f',[AmplitGr[6]]);
     EditGamY.Text:=Format('%6.1f',[AmplitGr[7]]);
     EditTr.Text:=Format('%6.2f',[AmplitGr[8]]);
    end;
  2:begin
     Edit1.Text:=IntToStr(round(AmplitGr[0]));
     Edit2.Text:=IntToStr(round(AmplitGr[1]));
     Edit3.Text:=IntToStr(round(AmplitGr[2]));
     Edit4.Text:=IntToStr(round(AmplitGr[3]));
     Edit5.Text:=IntToStr(round(AmplitGr[4]));
     Edit6.Text:=IntToStr(round(AmplitGr[5]));
     EditGamX.Text:=Format('%8.4f',[AmplitGr[6]]);
     EditGamY.Text:=Format('%8.4f',[AmplitGr[7]]);
     EditTr.Text:=Format('%7.1f',[AmplitGr[8]]);
    end;
  3:begin
     Edit1.Text:=Format('%6.2f',[AmplitGr[0]]);
     if SizeDirWind=1 then Edit2.Text:=Format('%6.1f',[AmplitGr[1]])
                 else Edit2.Text:=GetStrDirWindDYg(AmplitGr[1]);
```

```
      Edit3.Text:=Format('%6.2f',[AmplitGr[2]]);
      Edit4.Text:='';
      if SizeDavlen=1 then Edit5.Text:=Format('%6.1f',[AmplitGr[4]])
                 else Edit5.Text:=Format('%6.1f',[AmplitGr[4]/0.75]);
      Edit6.Text:=Format('%6.1f',[AmplitGr[5]]);
      EditGamX.Text:='';
      EditGamY.Text:='';
      EditTr.Text:='';
    end;
  end;
   EditOrient.Text:=Format('%6.1f',[DeviceOrient]);
   if MinForm.Visible then MinForm.UpdateEdit;
end;

//*******************************************
procedure TMainForm.DrawAmplitGr_1(MyColor:TColor);
begin
  if AmplitGr[0]<=-1e5 then Exit;
  case RegimVabor.ItemIndex of
  0:if not ((AmplitGr[0]<Vh_a0-Vh_xd) or (AmplitGr[0]>Vh_a0+Vh_xd)) then
    DrawStrelkaAmpl(AmplitGr[0],Vh_a0,Vh_xd,MyColor,PaintBox1.Canvas,PaintBox1.Height,PaintBox1.Width);
  1:if not ((AmplitGr[0]<Si_a0-Si_xd) or (AmplitGr[0]>Si_a0+Si_xd)) then
    DrawStrelkaAmpl(AmplitGr[0],Si_a0,Si_xd,MyColor,PaintBox1.Canvas,PaintBox1.Height,PaintBox1.Width);
  2:if not ((AmplitGr[0]<ti_a0-ti_xd) or (AmplitGr[0]>ti_a0+ti_xd)) then
    DrawStrelkaAmpl(AmplitGr[0],ti_a0,ti_xd,MyColor,PaintBox1.Canvas,PaintBox1.Height,PaintBox1.Width);
  end;
end;

{****************************}
procedure TMainForm.DrawAmplitGr_2(MyColor:TColor);
begin
  if AmplitGr[1]<=-1e5 then Exit;
  case RegimVabor.ItemIndex of
  0:DrawStrelkaDirWind(AmplitGr[1],MyColor,PaintBox2.Canvas,PaintBox2.Height,PaintBox2.Width);
  1:if not ((AmplitGr[1]<Si_a0-Si_xd) or (AmplitGr[1]>Si_a0+Si_xd)) then
    DrawStrelkaAmpl(AmplitGr[1],Si_a0,Si_xd,MyColor,PaintBox2.Canvas,PaintBox2.Height,PaintBox2.Width);
  2:if not ((AmplitGr[1]<ti_a0-ti_xd) or (AmplitGr[1]>ti_a0+ti_xd)) then
    DrawStrelkaAmpl(AmplitGr[1],ti_a0,ti_xd,MyColor,PaintBox2.Canvas,PaintBox2.Height,PaintBox2.Width);
  end;
end;

{****************************}
procedure TMainForm.DrawAmplitGr_3(MyColor:TColor);
begin
  if AmplitGr[2]<=-1e5 then Exit;
  case RegimVabor.ItemIndex of
  0:if not ((AmplitGr[2]<TC_a0-TC_xd) or (AmplitGr[2]>TC_a0+TC_xd)) then
    DrawStrelkaAmpl(AmplitGr[2],TC_a0,TC_xd,MyColor,PaintBox3.Canvas,PaintBox3.Height,PaintBox3.Width);
  1:if not ((AmplitGr[2]<Si_a0-Si_xd) or (AmplitGr[2]>Si_a0+Si_xd)) then
    DrawStrelkaAmpl(AmplitGr[2],Si_a0,Si_xd,MyColor,PaintBox3.Canvas,PaintBox3.Height,PaintBox3.Width);
  2:if not ((AmplitGr[2]<ti_a0-ti_xd) or (AmplitGr[2]>ti_a0+ti_xd)) then
    DrawStrelkaAmpl(AmplitGr[2],ti_a0,ti_xd,MyColor,PaintBox3.Canvas,PaintBox3.Height,PaintBox3.Width);
  end;
end;

{****************************}
procedure TMainForm.DrawAmplitGr_4(MyColor:TColor);
begin
```

```
  if AmplitGr[3]<=-1e5 then Exit;
  case RegimVabor.ItemIndex of
  0:if not ((AmplitGr[3]<Vz_a0-Vz_xd) or (AmplitGr[3]>Vz_a0+Vz_xd)) then
    DrawStrelkaAmpl(AmplitGr[3],Vz_a0,Vz_xd,MyColor,PaintBox4.Canvas,PaintBox4.Height,PaintBox4.Width);
  1:if not ((AmplitGr[3]<Si_a0-Si_xd) or (AmplitGr[3]>Si_a0+Si_xd)) then
    DrawStrelkaAmpl(AmplitGr[3],Si_a0,Si_xd,MyColor,PaintBox4.Canvas,PaintBox4.Height,PaintBox4.Width);
  2:if not ((AmplitGr[3]<ti_a0-ti_xd) or (AmplitGr[3]>ti_a0+ti_xd)) then
    DrawStrelkaAmpl(AmplitGr[3],ti_a0,ti_xd,MyColor,PaintBox4.Canvas,PaintBox4.Height,PaintBox4.Width);
  end;
end;

{****************************}
procedure TMainForm.DrawAmplitGr_5(MyColor:TColor);
begin
  if AmplitGr[4]<=-1e5 then Exit;
  if RegimVabor.ItemIndex <> 2 then begin
  if SizeDavlen=1 then begin
  if not ((AmplitGr[4]<P_a0-P_xd) or (AmplitGr[4]>P_a0+P_xd)) then
  DrawStrelkaAmpl(AmplitGr[4],P_a0,P_xd,MyColor,PaintBox5.Canvas,PaintBox5.Height,PaintBox5.Width)
  end else begin
  if not ((AmplitGr[4]/0.75<P_a02-P_xd2) or (AmplitGr[4]/0.75>P_a02+P_xd2)) then
  DrawStrelkaAmpl(AmplitGr[4]/0.75,P_a02,P_xd2,MyColor,PaintBox5.Canvas,PaintBox5.Height,PaintBox5.Width)
  end;
  end else begin
  if not ((AmplitGr[4]<ACP_a0-ACP_xd) or (AmplitGr[4]>ACP_a0+ACP_xd)) then
  DrawStrelkaAmpl(AmplitGr[4],ACP_a0,ACP_xd,MyColor,PaintBox5.Canvas,PaintBox5.Height,PaintBox5.Width);
  end;
end;

{****************************}
procedure TMainForm.DrawAmplitGr_6(MyColor:TColor);
begin
  if AmplitGr[5]<=-1e5 then Exit;
  if RegimVabor.ItemIndex <> 2 then begin
  if not ((AmplitGr[5]<Hu_a0-Hu_xd) or (AmplitGr[5]>Hu_a0+Hu_xd)) then
  DrawStrelkaAmpl(AmplitGr[5],Hu_a0,Hu_xd,MyColor,PaintBox6.Canvas,PaintBox6.Height,PaintBox6.Width)
  end else begin
  if not ((AmplitGr[5]<ACPr_a0-ACPr_xd) or (AmplitGr[5]>ACPr_a0+ACPr_xd)) then
  DrawStrelkaAmpl(AmplitGr[5],ACPr_a0,ACPr_xd,MyColor,PaintBox6.Canvas,PaintBox6.Height,PaintBox6.Width);
  end;
end;

{****************************}
procedure TMainForm.DrawAmplitGrAll(MyColor:TColor);
begin
  DrawAmplitGr_1(MyColor);
  DrawAmplitGr_2(MyColor);
  DrawAmplitGr_3(MyColor);
  DrawAmplitGr_4(MyColor);
  DrawAmplitGr_5(MyColor);
  DrawAmplitGr_6(MyColor);
end;

{****************************************}
procedure TMainForm.FormPaint(Sender: TObject);
begin
  try
  FormResize(Sender);
```

```
  UpdateAmplitGr;
  UpdatePanel;
  UpdateViewBtnEnabled;
  UpdateEdit;
  PaintBox1Paint(Sender);
  PaintBox2Paint(Sender);
  PaintBox3Paint(Sender);
  PaintBox4Paint(Sender);
  PaintBox5Paint(Sender);
  PaintBox6Paint(Sender);
  except
  end;
end;

{*********************************************}
procedure TMainForm.PaintBox1Paint(Sender: TObject);
begin
  try
  case RegimVabor.ItemIndex of
  0:DrawWinAmpl(Vh_a0,Vh_ad,Vh_xd,PaintBox1.Canvas,PaintBox1.Height,PaintBox1.Width);
  1:DrawWinAmpl(Si_a0,Si_ad,Si_xd,PaintBox1.Canvas,PaintBox1.Height,PaintBox1.Width);
  2:DrawWinAmpl(ti_a0,ti_ad,ti_xd,PaintBox1.Canvas,PaintBox1.Height,PaintBox1.Width);
  end;
  DrawAmplitGr_1(GetColorFromIndex(ArrowColor));
  except
  end;
end;

{*********************************************}
procedure TMainForm.PaintBox2Paint(Sender: TObject);
begin
  try
  case RegimVabor.ItemIndex of
  0:DrawWinDirWind(PaintBox2.Canvas,PaintBox2.Height,PaintBox2.Width,false);
  1:DrawWinAmpl(Si_a0,Si_ad,Si_xd,PaintBox2.Canvas,PaintBox2.Height,PaintBox2.Width);
  2:DrawWinAmpl(ti_a0,ti_ad,ti_xd,PaintBox2.Canvas,PaintBox2.Height,PaintBox2.Width);
  end;
  DrawAmplitGr_2(GetColorFromIndex(ArrowColor));
  except
  end;
end;

{*********************************************}
procedure TMainForm.PaintBox3Paint(Sender: TObject);
begin
  try
  case RegimVabor.ItemIndex of
  0:DrawWinAmpl(TC_a0,TC_ad,TC_xd,PaintBox3.Canvas,PaintBox3.Height,PaintBox3.Width);
  1:DrawWinAmpl(Si_a0,Si_ad,Si_xd,PaintBox3.Canvas,PaintBox3.Height,PaintBox3.Width);
  2:DrawWinAmpl(ti_a0,ti_ad,ti_xd,PaintBox3.Canvas,PaintBox3.Height,PaintBox3.Width);
  end;
  DrawAmplitGr_3(GetColorFromIndex(ArrowColor));
  except
  end;
end;

{*********************************************}
procedure TMainForm.PaintBox4Paint(Sender: TObject);
```

```
begin
  try
  case RegimVabor.ItemIndex of
  0:DrawWinAmpl(Vz_a0,Vz_ad,Vz_xd,PaintBox4.Canvas,PaintBox4.Height,PaintBox4.Width);
  1:DrawWinAmpl(Si_a0,Si_ad,Si_xd,PaintBox4.Canvas,PaintBox4.Height,PaintBox4.Width);
  2:DrawWinAmpl(ti_a0,ti_ad,ti_xd,PaintBox4.Canvas,PaintBox4.Height,PaintBox4.Width);
  end;
  DrawAmplitGr_4(GetColorFromIndex(ArrowColor));
  except
  end;
end;

{*********************************************}
procedure TMainForm.PaintBox5Paint(Sender: TObject);
begin
  try
  if RegimVabor.ItemIndex<>2 then begin
  if SizeDavlen=1 then
  DrawWinAmpl(P_a0,P_ad,P_xd,PaintBox5.Canvas,PaintBox5.Height,PaintBox5.Width)
  else DrawWinAmpl(P_a02,P_ad2,P_xd2,PaintBox5.Canvas,PaintBox5.Height,PaintBox5.Width);
  end else
  DrawWinAmpl(ACP_a0,ACP_ad,ACP_xd,PaintBox5.Canvas,PaintBox5.Height,PaintBox5.Width);
  DrawAmplitGr_5(GetColorFromIndex(ArrowColor));
  except
  end;
end;

{*********************************************}
procedure TMainForm.PaintBox6Paint(Sender: TObject);
begin
  try
  if RegimVabor.ItemIndex<>2 then
  DrawWinAmpl(Hu_a0,Hu_ad,Hu_xd,PaintBox6.Canvas,PaintBox6.Height,PaintBox6.Width)
  else
  DrawWinAmpl(ACPr_a0,ACPr_ad,ACPr_xd,PaintBox6.Canvas,PaintBox6.Height,PaintBox6.Width);
  DrawAmplitGr_6(GetColorFromIndex(ArrowColor));
  except
  end;
end;

{*****************************************************}
procedure TMainForm.UpdateAllGraph(Sender: TObject);
begin
  UpdateGraphics;
  FormPaint(Sender);
  IzmerenForm.UpdateVidGraph(Sender);
  CalibUGIForm.UpdateVidGraph(Sender);
  ControlUGIForm.UpdateVidGraph(Sender);
end;

{*************************************}
procedure TMainForm.CloseAnalysis;
begin
  try
  ClearGlobData(Con_Data);
  ClearGlobData(Cal_Data);
  ClearGlobData(Met_Data);
  if ReportForm.Visible then ReportForm.Close;
```

```
  if (RegimVabor.ItemIndex<>0) and IzmerenForm.Visible then
    IzmerenForm.Close;
  if (RegimVabor.ItemIndex<>1) and CalibUGIForm.Visible then
    CalibUGIForm.Close;
  if (RegimVabor.ItemIndex<>2) and ControlUGIForm.Visible then
    ControlUGIForm.Close;
  Report.Enabled:=false;
  UpdateViewBtnEnabled;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.OpenDevice;
begin
  Izmeren.Enabled:=true;
  Calibrate.Enabled:=true;
  Control.Enabled:=true;
  StartBtn.Enabled:=true;
  StopBtn.Enabled:=false;
  StatusBar0.SimpleText:='';
  EditDevice.Color:=ClNavy;
  EditDevice.Text:=Msg_OtklStatus;
  Lab_UGI1.Caption:=Msg_Chtenie;
  Lab_UGI2.Caption:=Msg_Chtenie1;
  EditCount.Color:=ClNavy;
  EditUsredn.Color:=ClNavy;
  Data_Ugi[0]:=0;
end;

{*********************************}
procedure TMainForm.CloseDevice;
begin
  Izmeren.Enabled:=false;
  Calibrate.Enabled:=false;
  Control.Enabled:=false;
  StopBtn.Enabled:=false;
  StartBtn.Enabled:=false;
end;

{*********************************}
procedure TMainForm.OpenMenu;
begin
  OpenDevice;
  if IsAvtoLoadProgram then begin
   IsAvtoLoadProgram:=false;
   ChangeRegistrWindows;
   WriteIniFile;
   end;
  Options.Enabled:=true;
  GetConstants.Enabled:=true;
  GetTestPPU.Enabled:=true;
  RegimVabor.Enabled:=true;
  if ValidGlobData(Met_Data)>=60 then Report.Enabled:=true;
end;

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

```
procedure TMainForm.CloseMenu;
begin
  CloseDevice;
  StopBtn.Enabled:=true;
  Options.Enabled:=false;
  GetConstants.Enabled:=false;
  GetTestPPU.Enabled:=false;
  Report.Enabled:=false;
  RegimVabor.Enabled:=false;
end;

{*********************************}
procedure TMainForm.StartBtnClick(Sender: TObject);
begin
  case RegimVabor.ItemIndex of
  0:IzmerenExecute(Sender);
  1:CalibrateExecute(Sender);
  2:ControlExecute(Sender);
  end;
end;

{*********************************}
procedure TMainForm.StopBtnClick(Sender: TObject);
begin
  StopReadData;
  if not IsModem then SetPPU_inWork(Handle_AMK,OverLap_AMK);
  Close_AMKport(Handle_AMK,IsOpenCOMport);
  OpenMenu;
end;

//*****************************************
procedure TMainForm.OutAllDataToMainForm;
begin
  try
  DrawAmplitGrAll(GetColorFromIndex(DiagColor));
  UpdateAmplitGr;
  DrawAmplitGrAll(GetColorFromIndex(ArrowColor));
  UpdateEdit;
  if MinForm.Visible then MinForm.UpdateEdit;
  except
  raise
  end;
end;

{*************************************}
procedure TMainForm.ControlDataMainForm;
const
  RepOut = 5;
begin
  try
  case  RegOutData.ItemIndex of
  0:if Now > ChekTimeOutSred then Begin
      OutAllDataToMainForm;
      ChekTimeOutSred:=Now+EncodeTime(0,0,RepOut,0);
      End;
  1:OutAllDataToMainForm;
    end;
  except
```

```
  raise
  end;
end;

{*************************************************************************}
procedure TMainForm.SaveDataFile;
var
  Data:TStringList;
  FileName1:String;
begin
  if not IsSaveDataFile then Exit;
  if ValidGlobData(Met_Data)<60 then Exit;
  try
  try
  Screen.Cursor:=crHourGlass;
  FileName1:=GetNameFileDef(Met_Data[NumPosGlobData].TDr);
  FileName1:=Format('%s%s%s%s',[DataDir,'\',FileName1,'D']);
  Data:=TStringList.Create;
  if MeteoData_GetDataStrList(Data) then
  Data.SaveToFile(FileName1);
  except
  raise;
  end;
  finally
  Data.Free;
  Screen.Cursor:=crDefault;
  end;
end;

{*************************************************************************}
procedure TMainForm.SaveReportFile;
var
  Report:TStringList;
  FileName1:String;
begin
  if not IsSaveReportFile then Exit;
  if ValidGlobData(Met_Data)<60 then Exit;
  try
  try
  Screen.Cursor:=crHourGlass;
  FileName1:=GetNameFileDef(Met_Data[NumPosGlobData].TDr);
  FileName1:=Format('%s%s%s%s',[DataDir,'\',FileName1,'R']);
  Report:=TStringList.Create;
  if MeteoData_GetReportStrList(Report) then begin
  Report.SaveToFile(FileName1);
  end;
  except
  raise;
  end;
  finally
  Report.Free;
  Screen.Cursor:=crDefault;
  end;
end;

{*************************************************************************}
procedure TMainForm.SaveArrStrFile;
const
```

```
st01='   Date    Time  <t>     T     V  Vmin  Vmax     D     w     P';
st02='     r RMS(T) RMS(V) RMS(D) RMS(w) RMS(u) RMS(v)        Ct2        Cv2';
st03='       Cn2o       Cn2a         Tau          H          T*         u*          L';
var
  Rep1:TStringList;
  St,st1:String;
  k0:Integer;
  Hour1,Min1,Sec1,MSec1:Word;
  per,per2:Real;
begin
  if not IsSaveArrStrFile then Exit;
  if ValidGlobData(Met_Data)<60 then Exit;
  try
  try
  Screen.Cursor:=crHourGlass;
  Rep1:=TStringList.Create;
  if not FileExists(FileNameArrStr) then Rep1.Add(Format('%s%s%s',[st01,st02,st03]))
  else Rep1.LoadFromFile(FileNameArrStr);
  st:='';
  st:=st+DateTo_MyStr(Met_Data[NumPosGlobData].TDr)+' ';
  st:=st+TimeTo_MyStr(Met_Data[NumPosGlobData].TDr);
  if NumPosGlobData=High(Met_Data) then k0:=0
                          else k0:=NumPosGlobData+1;
  DecodeTime(Met_Data[NumPosGlobData].TDr-Met_Data[k0].TDr,Hour1,Min1,Sec1,MSec1);
  per:=(3600*Hour1+60*Min1+Sec1+MSec1/1000)/60;
  str(per:6:2,st1);
  st:=st+st1;
  With MeteoData do Begin
  str(Sred.T:7:2,st1);
  st:=st+st1;
  str(Sred.Vh:7:2,st1);
  st:=st+st1;
  str(VminSrok:7:2,st1);
  st:=st+st1;
  str(VmaxSrok:7:2,st1);
  st:=st+st1;
  str(Sred.Fh:7:1,st1);
  st:=st+st1;
  str(Sred.w:7:2,st1);
  st:=st+st1;
  str(Sred.MMd:7:1,st1);
  st:=st+st1;
  str(Sred.Otvl:7:1,st1);
  st:=st+st1;
  str(SKO.T:7:2,st1);
  st:=st+st1;
  str(SKO.Vh:7:2,st1);
  st:=st+st1;
  str(SKO.Fh:7:2,st1);
  st:=st+st1;
  str(SKO.w:7:2,st1);
  st:=st+st1;
  str(SKO.u:7:2,st1);
  st:=st+st1;
  str(SKO.v:7:2,st1);
  st:=st+st1;
  per2:=exp(-4./3*ln(10));
  per:=per2*Turb.Ct2;
```

```
    str(per:12,st1);
    st:=st+st1;
    per:=per2*Turb.Cv2;
    str(per:12,st1);
    st:=st+st1;
    per:=Turb.Cn2opt*per2;
    str(per:12,st1);
    st:=st+st1;
    str(Turb.Cn2:12,st1);
    st:=st+st1;
    str(Turb.Taux:12,st1);
    st:=st+' '+st1;
    str(Turb.H:12,st1);
    st:=st+' '+st1;
    str(Turb.Tmo:12,st1);
    st:=st+' '+st1;
    str(Turb.Vmo:12,st1);
    st:=st+st1;
    str(Turb.Lmo:12,st1);
    st:=st+' '+st1;
    End;
    Rep1.Add(st);
    Rep1.SaveToFile(FileNameArrStr);
    Rep1.Free;
    except
    raise;
    end;
    finally
    Screen.Cursor:=crDefault;
    end;
  end;

  {*************************************************************************}
  procedure TMainForm.SaveInfoFile;
  var
    Info:TStringList;
    FileName1,MyStr:String;
  begin
    if not IsSaveInfoFile then Exit;
    if ValidGlobData(Met_Data)<60 then Exit;
    try
    try
    Screen.Cursor:=crHourGlass;
    FileName1:=GetNameFileDef(Met_Data[NumPosGlobData].TDr);
    FileName1:=ChangeFileExt(FileName1,'.TXT');
    FileName1:=Format('%s%s%s',[DataDir,'\',FileName1]);
    info:=TStringList.Create;
    if GetMeteoInformStr(MyStr) then begin
    Info.Add(MyStr);
    Info.SaveToFile(FileName1);
    Info.Free;
    end;
    except
    raise;
    end;
    finally
    Screen.Cursor:=crDefault;
    end;
```

```
end;

{*****************************************************************************}
procedure TMainForm.SaveDataBase;
label 1;
var
  Hour1,Min1,Sec1,MSec1:Word;
  per,res:Real;
  st:String;
  d1,t1:TDateTime;
  k0:Integer;
  F1:Boolean;
begin
  if not IsSaveBazaData then Exit;
  if ValidGlobData(Met_Data)<60 then Exit;
  try
1:F1:=OpenMyDataBase;
  if not F1 then GOTO 1;
  DataTable.Append;
  st:=DateTo_MyStr(Met_Data[NumPosGlobData].TDr);
  MyStr_ToDate(st,d1);
  DataTableDateRec.AsDateTime:=d1;
  st:=TimeTo_MyStr(Met_Data[NumPosGlobData].TDr);
  MyStr_ToTime(st,t1);
  DataTableTimeRec.AsDateTime:=t1;
  if NumPosGlobData=High(Met_Data) then k0:=0
                          else k0:=NumPosGlobData+1;
  DecodeTime(Met_Data[NumPosGlobData].TDr-Met_Data[k0].TDr,Hour1,Min1,Sec1,MSec1);
  res:=(3600*Hour1+60*Min1+Sec1+MSec1/1000)/60;
  if res<0.001 then res:=0.001;
  if res>20.4 then res:=20.4;
  DataTableTimeAver.AsFloat:=res;
  With MeteoData do Begin
  res:=Sred.T;
  if res<-69.99 then res:=-69.99;
  if res>69.99 then res:=69.99;
  DataTableTemperat.AsFloat:=res;
  res:=Sred.Vh;
  if res<0 then res:=0.;
  if res>99.99 then res:=99.99;
  DataTableHorWind.AsFloat:=res;
  res:=VminSrok;
  if res<0 then res:=0.;
  if res>99.99 then res:=99.99;
  DataTableVmin.AsFloat:=res;
  res:=VmaxSrok;
  if res<0 then res:=0.;
  if res>99.99 then res:=99.99;
  DataTableVmax.AsFloat:=res;
  res:=Sred.Fh;
  if res<0 then res:=0;
  if res>359.999 then res:=359.999;
  DataTableDirWind.AsFloat:=res;
  res:=Sred.MMd;
  if res<500.001 then res:=500.001;
  if res>799.99 then res:=799.99;
  DataTablePressureRec.AsFloat:=res;
  res:=Sred.Otvl;
```

```
if res<0.001 then res:=0.001;
if res>129.99 then res:=129.99;
DataTableRelHuRec.AsFloat:=res;
res:=Sred.w;
if res<-19.99 then res:=-19.99;
if res>19.99 then res:=19.99;
DataTableVertWind.AsFloat:=res;
res:=SKO.T;
if res<0 then res:=0;
if res>70 then res:=70;
DataTableSko_T.AsFloat:=res;
res:=SKO.Vh;
if res<0 then res:=0.;
if res>99.99 then res:=99.99;
DataTableSko_V.AsFloat:=res;
res:=SKO.Fh;
if res<0 then res:=0;
if res>360 then res:=360;
DataTableSko_F.AsFloat:=res;
res:=SKO.w;
if res<0 then res:=0;
if res>19.99 then res:=19.99;
DataTableSko_w.AsFloat:=res;
res:=SKO.u;
if res<0 then res:=0.;
if res>99.99 then res:=99.99;
DataTableSko_ux.AsFloat:=res;
res:=SKO.v;
if res<0 then res:=0.;
if res>99.99 then res:=99.99;
DataTableSko_vy.AsFloat:=res;
res:=Asim.T;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
DataTableAss_T.AsFloat:=res;
res:=Asim.Vh;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
DataTableAss_V.AsFloat:=res;
res:=Asim.Fh;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
DataTableAss_F.AsFloat:=res;
res:=Asim.w;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
DataTableAss_w.AsFloat:=res;
res:=Asim.u;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
DataTableAss_ux.AsFloat:=res;
res:=Asim.v;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
DataTableAss_vy.AsFloat:=res;
res:=Exc.T;
if res<-99.99 then res:=-99.99;
if res>99.99 then res:=99.99;
```

```
    DataTableExc_T.AsFloat:=res;
    res:=Exc.Vh;
    if res<-99.99 then res:=-99.99;
    if res>99.99 then res:=99.99;
    DataTableExc_V.AsFloat:=res;
    res:=Exc.Fh;
    if res<-99.99 then res:=-99.99;
    if res>99.99 then res:=99.99;
    DataTableExc_F.AsFloat:=res;
    res:=Exc.w;
    if res<-99.99 then res:=-99.99;
    if res>99.99 then res:=99.99;
    DataTableExc_w.AsFloat:=res;
    res:=Exc.u;
    if res<-99.99 then res:=-99.99;
    if res>99.99 then res:=99.99;
    DataTableExc_ux.AsFloat:=res;
    res:=Exc.v;
    if res<-99.99 then res:=-99.99;
    if res>99.99 then res:=99.99;
    DataTableExc_vy.AsFloat:=res;
    DataTableR_Tw.AsFloat:=KorM.T_w;
    DataTableR_TV.AsFloat:=KorM.T_Vh;
    DataTableR_Tux.AsFloat:=KorM.T_u;
    DataTableR_Tvy.AsFloat:=KorM.T_v;
    DataTableR_Vw.AsFloat:=KorM.w_Vh;
    DataTableR_uxw.AsFloat:=KorM.u_w;
    DataTableR_vyw.AsFloat:=KorM.v_w;
    DataTableR_uxvy.AsFloat:=KorM.u_v;
    per:=exp(-4./3*ln(10));
    DataTableT_Ct2.AsFloat:=Turb.Ct2*per;
    DataTableT_Cv2.AsFloat:=Turb.Cv2*per;
    End; {With MeteoData}
    finally
    ModifDataBase:=true;
    DataTable.Post;
    DataTable.Close;
    end;
  end;

  {*********************************}
  procedure TMainForm.ControlSaveFiles;
  begin
    if (RegimIzmer=0) and (Now>=CurStartSave) then begin
      SetNewCurStartSave;
      if ValidGlobData(Met_Data)>=60 then begin
      if MeteoData_CalcStatistic then begin
        SaveInfoFile;
        SaveDataBase;
        SaveReportFile;
        SaveDataFile;
        SaveArrStrFile;
        end;
      end;
      MeteoData.Trmin:=100;
      MeteoData.Trmax:=-100;
      MeteoData.Tmin:=100;
      MeteoData.Tmax:=-100;
```

```
      MeteoData.Vmin:=100;
      MeteoData.Vmax:=-100;
     end;

   if IsSaveBinFile and (Now>=CheckTimeSaveBinF) then begin
     WriteFromBufToBinF;
     CheckTimeSaveBinF:=Now+EncodeTime(0,0,30,0);
     if Now-CheckTimeCreateBinF > EncodeTime(0,TimeUsredn div 60,0,0) then begin
       Close_BinF;
       GetNewBinFile;
       end;
     end;
  end;

  {*************************************}
  procedure TMainForm.SaveDataForDDE;
  var
    Data:TStringList;
    i:Integer;
  begin
    if ValidGlobData(Met_Data)<60 then Exit;
    try
    try
    Data:=TStringList.Create;
    if MeteoData_GetDataStrList(Data) then Begin
      AMKDataDdeItem.Lines.Clear;
      for i:=0 to Data.Count-1 do
       AMKDataDdeItem.Lines.Add(Data.Strings[i]);
      End;
      except
    raise;
    end;
    finally
    Data.Free;
    end;
  end;

  {*************************************}
  procedure TMainForm.SaveReportForDDE;
  var
    Data:TStringList;
    i:Integer;
  begin
    if ValidGlobData(Met_Data)<60 then Exit;
    try
    try
    Data:=TStringList.Create;
    if MeteoData_GetReportStrList(Data) then Begin
      AMKReportDdeItem.Lines.Clear;
      for i:=0 to Data.Count-1 do
       AMKReportDdeItem.Lines.Add(Data.Strings[i]);
      End;
      except
    raise;
    end;
    finally
    Data.Free;
    end;
```

```
end;

{*************************************}
procedure TMainForm.MakeMsgToDataDDE(const Msg:String);
begin
  AMKDataDdeItem.Lines.Clear;
  AMKDataDdeItem.Lines.Add(Msg);
end;

{*************************************}
procedure TMainForm.MakeMsgToDataStrDDE(const Msg:String);
begin
  AMKDataStrDdeItem.Lines.Clear;
  AMKDataStrDdeItem.Lines.Add(Msg);
end;

{*************************************}
procedure TMainForm.MakeMsgToReportDDE(const Msg:String);
begin
  AMKReportDdeItem.Lines.Clear;
  AMKReportDdeItem.Lines.Add(Msg);
end;

//***********************************
procedure TMainForm.UpdateDataStrDDE(Tim1:TDateTime; err1:Byte);
const
  Cherta = '|';
  Probel = ' ';
var
  st,st1,st2:String;
  hour1,min1,sec1,msec1,y1,m1,d1:Word;
begin
  try
  DecodeDate(Tim1,y1,m1,d1);
  st:=IntTostr(d1);
  if d1<10 then st:='0'+st;
  st2:=IntTostr(m1);
  if m1<10 then st2:='0'+st2;
  st:=Format('%s%s%s',[st,'.',st2]);
  st1:=IntTostr(y1);
  st2:=Copy(st1,3,2);
  st:=Format('%s%s%s',[st,'.',st2]);
  DecodeTime(Tim1,hour1,min1,sec1,msec1);
  st1:=IntTostr(hour1);
  if hour1<10 then st1:='0'+st1;
  st2:=IntTostr(min1);
  if min1<10 then st2:='0'+st2;
  st1:=Format('%s%s%s',[st1,':',st2]);
  st2:=IntTostr(sec1);
  if sec1<10 then st2:='0'+st2;
  st1:=Format('%s%s%s',[st1,':',st2]);
  st2:=IntTostr(msec1);
  if msec1<100 then st2:='0'+st2;
  if msec1<10 then st2:='0'+st2;
  st1:=Format('%s%s%s',[st1,':',st2]);
  st:=Format('%s%s%s%s',[st,Probel,st1,Cherta]);
  st1:=Format('%5d',[round(TC*100)]);
  st:=Format('%s%s%s',[st,st1,Cherta]);
```

```
  st1:=Format('%5d',[round(Vhor*100)]);
  st:=Format('%s%s%s',[st,st1,Cherta]);
  st1:=Format('%5d',[round(Fhor*100)]);
  st:=Format('%s%s%s',[st,st1,Cherta]);
  st1:=Format('%5d',[round(Vz*100)]);
  st:=Format('%s%s%s',[st,st1,Cherta]);
  st1:=Format('%5d',[round(MMdavl*100)]);
  st:=Format('%s%s%s',[st,st1,Cherta]);
  st1:=Format('%5d',[round(OtVlag*100)]);
  st:=Format('%s%s%s',[st,st1,Cherta]);
  st1:=Format('%1d',[err1]);
  st:=Format('%s%s',[st,st1]);
  MakeMsgToDataStrDDE(st);
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.ControlUpdateDDE;
const
  RepDDE = 1;
begin
  try
  if Now >= CheckTimeDDE then Begin
  if EditDevice.Text = Msg_NoData then begin
    MakeMsgToDataDDE(MsgForDDE_e);
    MakeMsgToDataStrDDE(MsgForDDE_e);
    MakeMsgToReportDDE(MsgForDDE_e);
    CheckTimeDDE:=Now+EncodeTime(0,RepDDE,0,0);
    Exit
    end;
  if (EditDevice.Text=Msg_Sboj) or (EditDevice.Text=Msg_FormatData) then begin
    MakeMsgToDataDDE(MsgForDDE_s);
    MakeMsgToReportDDE(MsgForDDE_s);
    CheckTimeDDE:=Now+EncodeTime(0,RepDDE,0,0);
    Exit
    end;
  if ValidGlobData(Met_Data)>=60 then begin
  if MeteoData_CalcStatistic then begin
    SaveDataForDDE;
    SaveReportForDDE;
    end;
  end;
  CheckTimeDDE:=Now+EncodeTime(0,RepDDE,0,0);
  End; //if Now >= CheckTimeDDE
  except
  raise
  end;
end;

//*****************************************
procedure TMainForm.MyDelay(Durat1:Real);
var
  Start1,Stop1:Real;
begin
  Start1:=GetTickCount/1000;
  Stop1:=Start1;
```

```
  while Stop1 < (Start1+Durat1) do
  Stop1:=GetTickCount/1000;
end;

{*********************************}
procedure TMainForm.GetStartStopTime;
var
  Hour1,Min1,Sec1:Word;
begin
  try
  Hour1:=TimeUsredn div 3600;
  Min1:=(TimeUsredn - 60*Hour1) div 60;
  Sec1:=TimeUsredn - 3600*Hour1 - 60*Min1;
  StartTime:=Now;
  StopTime:=StartTime+EncodeTime(Hour1,Min1,Sec1,0);
  CheckTimeDDE:=StartTime+EncodeTime(0,0,3,0);
  except
  raise
  end;
end;

{*********************************}
function TMainForm.UpdateTimeCikle(var Data1:TGlobData):Real;
var
  Hour1,Min1,Sec1,MSec1:Word;
  L1,k0:Integer;
begin
  try
  Result:=TimeCikle;
  if Data1=nil then Exit;
  L1:=Length(Data1);
  if L1>590 then begin
    if NumPosGlobData=High(Data1) then k0:=0
                          else k0:=NumPosGlobData+1;
    DecodeTime(Data1[NumPosGlobData].TDr-Data1[k0].TDr,Hour1,Min1,Sec1,MSec1);
    Result:=(3600*Hour1+60*Min1+Sec1+MSec1/1000)/(L1-1);
    end;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.SetFirstCurStartSave;
var
  beg:TDateTime;
  Hour1,Min1,dt:Word;
begin
  Min1:=BegMinAutSave;
  Hour1:=BegHourAutSave;
  dt:=TimeUsredn div 60;
  if RegimIzmer=1 then begin
   if Min1>=dt then Min1:=Min1-dt
            else begin
            Hour1:=Hour1-1;
            Min1:=60-dt+Min1;
            end;
   end; {if RegimIzmer=1}
```

```
  beg:=EncodeTime(Hour1,Min1,0,0);
  if Beg>=SysUtils.Time then CurStartSave:=Date+Beg
                  else CurStartSave:=Date+1+Beg;
  if IsModem then begin
    if Min1>=3 then Min1:=Min1-3
            else begin
            Hour1:=Hour1-1;
            Min1:=57+Min1;
            end;
    beg:=EncodeTime(Hour1,Min1,0,0);
    if Beg>=SysUtils.Time then CurStartRing:=Date+Beg
                   else CurStartRing:=Date+1+Beg;
    end; {if IsModem}
end;

{*********************************}
procedure TMainForm.SetNewCurStartSave;
begin
  CurStartSave:=CurStartSave+EncodeTime(InterHourAutSave,InterMinAutSave,0,0);
  if IsModem then
  CurStartRing:=CurStartRing+EncodeTime(InterHourAutSave,InterMinAutSave,0,0);
end;

{************************************}
procedure TMainForm.UpdateStatusWork;
var
  Flag:Boolean;
  st1:String;
  dt1:TDateTime;
  k0:Integer;
begin
  flag:=false;
  EditTime.Text:='';
  EditCount.Text:='';
  EditUsredn.Text:='';
  case RegimVabor.ItemIndex of
  0:if Met_Data<>nil then Begin
    EditCount.Text:=IntToStr(Length(Met_Data));
    EditTime.Text:=TimeTo_MyStr(Met_Data[NumPosGlobData].TDr);
    if Length(Met_Data)>1 then begin
    if NumPosGlobData=High(Met_Data) then k0:=0
                        else k0:=NumPosGlobData+1;
    dt1:=Met_Data[NumPosGlobData].TDr-Met_Data[k0].TDr;
    DateTimeToString(st1,'n:ss',dt1);
    EditUsredn.Text:=st1;
    if dt1>=EncodeTime(0,TimeUsredn div 60,0,0)-EncodeTime(0,0,1,0) then flag:=true;
    end; End;
  1:if Cal_Data<>nil then Begin
    EditCount.Text:=IntToStr(Length(Cal_Data));
    EditTime.Text:=TimeTo_MyStr(Cal_Data[NumPosGlobData].TDr);
    if Length(Cal_Data)>1 then begin
    if NumPosGlobData=High(Cal_Data) then k0:=0
                        else k0:=NumPosGlobData+1;
    dt1:=Cal_Data[NumPosGlobData].TDr-Cal_Data[k0].TDr;
    DateTimeToString(st1,'n:ss',dt1);
    EditUsredn.Text:=st1;
    if dt1>=EncodeTime(0,TimeUsredn div 60,0,0)-EncodeTime(0,0,1,0) then flag:=true;
    end; End;
```

```
    2:if Con_Data<>nil then Begin
      EditCount.Text:=IntToStr(Length(Con_Data));
      EditTime.Text:=TimeTo_MyStr(Con_Data[NumPosGlobData].TDr);
      if Length(Con_Data)>1 then begin
      if NumPosGlobData=High(Con_Data) then k0:=0
                                 else k0:=NumPosGlobData+1;
      dt1:=Con_Data[NumPosGlobData].TDr-Con_Data[k0].TDr;
      DateTimeToString(st1,'n:ss',dt1);
      EditUsredn.Text:=st1;
      if dt1>=EncodeTime(0,TimeUsredn div 60,0,0)-EncodeTime(0,0,1,0) then flag:=true;
      end; End;
    end; {case}
    if not flag then begin
      EditCount.Color:=ClNavy;
      EditUsredn.Color:=ClNavy;
      end else begin
      EditCount.Color:=ClGreen;
      EditUsredn.Color:=ClGreen;
      if (Met_Data<>nil) then
      if (Length(Met_Data)>3) then
      Report.Enabled:=true;
      end;
    if MinForm<>nil then
    if MinForm.Visible then MinForm.UpdateStatusWork;
  end;

  {*********************************}
  procedure TMainForm.ControlWorkDevice;
  begin
    case Data_Ugi[0] of
    9:begin
        EditDevice.Color:=ClMaroon;
        EditDevice.Text:=Msg_NoData;
      end;
   10:begin
        EditDevice.Color:=ClMaroon;
        EditDevice.Text:=Msg_FormatData;
      end;
   11:begin
        EditDevice.Color:=ClBlack;
        EditDevice.Text:=Msg_Zaderqka;
      end;
      else begin
        if Data_UGI[0]<4 then Begin
       EditDevice.Color:=ClGreen;
       EditDevice.Text:=Msg_Normal;
       End else Begin
       EditDevice.Color:=ClTeal;
       EditDevice.Text:=Msg_Sboj;
       End;
      end;
    end;
  end;

  {*********************************}
  procedure TMainForm.ControlWorkModems;
  begin
    if not IsPhoneForm then Exit;
```

```
  if (Data_Ugi[0]=9) or (not IsConnectPhone) then begin
    PhoneForm.EditConnect.Color:=ClNavy;
    PhoneForm.EditConnect.Text:=MsgModems_12;
    end else begin
    PhoneForm.EditConnect.Color:=ClGreen;
    PhoneForm.EditConnect.Text:=MsgModems_11;
    end;
  if (Data_Ugi[0]=9) and (not IsSbojPhone) and IsConnectPhone then Begin
    IsSbojPhone:=true;
    Data_Ugi[0]:=0;
    PhoneForm.Close;
    End;
end;

{*********************************}
procedure TMainForm.ControlAutomatic(Sender: TObject);
begin
  if Now>=CurStartSave then Begin
    Lab_UGI1.Caption:=Msg_Chtenie;
    Lab_UGI2.Caption:=Msg_Chtenie1;
    if isModem then begin
      if BreakReadDataUGI and IsConnectPhone then begin
        UpdateStartIzmeren(Sender);
        StartReadDataUgi;
        end;
      end else begin
      if BreakReadDataUGI then begin
        UpdateStartIzmeren(Sender);
        StartReadDataUgi;
        end;
      end;
    end else begin
    Lab_UGI1.Caption:=Msg_AvtoZapusk;
    Lab_UGI2.Caption:=Msg_IntTime;
    EditDevice.Text:=TimeToStr(Now-CurStartSave);
    EditDevice.Color:=ClGreen;
    End; {if Now>=CurStartPilot}
 if isModem and (Now>=CurStartRing) then Begin
    if not isPhoneForm then begin
      OpenPhoneForm;
      StopBtn.Enabled:=true;
      end else begin
      if not (IsRingPhone or IsConnectPhone) then
      PhoneForm.UpTrubBtnClick(Sender);
      end;
    End;
end;

{*********************************}
procedure TMainForm.RestoreModems(Sender: TObject);
begin
  try
  if IsSbojPhone and (not IsRingPhone) and (Now>=CheckTimeRestModems) then Begin
    if not isPhoneForm then OpenPhoneForm;
    PhoneForm.UpTrubBtnClick(Sender);
    End;
  except
  raise
```

```
  end;
end;

//*****************************************
procedure TMainForm.UpdateStartControl(Sender:TObject);
begin
  try
  CountInData:=0;
  RegimVabor.ItemIndex:=2;
  CloseAnalysis;
  FormPaint(Sender);
  if ControlUGIForm.Visible then begin
    ControlUGIForm.UpdateVidGraph(Sender);
    ControlUGIForm.BringToFront;
    end;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.MakeCikleControl(Sender:TObject);
var
  k:Integer;
  Mas1:TDataRecord;
begin
try
  if IsFiltrInput then FiltrateInputDataUgi;
  for k:=0 to 3 do
    Mas1[k]:=InputData[k+1];
  Mas1[4]:=InputData[5];
  Mas1[5]:=InputData[6];
  Mas1[6]:=InputData[7]/InputData[8];
  Mas1[7]:=InputData[9]/InputData[8];
  Mas1[8]:=InputData[10];
  AddRecGlobData(Mas1,Data_Ugi[0],Now,TimeUsredn,Con_Data,Con_SrVar);
  if ControlUGIForm.Visible then with ControlUGIForm do begin
    if Graf_Data=nil then ControlUGIForm.FormPaint(Sender)
                 else DrawLines(High(Graf_Data));
  end;
except
raise
end;
end;

{*********************************}
procedure TMainForm.UpdateStopControl(Sender: TObject);
begin
  try
  try
  Screen.Cursor:=crHourGlass;
  if not IsModem then
  TimeCikle:=UpdateTimeCikle(Con_Data);
  CalcStopControl;
  FormPaint(Sender);
  if ControlUGIForm.Visible then ControlUGIForm.FormPaint(Sender);
  finally
  Screen.Cursor:=crDefault;
```

```
  end;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.UpdateStartCalibr(Sender: TObject);
begin
  try
  CountInData:=0;
  RegimVabor.ItemIndex:=1;
  CloseAnalysis;
  FormPaint(Sender);
  if CalibUGIForm.Visible then begin
    CalibUGIForm.UpdateVidGraph(Sender);
    CalibUGIForm.BringToFront;
    end;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.MakeCikleCalibr(Sender:TObject);
var
  k:Integer;
  Mas1:TDataRecord;
begin
try
  if IsFiltrInput then FiltrateInputDataUgi;
  if Data_Ugi[0]<4 then CalcCikleCalibr;
  Mas1[4]:=MMdavl_C; Mas1[5]:=Otvlag_C;
  Mas1[6]:=GammaX; Mas1[7]:=GammaY; Mas1[8]:=DatTC;
  for k:=0 to 3 do
    Mas1[k]:=Smm_C[k];
  AddRecGlobData(Mas1,Data_Ugi[0],Now,TimeUsredn,Cal_Data,Cal_SrVar);
  if CalibUGIForm.Visible then with CalibUGIForm do begin
    if Graf_Data=nil then CalibUGIForm.FormPaint(Sender)
                 else DrawLines(High(Graf_Data));
   end;
except
raise
end;
end;

{*********************************}
procedure TMainForm.UpdateStopCalibr(Sender: TObject);
var
  W:Integer;
  i:Byte;
begin
try
  if not IsModem then TimeCikle:=UpdateTimeCikle(Cal_Data);
  CalcStopCalibr;
  FormPaint(Sender);
  if CalibUGIForm.Visible then CalibUGIForm.FormPaint(Sender);
  if ValidGlobData(Cal_Data)<60 then begin
```

```
      GetMyErrorDlg(Msg_NoSaveGraduir);
      Exit;
      end;
    W:=GetMyConfirm2Dlg(Msg_SaveGraduir);
    if W = IdYes then Begin
      Tcalibr:=TkalibrNew;
      for i:=0 to 3 do
        RastMM[i]:=Smm_C[i];
      End;
  except
  raise
  end;
  end;

  {*********************************}
  procedure TMainForm.UpdateStartIzmeren(Sender: TObject);
  begin
    try
    CountInData:=0;
    RegimVabor.ItemIndex:=0;
    MeteoData.Trmin:=100;
    MeteoData.Trmax:=-100;
    MeteoData.Tmin:=100;
    MeteoData.Tmax:=-100;
    MeteoData.Vmin:=100;
    MeteoData.Vmax:=-100;
    CloseAnalysis;
    FormPaint(Sender);
    if IzmerenForm.Visible then IzmerenForm.FormPaint(Sender);
    if IzmerenForm.Visible then begin
      IzmerenForm.UpdateVidGraph(Sender);
      IzmerenForm.BringToFront;
      end;
    if IsTehnologyDDE then begin
      if Data_Ugi[0]<4 then begin
        MakeMsgToDataDDE(MsgForDDE_n);
        MakeMsgToReportDDE(MsgForDDE_n);
        end else begin
        if Data_Ugi[0]=9 then begin
          MakeMsgToDataDDE(MsgForDDE_e);
          MakeMsgToReportDDE(MsgForDDE_e);
          end else begin
            MakeMsgToDataDDE(MsgForDDE_s);
            MakeMsgToReportDDE(MsgForDDE_s);
            end;
        end;
      end;
    if IsSaveBinFile then GetNewBinFile;
    except
    raise
    end;
  end;

  {*********************************}
  procedure TMainForm.MakeCikleIzmeren(Sender: TObject);
  var
    Mas1:TDataRecord;
    V1:Real;
```

```
  Tim1:TDateTime;
  k0:Integer;
begin
try
  if IsFiltrInput then FiltrateInputDataUgi;
  if Data_Ugi[0]<4 then CalcCikleMeteo;
  Mas1[0]:=TC; Mas1[1]:=Vx; Mas1[2]:=Vy; Mas1[3]:=Vz;
  Mas1[4]:=MMdavl; Mas1[5]:=Otvlag;
  Mas1[6]:=GammaX; Mas1[7]:=GammaY;
  Mas1[8]:=DatTC;
  Tim1:=Now;
  AddRecGlobData(Mas1,Data_Ugi[0],Tim1,TimeUsredn,Met_Data,Met_SrVar);
  if RegimIzmer=0 then Begin
  if NumPosGlobData=High(Met_Data) then k0:=0
                          else k0:=NumPosGlobData+1;
  if Met_Data[NumPosGlobData].TDr>=
   Met_Data[k0].TDr+EncodeTime(0,TimeUsredn div 60,0,0)- EncodeTime(0,0,2,0) then begin
    if Met_SrVar[0]<MeteoData.Tmin then MeteoData.Tmin:=Met_SrVar[0];
    if Met_SrVar[0]>MeteoData.Tmax then MeteoData.Tmax:=Met_SrVar[0];
    if Met_SrVar[8]<MeteoData.Trmin then MeteoData.Trmin:=Met_SrVar[8];
    if Met_SrVar[8]>MeteoData.Trmax then MeteoData.Trmax:=Met_SrVar[8];
    MeteoData_CalcWindMgnoven(NumPosGlobData,V1);
    if V1>=0 then begin
      if V1<MeteoData.Vmin then MeteoData.Vmin:=V1;
      if V1>MeteoData.Vmax then MeteoData.Vmax:=V1;
      end;
    end;
  End;
  if IsTehnologyDDE then UpdateDataStrDDE(Tim1,Data_Ugi[0]);
  if IsSaveBinFile and (Data_Ugi[0]<9) then
     AddDataBufBin(Data_Ugi[0],TC,Vx,Vy,Vz,MMdavl,Otvlag);
  if IzmerenForm.Visible then with IzmerenForm do begin
    if Graf_Data=nil then IzmerenForm.FormPaint(Sender)
                 else DrawLines(High(Graf_Data));
  end;
except
raise
end;
end;

{*********************************}
procedure TMainForm.UpdateStopIzmeren(Sender: TObject);
begin
try
  Screen.Cursor:=crHourGlass;
  if not IsModem then
    TimeCikle:=UpdateTimeCikle(Met_Data);
  CalcStopMeteo;
  FormPaint(Sender);
  if IzmerenForm.Visible then IzmerenForm.FormPaint(Sender);
  if IsSaveBinFile and (Met_Data<>nil) then begin
    WriteFromBufToBinF;
    Close_BinF;
    end;
  if ValidGlobData(Met_Data)>=60 then begin
  SaveInfoFile;
  SaveReportFile;
  SaveDataFile;
```

```
    SaveDataBase;
    SaveArrStrFile;
    if IsTehnologyDDE then  begin
      SaveDataForDDE;
      SaveReportForDDE;
      end;
    end; {if ValidGlobData(Met_Data)}
    if RegimIzmer=1 then SetNewCurStartSave;
  finally
  Screen.Cursor:=crDefault;
  if isPhoneForm then begin
    if (not NotBreakPhone) and (RegimIzmer=1) and  (Now<CurStartRing)
    then PhoneForm.Close;
    end;
  end;
  end;

  //*****************************************
  procedure TMainForm.StartReadDataUGI;
  begin
    try
    CloseMenu;
    ClearRxBuf_AMKport(Handle_AMK,InQue);
    StartCenzuir;
    ChekTimeOutSred:=Now+EncodeTime(0,0,1,0);
    GetStartStopTime;
    IsSbojPhone:=false;
    BreakReadDataUGI:=true;
    ZapretReadCOMbuf:=false;
    except
    raise
    end;
  end;

  {*********************************}
  procedure TMainForm.StopReadData;
  begin
    BreakReadDataUGI:=false;
    IsRunAutomatic:=false;
    StopTime:=Now;
  end;

  {*********************************}
  procedure TMainForm.ControlReadDataUGI;
  label 1;
  var
    Errors:DWord;
    k:Byte;
    chec:Integer;
    COM_Stat:tCOMStat;
  begin
    try
    try
    Screen.Cursor:=crHourGlass;
    StartAMKport_UGI;
    if not IsOpenCOMport then begin
      Data_Ugi[0]:=9;
      GOTO 1;
```

```
    end;
  Data_Ugi[0]:=0;
  ClearRxBuf_AMKport(Handle_AMK,InQue);
  if isFiltrInput then
    for k:=0 to 3 do
      CountNumFiltr1[k]:=0;
  StartCenzuir;
  MyDelay(DelayDataCable);
  Chec:=0;
  ClearCommError(Handle_AMK,Errors,@COM_Stat);
  while COM_Stat.cbInQue div 3 >1 do Begin
      if ReadUGI_Data(Handle_AMK,CountByteData_UGI+1,BufData_UGI,OverLap_AMK) then begin
      GetDataUgi;
      Inc(Chec);
      if isFiltrInput then
        for k:=0 to 3 do
          Filtr1AddNumData(k,Data_Ugi[k+1],UGI_Amin,UGI_Amax,MaxMasFiltrInput);
      for k:=0 to CountParam_UGI do
        InputData[k]:=Data_Ugi[k];
      end; {if ReadDataUGI}
      ClearCommError(Handle_AMK,Errors,@COM_Stat);
  End;{while}
  if Chec=0 then begin
      Data_Ugi[0]:=10;
      GetMyErrorDlg(MsgErr101+MsgErr10_1);
      GOTO 1;
      end else begin
      if Data_Ugi[0]=4 then begin
        GetMyErrorDlg(MsgErr4);
        GOTO 1;
        end;
      if Data_Ugi[0]=5 then begin
        GetMyErrorDlg(MsgErr5);
        GOTO 1;
        end;
      if Data_Ugi[0]=6 then begin
        GetMyErrorDlg(MsgErr6);
        GOTO 1;
        end;
      if Data_Ugi[0]=7 then begin
        GetMyErrorDlg(MsgErr7);
        GOTO 1;;
        end;
      end; {if Chec=0}
 1:finally
  Screen.Cursor:=crDefault;
end;
  except
  raise
  end;
end;

{*********************************}
procedure TMainForm.Timer1Timer(Sender: TObject);
var
  Errors:DWord;
  CanRead1,DelayRead1,CountRead1,i:Cardinal;
  COM_Stat:tCOMStat;
```

```
begin
  try
  ClearCommError(Handle_AMK,Errors,@COM_Stat);
  UpdateStatusWork;
  if IsModem then begin
    if IsSbojPhone then RestoreModems(Sender);
    if isPhoneForm then PhoneForm.ViewAnswerModem;
    end;
  if (RegimIzmer=1) and  IsRunAutomatic then begin
    ControlAutomatic(Sender);
    if Now >= StopTime then BreakReadDataUGI:=true;
    end;
  if not ZapretReadCOMbuf then begin
    ZapretReadCOMbuf:=true;
    if IsOpenCOMport then Begin
    if IsModem then DelayRead1:=DelayDataModem
            else DelayRead1:=DelayDataCable;
    CanRead1:=trunc(2*TimeCikle)+1;
    if DelayRead1<CanRead1 then DelayRead1:=CanRead1;
    if BreakReadDataUGI then CanRead1:=1 else begin
      if IsModem then CanRead1:=BlockDataModem*CoefFreqIzm
              else CanRead1:=BlockDataCable*CoefFreqIzm;
      end;
    ClearCommError(Handle_AMK,Errors,@COM_Stat);
    if COM_Stat.cbInQue=InQue then begin
      if Now <= CheckTimeInQue+EncodeTime(0,0,DelayRead1,0) then
      Data_UGI[0]:=11 else Data_UGI[0]:=9;
      end else begin
      InQue:=COM_Stat.cbInQue;
      end; {if COM_Stat.cbInQue=InQue}
    if COM_Stat.cbInQue div 3 >=CanRead1 then Begin
    CountRead1:=0;
    while COM_Stat.cbInQue div (CountByteData_UGI+3)>=1 do Begin
      if CoefFreqIzm>1 then
        for i:=0 to CoefFreqIzm-2 do
          ReadUGI_Data(Handle_AMK,CountByteData_UGI+1,BufData_UGI,OverLap_AMK);
      if ReadUGI_Data(Handle_AMK,CountByteData_UGI+1,BufData_UGI,OverLap_AMK) then begin
        Inc(CountRead1);
        GetDataUgi;
        GetInputData;
        case RegimVabor.ItemIndex of
          0:MakeCikleIzmeren(Sender);
          1:MakeCikleCalibr(Sender);
          2:MakeCikleControl(Sender);
          end;
        end; {if ReadDataUGI}
    End; {while}
     if CountRead1=0 then Data_UGI[0]:=10;
    ControlDataMainForm;
    End; {if COM_Stat.cbInQue}
    End; {if IsOpenCOMport}
    ControlWorkDevice;
    if IsModem then ControlWorkModems;
    if RegimVabor.ItemIndex=0 then begin
      if IsTehnologyDDE then ControlUpdateDDE;
      ControlSaveFiles;
      end;
    if BreakReadDataUGI then begin
```

```
      case RegimVabor.ItemIndex of
        0:UpdateStopIzmeren(Sender);
        1:UpdateStopCalibr(Sender);
        2:UpdateStopControl(Sender);
        end;
        if not ((RegimIzmer=1) and IsRunAutomatic) then begin
         OpenMenu;
         if IsModem and (not isPhoneForm) then CloseDevice;
         end;
      end else begin
      if not IsSbojPhone then ZapretReadCOMbuf:=false;
      end;
  end; {if not ZapretReadCOMbuf}
  if Now > TimeCloseProgram then ExitProgrExecute(Sender);
  if isNewReStartProgam and BreakReadDataUGI and IsAvtoLoadProgram
    and (GetTickCount>TimeStartAvtoIzmer) then IzmerenExecute(Sender);
  except
  raise
  end;
end;

{**********************************}
procedure TMainForm.CheckErrorsDevice;
var
  i:Integer;
begin
  if CountByteData_UGI div 2=10 then begin
  if IsDeviceHu then
  if (Data_UGI[i+4]<=50) or (Data_UGI[i+4]>=65535) then Data_UGI[0]:=8;
  if (Data_UGI[i+3]<=0) or (Data_UGI[i+3]>=65535) then Data_UGI[0]:=7;
  if (Data_UGI[i+1]<=0) or (Data_UGI[i+1]>=65535) then Data_UGI[0]:=7;
  if (Data_UGI[i+2]<=0) or (Data_UGI[i+2]>=65535) then Data_UGI[0]:=7;
  end;
  if IsDeviceHu then
    if (Data_UGI[6]<=50) or (Data_UGI[6]>=65535) then Data_UGI[0]:=6;
  if IsDeviceP then
    if (Data_UGI[5]<=50) or (Data_UGI[5]>=65535) then Data_UGI[0]:=5;
  for i:=0 to 3 do
  if (Data_UGI[i+1] < UGI_Amin) or (Data_UGI[i+1] > UGI_Amax)  then Data_UGI[0]:=4;
  if IsCenzuir and (RegimVabor.ItemIndex<>2) and (Data_UGI[0]<4) then
   MakeCikleCenzuir(UGI_Fkg);
end;

{**********************************}
procedure TMainForm.GetDataUGI;
var
  i,ks:Byte;
begin
  ks:=0;
  for i:=0 to CountByteData_UGI-1 do
    ks:=ks+BufData_UGI[i];
  if ks = BufData_UGI[CountByteData_UGI] then Data_Ugi[0]:=0
                                  else Data_Ugi[0]:=3;
  for i:=1 to CountParam_UGI do
    Data_Ugi[i]:=BufData_UGI[2*(i-1)]+256*BufData_UGI[2*(i-1)+1];
  if RegimVabor.ItemIndex<>2 then CheckErrorsDevice;
end;
END.
```

```
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT UnAMKini;
INTERFACE
uses
  SysUtils, WinProcs, Forms, IniFiles;

const
  NameDevice = 'АМК-03';
  NameProgram =' для программы версии METEO 3.0.R';
  LineCaption_R:String = '*     " '+NameDevice+'", рег.№ ';
  Capt_SonicBox_1 = 'Расстояния между датчиками термоанемометра:';
  Capt_SonicBox_2 = 'Временные задержки в каналах термоанемометра:';

  {Dir}
  DataDir:    String = 'C:\AMK_03\DATA';

  {Application}
  IsStatistica:Boolean = true; {признак вывода в отчет статистических данных}
  IsTurbulent:Boolean = true; {признак вывода в отчет турбулентных параметров}
  IsDeviceP:Boolean = true; {признак подключения датчика давления}
  IsDeviceHu:Boolean = true; {признак подключения датчика относительной влажности}
  IsCutHu_100:Boolean = true;{признак отсечения значений Hu, больших 100%}
  IsCorrectZ:Boolean = true;  {признак учета затенения}
  TimeTimerSpeed:Cardinal=100; {скорость опроса таймера - 10 раз в 1 сек}
  IsSaveReportFile:Boolean = true;{признак сохранения файла *.00R }
  IsSaveInfoFile:Boolean = false;{признак сохранения файла *.00i }
  IsSaveDataFile:Boolean = false; {признак сохранения файла *.00D }
  IsSaveBinFile:Boolean = false; {признак сохранения файла *.00B }
  IsSaveArrStrFile:Boolean = false; {признак сохранения файла *.00T }
  IsTehnologyDDE:Boolean = false; {признак использования DDE-технологии}

  {Default}
  MMdavlKl:Real=750;
  OtVlagKl:Real=65;
  DeviceOrient:Real=0;
  SizeDirWind:Byte=1;
  SizeDavlen:Byte=1;
  NumTimePPU25:Byte=1;

  TypeSurface:Byte=1;
  LogNumServer:String ='    12';

  {Communication}
  COM_Number:Byte = 0;{0-COM1, 1-COM2, 2-COM3, 4-COM4}
  COM_BaudRate:DWord = 57600;
  COM_SizeRxBuf:DWord=24000;
  COM_SizeTxBuf:DWord=110;
  CoefFreqIzm:Byte=1;
  BlockDataCable:Word=2;
  DelayDataCable:Word=1;

  IsModem:Boolean = false;
  NumberTelefon:String = '49-15-79';
  BlockDataModem:Integer=5;
  DelayDataModem:Integer=30;
```

```
  DelayConnectModems:Integer=3;
  DelayAutRingModem:Integer=10;
  MethodRingModem:Byte = 0;

 {Measurements}
  TimeUsredn:Word=60;
  TimeCikle:Real=0.1;
  TimeUsredPPU:Byte=0;
  PerReadPPU:Word=2;
  CountReadPPU:Word=30;
  BegHourAutSave:Word = 12;
  BegMinAutSave:Word = 0;
  InterHourAutSave:Word = 3;
  InterMinAutSave:Word = 0;

  RegimIzmer:Byte = 0;
  IsSaveBazaData:Boolean = false;
  NotBreakPhone:Boolean = true;
  IsAvtoLoadProgram:Boolean = false;

 {Filtres}
  UGI_Amin:Word = 11000;
  UGI_Amax:Word = 24000;
  IsCenzuir:Boolean = true;
  VolumeCenzuir:Word = 7;
  BoundsCenzuir:Word = 30;
  MaxMasFiltrInput:Word = 7;
  IntWinFiltrInput:Word = 70;
  IsFiltrInput:Boolean = false;
  IsFiltrOutput:Boolean = false;
 {Graph}
  WindowColor:Integer=17;
  DiagColor:Integer=8;
  ArrowColor:Integer=4;
  EditColor:Integer=4;
  DigitColor:Integer=14;
  GridColor:Integer=7;
  LabelColor:Integer=0;
  LineColor:Integer=9;
  LineStyle:Integer=0;
  LineTick:Integer=1;
  IsOutGrid:Boolean = true;
  rayTC:Integer=4;
  rayVg:Integer=10;
  rayFi:Integer=360;
  rayVz:Integer=2;
  raySi:Integer=2;
  rayDi:Integer=500;
  rayti:Integer=500;

var
  IniFile:TIniFile;
  IniFileName:String;

procedure WriteIniFile;
procedure ReadIniFile;

IMPLEMENTATION
```

```
//*****************************************************************
procedure MyWriteFloatToIni(const HeadStr,ItemStr:String; par:Real; W,D:Byte);
var
  S:String;
begin
  str(par:W:D,S);
  IniFile.WriteString(HeadStr,ItemStr,S);
end;

//*******************************************************************
function MyReadFloatFromIni(const HeadStr,ItemStr:String; def:Real):Real;
var
  S:String;
  code:Integer;
  A:Real;
begin
  S:=IniFile.ReadString(HeadStr,ItemStr,'');
  Val(S,A,code);
  if code <> 0 then Result:=def else Result:=A;
end;

//****************************************
procedure WriteIniFile;
var
  per:Real;
Begin
 try
 {Dir}
  IniFile.WriteString('Dir','DataDir',DataDir);
 {Application}
  IniFile.WriteBool('Application','IsStatistica',IsStatistica);
  IniFile.WriteBool('Application','IsTurbulent',IsTurbulent);
  IniFile.WriteBool('Application','IsDeviceP',IsDeviceP);
  IniFile.WriteBool('Application','IsDeviceHu',IsDeviceHu);
  IniFile.WriteBool('Application','IsCutHu_100',IsCutHu_100);
  IniFile.WriteBool('Application','IsCorrectZ',IsCorrectZ);
  per:=1000/TimeTimerSpeed;
  MyWriteFloatToIni('Application','FrequencyTimer',per,6,1);
  IniFile.WriteBool('Application','IsSaveReportFile',IsSaveReportFile);
  IniFile.WriteBool('Application','IsSaveInfoFile',IsSaveInfoFile);
  IniFile.WriteBool('Application','IsSaveDataFile',IsSaveDataFile);
  IniFile.WriteBool('Application','IsSaveBinFile',IsSaveBinFile);
  IniFile.WriteBool('Application','IsSaveArrStrFile',IsSaveArrStrFile);
  IniFile.WriteBool('Application','IsTehnologyDDE',IsTehnologyDDE);
 {Default}
  MyWriteFloatToIni('Default','MMdavlKl',MMdavlKl,6,1);
  MyWriteFloatToIni('Default','OtVlagKl',OtVlagKl,6,1);
  MyWriteFloatToIni('Default','HorOrient',DeviceOrient,6,1);
  IniFile.WriteInteger('Default','SizeDirWind',SizeDirWind);
  IniFile.WriteInteger('Default','SizeDavlen',SizeDavlen);
  IniFile.WriteInteger('Default','TypeSurface',TypeSurface);
  IniFile.WriteString('Default','LogNumServer',LogNumServer);
 {Communication}
  IniFile.WriteInteger('Communication','COM_Number',COM_Number);
  IniFile.WriteInteger('Communication','COM_BaudRate',COM_BaudRate);
  IniFile.WriteInteger('Communication','COM_SizeRxBuf',COM_SizeRxBuf);
  IniFile.WriteInteger('Communication','COM_SizeTxBuf',COM_SizeTxBuf);
```

```
  IniFile.WriteInteger('Communication','CoefFreqIzm',CoefFreqIzm);
  IniFile.WriteInteger('Communication','BlockDataCable',BlockDataCable);
  IniFile.WriteInteger('Communication','DelayDataCable',DelayDataCable);
  IniFile.WriteBool('Communication','IsModem',IsModem);
  IniFile.WriteString('Communication','NumberTelefon',NumberTelefon);
  IniFile.WriteInteger('Communication','BlockDataModem',BlockDataModem);
  IniFile.WriteInteger('Communication','DelayDataModem',DelayDataModem);
  IniFile.WriteInteger('Communication','DelayConnectModems',DelayConnectModems);
  IniFile.WriteInteger('Communication','DelayAutRingModem',DelayAutRingModem);
  IniFile.WriteInteger('Communication','MethodRingModem',MethodRingModem);
  {Measurements}
  IniFile.WriteInteger('Measurements','TimeUsredn',TimeUsredn);
  MyWriteFloatToIni('Measurements','Cikle',TimeCikle,10,7);
  IniFile.WriteInteger('Measurements','RegimIzmeren',RegimIzmer);
  IniFile.WriteBool('Measurements','IsSaveBazaData',IsSaveBazaData);
  IniFile.WriteInteger('Measurements','BegHourAutSave',BegHourAutSave);
  IniFile.WriteInteger('Measurements','BegMinAutSave',BegMinAutSave);
  IniFile.WriteInteger('Measurements','InterHourAutSave',InterHourAutSave);
  IniFile.WriteInteger('Measurements','InterMinAutSave',InterMinAutSave);
  IniFile.WriteBool('Measurements','IsAvtoLoadProgram',IsAvtoLoadProgram);
  IniFile.WriteBool('Measurements','NotBreakPhone',NotBreakPhone);
  {Filtres}
  IniFile.WriteInteger('Filtres','Amin',UGI_Amin);
  IniFile.WriteInteger('Filtres','Amax',UGI_Amax);
  IniFile.WriteBool('Filtres','IsCenzuir',IsCenzuir);
  IniFile.WriteInteger('Filtres','VolumeCenzuir',VolumeCenzuir);
  IniFile.WriteInteger('Filtres','BoundsCenzuir',BoundsCenzuir);
  IniFile.WriteBool('Filtres','isFiltrInput',isFiltrInput);
  IniFile.WriteInteger('Filtres','MaxMasFiltrInput',MaxMasFiltrInput);
  IniFile.WriteInteger('Filtres','IntWinFiltrInput',IntWinFiltrInput);
  IniFile.WriteBool('Filtres','isFiltrOutput',isFiltrOutput);
  {Graph}
  IniFile.WriteInteger('Graph','WindowColor',WindowColor);
  IniFile.WriteInteger('Graph','DiagColor',DiagColor);
  IniFile.WriteInteger('Graph','ArrowColor',ArrowColor);
  IniFile.WriteInteger('Graph','EditColor',EditColor);
  IniFile.WriteInteger('Graph','DigitColor',DigitColor);
  IniFile.WriteInteger('Graph','GridColor',GridColor);
  IniFile.WriteInteger('Graph','LabelColor',LabelColor);
  IniFile.WriteInteger('Graph','LineColor',LineColor);
  IniFile.WriteInteger('Graph','LineStyle',LineStyle);
  IniFile.WriteInteger('Graph','LineTick',LineTick);
  IniFile.WriteBool('Graph','IsOutGrid',IsOutGrid);
  IniFile.WriteInteger('Graph','rayT',rayTC);
  IniFile.WriteInteger('Graph','rayV',rayVg);
  IniFile.WriteInteger('Graph','rayF',rayFi);
  IniFile.WriteInteger('Graph','rayVz',rayVz);
  IniFile.WriteInteger('Graph','rayS',raySi);
  IniFile.WriteInteger('Graph','rayD',rayDi);
  IniFile.WriteInteger('Graph','rayti',rayti);
  except
  raise
  end;
end;

//*************************************************
procedure ReadIniFile;
var
```

```
 per:Real;
 Name1:String;
Begin
 try
 Name1:=ParamStr(0);
 IniFileName:=Copy(Name1,1,Length(Name1)-4)+'.INI';
 IniFile:=TIniFile.Create(IniFileName);
 {Dir}
 DataDir:=ExtractFileDir(ParamStr(0))+'\DATA';
 DataDir:=IniFile.ReadString('Dir','DataDir',DataDir);
 {Application}
 IsStatistica:=IniFile.ReadBool('Application','IsStatistica',IsStatistica);
 IsTurbulent:=IniFile.ReadBool('Application','IsTurbulent',IsTurbulent);
 IsDeviceP:=IniFile.ReadBool('Application','IsDeviceP',IsDeviceP);
 IsDeviceHu:=IniFile.ReadBool('Application','IsDeviceHu',IsDeviceHu);
 IsCutHu_100:=IniFile.ReadBool('Application','IsCutHu_100',IsCutHu_100);
 IsCorrectZ:=IniFile.ReadBool('Application','IsCorrectZ',IsCorrectZ);
 per:=1000/TimeTimerSpeed;
 per:=MyReadFloatFromIni('Application','FrequencyTimer',per);
 TimeTimerSpeed:=round(1000/per);
 if TimeTimerSpeed<10 then TimeTimerSpeed:=10;
 if TimeTimerSpeed>50000 then TimeTimerSpeed:=50000;
 IsSaveInfoFile:=IniFile.ReadBool('Application','IsSaveInfoFile',IsSaveInfoFile);
 IsSaveReportFile:=IniFile.ReadBool('Application','IsSaveReportFile',IsSaveReportFile);
 IsSaveDataFile:=IniFile.ReadBool('Application','IsSaveDataFile',IsSaveDataFile);
 IsSaveBinFile:=IniFile.ReadBool('Application','IsSaveBinFile',IsSaveBinFile);
 IsSaveBazaData:=IniFile.ReadBool('Application','IsSaveBazaData',IsSaveBazaData);
 IsSaveArrStrFile:=IniFile.ReadBool('Application','IsSaveArrStrFile',IsSaveArrStrFile);
 IsTehnologyDDE:=IniFile.ReadBool('Application','IsTehnologyDDE',IsTehnologyDDE);
 {Default}
 MMdavlKl:=MyReadFloatFromIni('Default','MMdavlKl',MMdavlKl);
 OtVlagKl:=MyReadFloatFromIni('Default','OtVlagKl',OtVlagKl);
 DeviceOrient:=MyReadFloatFromIni('Default','DeviceOrient',DeviceOrient);
 SizeDirWind:=IniFile.ReadInteger('Default','SizeDirWind',SizeDirWind);
 SizeDavlen:=IniFile.ReadInteger('Default','SizeDavlen',SizeDavlen);
 TypeSurface:=IniFile.ReadInteger('Default','TypeSurface',TypeSurface);
 LogNumServer:=IniFile.ReadString('Default','LogNumServer',LogNumServer);
 {Communication}
 COM_Number:=IniFile.ReadInteger('Communication','COM_Number',COM_Number);
 COM_BaudRate:=IniFile.ReadInteger('Communication','COM_BaudRate',COM_BaudRate);
 COM_SizeRxBuf:=IniFile.ReadInteger('Communication','COM_SizeRxBuf',COM_SizeRxBuf);
 COM_SizeTxBuf:=IniFile.ReadInteger('Communication','COM_SizeTxBuf',COM_SizeTxBuf);
 CoefFreqIzm:=IniFile.ReadInteger('Communication','CoefFreqIzm',CoefFreqIzm);
 BlockDataCable:=IniFile.ReadInteger('Communication','BlockDataCable',BlockDataCable);
 DelayDataCable:=IniFile.ReadInteger('Communication','DelayDataCable',DelayDataCable);
 IsModem:=IniFile.ReadBool('Communication','IsModem',IsModem);
 NumberTelefon:=IniFile.ReadString('Communication','NumberTelefon',NumberTelefon);
 BlockDataModem:=IniFile.ReadInteger('Communication','BlockDataModem',BlockDataModem);
 DelayDataModem:=IniFile.ReadInteger('Communication','DelayDataModem',DelayDataModem);
 DelayConnectModems:=IniFile.ReadInteger('Communication','DelayConnectModems',DelayConnectModems);
 DelayAutRingModem:=IniFile.ReadInteger('Communication','DelayAutRingModem',DelayAutRingModem);
 MethodRingModem:=IniFile.ReadInteger('Communication','MethodRingModem',MethodRingModem);
 {Measurements}
 TimeUsredn:=IniFile.ReadInteger('Measurements','TimeUsredn',TimeUsredn);
 TimeCikle:=MyReadFloatFromIni('Measurements','Cikle',TimeCikle);
 IniFile.WriteInteger('Measurements','RegimIzmeren',RegimIzmer);
 IniFile.WriteBool('Measurements','IsSaveBazaData',IsSaveBazaData);
 IniFile.WriteInteger('Measurements','BegHourAutSave',BegHourAutSave);
```

```
  IniFile.WriteInteger('Measurements','BegMinAutSave',BegMinAutSave);
  IniFile.WriteInteger('Measurements','InterHourAutSave',InterHourAutSave);
  IniFile.WriteInteger('Measurements','InterMinAutSave',InterMinAutSave);
  IniFile.WriteBool('Measurements','IsAvtoLoadProgram',IsAvtoLoadProgram);
  IniFile.WriteBool('Measurements','NotBreakPhone',NotBreakPhone);
  {Filtres}
  UGI_Amin:=IniFile.ReadInteger('Filtres','Amin',UGI_Amin);
  UGI_Amax:=IniFile.ReadInteger('Filtres','Amax',UGI_Amax);
  IsCenzuir:=IniFile.ReadBool('Filtres','IsCenzuir',IsCenzuir);
  VolumeCenzuir:=IniFile.ReadInteger('Filtres','VolumeCenzuir',VolumeCenzuir);
  BoundsCenzuir:=IniFile.ReadInteger('Filtres','BoundsCenzuir',BoundsCenzuir);
  isFiltrInput:=IniFile.ReadBool('Filtres','isFiltrInput',isFiltrInput);
  MaxMasFiltrInput:=IniFile.ReadInteger('Filtres','MaxMasFiltrInput',MaxMasFiltrInput);
  IntWinFiltrInput:=IniFile.ReadInteger('Filtres','IntWinFiltrInput',IntWinFiltrInput);
  isFiltrInput:=IniFile.ReadBool('Filtres','isFiltrInput',isFiltrInput);
  {Graph}
  WindowColor:=IniFile.ReadInteger('Graph','WindowColor',WindowColor);
  DiagColor:=IniFile.ReadInteger('Graph','DiagColor',DiagColor);
  ArrowColor:=IniFile.ReadInteger('Graph','ArrowColor',ArrowColor);
  EditColor:=IniFile.ReadInteger('Graph','EditColor',EditColor);
  DigitColor:=IniFile.ReadInteger('Graph','DigitColor',DigitColor);
  GridColor:=IniFile.ReadInteger('Graph','GridColor',GridColor);
  LabelColor:=IniFile.ReadInteger('Graph','LabelColor',LabelColor);
  LineColor:=IniFile.ReadInteger('Graph','LineColor',LineColor);
  LineStyle:=IniFile.ReadInteger('Graph','LineStyle',LineStyle);
  LineTick:=IniFile.ReadInteger('Graph','LineTick',LineTick);
  IsOutGrid:=IniFile.ReadBool('Graph','IsOutGrid',IsOutGrid);
  rayTC:=IniFile.ReadInteger('Graph','rayT',rayTC);
  rayVg:=IniFile.ReadInteger('Graph','rayV',rayVg);
  rayFi:=IniFile.ReadInteger('Graph','rayF',rayFi);
  rayVz:=IniFile.ReadInteger('Graph','rayVz',rayVz);
  raySi:=IniFile.ReadInteger('Graph','rayS',raySi);
  rayDi:=IniFile.ReadInteger('Graph','rayD',rayDi);
  rayti:=IniFile.ReadInteger('Graph','rayti',rayti);
 except
  raise;
  end;
End;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT ApllDlgs;
interface
uses
  Windows, WinProcs, SysUtils, Classes, Forms, Controls, ExtCtrls, StdCtrls, Buttons;

const
  Capt_Viewer = 'Отчет';
  Capt_SaveAsReport = 'Сохранить как файл отчета...';
  Msg_Warning='Опасное действие ! ';
  Msg_ChangeGraduir='Вы действительно хотите изменить градуировочные параметры устройства ?';
  Msg_NoSaveGraduir='Нет достоверных данных для завершения градуировки термоанемометра.';
  Msg_SaveGraduir = ' Сохранить результаты градуировки ?';
  Msg_File_0 = 'Файл ';
  MsgErr_FileName = ' является неверным именем файла.';
  Msg_Modify = 'Не могу выполнить чтение файла  ';
```

```
  Msg_File_NoFind = ' не найден.';
  Msg_File_Format = ' имеет неверный формат данных';
  Msg_BukSave0 = ' Файл ';
  Msg_BukSave1 = ' уже существует. Заменить его ?';
  Msg_NoDir = 'Не могу создать директорию (папку) ';
  Msg_EditFloat=' не является вещественным числом';
  MsgErr_so='Неправильный набор числа в поле ввода для ';
  MsgErr_sf='Неверное значение для ';
  MsgErr_or = ' или ';
  MsgErr_and= ' и ';
  MsgErr_differ = 'разности между ';
  MsgErr_Period = 'не должна быть меньше времени усреднения.';
  Msg_NoDataPrint = ' Нет данных для печати.';
  Msg_NoSelectText = ' Не выделен текст для печати. ';

function GetAdminParol(const sys1,stand1:ShortString; var cur1:ShortString):WordBool;
procedure GetMyErrorDlg(const Msg1:ShortString);
function GetMyConfirm2Dlg(const Msg1:ShortString):Integer;
function GetMyConfirm3Dlg(const Msg1:ShortString):Integer;
procedure GetMyInformDlg(const Msg1:ShortString);
procedure GetMyInvalidDlg(const Msg1:ShortString);

function ControlEdit(const MyEdit:TEdit):Boolean; far; external 'StanDlgs.DLL';
function GetNameFileDef(const Time1:TDateTime): ShortString;far; external 'StanDlgs.DLL';
function TimeTo_MyStr(t1:TDateTime):ShortString;far; external 'StanDlgs.DLL';
function DateTo_MyStr(t1:TDateTime):ShortString;far; external 'StanDlgs.DLL';
function MyStr_ToTime(const MyStr:ShortString; var Time1:TDateTime):Boolean;far; external 'StanDlgs.DLL';
function MyStr_ToDate(const MyStr:ShortString; var Data1:TDateTime):Boolean;far; external 'StanDlgs.DLL';
function WordToChar(w1,n1:Word):Char;far; external 'StanDlgs.DLL';
function CharToWord(Ch1:Char;n1:Word):Word;far; external 'StanDlgs.DLL';

implementation

function Get_AdminParol(const Title1,sys1,stand1:ShortString; var cur1:ShortString):WordBool;far; external
'StanDlgs.DLL';
procedure Get_ErrorDlg(const Title1,Msg1:ShortString);far; external 'StanDlgs.DLL';
function Get_Confirm2Dlg(const Title1,Msg1:ShortString):Integer;far; external 'StanDlgs.DLL';
function Get_Confirm3Dlg(const Title1,Msg1:ShortString):Integer;far; external 'StanDlgs.DLL';
procedure Get_InformDlg(const Title1,Msg1:ShortString); far; external 'StanDlgs.DLL';
procedure Get_InvalidDlg(const Title1,Msg1:ShortString);far; external 'StanDlgs.DLL';

{*****************************************}
function GetAdminParol(const sys1,stand1:ShortString; var cur1:ShortString):WordBool;
var
  st1:ShortString;
begin
  st1:=Application.Title;
  Result:=Get_AdminParol(st1,sys1,stand1,cur1);
end;

//*********************************************
procedure GetMyErrorDlg(const Msg1:ShortString);
var
  st1:ShortString;
begin
  st1:=Application.Title;
  Get_ErrorDlg(st1,Msg1);
end;

//****************************************************
```

```
function GetMyConfirm2Dlg(const Msg1:ShortString):Integer;
var
  st1:ShortString;
begin
  st1:=Application.Title;
  Result:=Get_Confirm2Dlg(st1,Msg1);
end;

//*******************************************************
function GetMyConfirm3Dlg(const Msg1:ShortString):Integer;
var
  st1:ShortString;
begin
  st1:=Application.Title;
  Result:=Get_Confirm3Dlg(st1,Msg1);
end;

//**************************************************
procedure GetMyInformDlg(const Msg1:ShortString);
var
  st1:ShortString;
begin
  st1:=Application.Title;
  Get_InformDlg(st1,Msg1);
end;

//**************************************************
procedure GetMyInvalidDlg(const Msg1:ShortString);
var
  st1:ShortString;
begin
  st1:=Application.Title;
  Get_InvalidDlg(st1,Msg1);
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT UnAMKsys;
interface
uses
  SysUtils, WinProcs, Classes, Controls, Forms;

const
  StrSysDevice='3gkafu.lo89vjc9cj de9 e9009 09dflfacl dhllkeofeo8cnlTILKL';

var
  SysFile:File;
  SysData:TStringList;
  regNumber,st_Copyr,Par_Standart,Par_System,ParolCurrent:ShortString;
  SysFileName, ProgramDir,st_FullNamePr,st_BriefNamePr:String;
  BasaFileName,AnalysisFileName:String;
  RastMM:array[0..3] of Real;
  UGI_Dela:array[0..3] of Integer;
  IsDCB_15:Byte;
  UGI_Fkg:Integer;
  Tcalibr,UGI_Tcoef,UGI_hor,UGI_Z,SetMinVolt,UGI_Coefz,UGI_Fiz,UGI_Az0,UGI_Az1,
  UGI_Az2,UGI_Az3,UGI_Az4,P_P0,P_x0,P_ln,P_delta,P_T0,PT_ln,PT_kv,PT_kb,r_ln,
```

```
  r_kv,r_kb,r_X0,r_r0,r_delta,Tr_T0,Tr_x0,Tr_ln,Tr_kv,Tr_delta,G_x0,G_y0,G_xd,
  G_yd,G_hor,G_xdelta,G_ydelta:Real;
  AvtoZapuskIzm,IsPhoneForm,IsConnectPhone,IsSbojPhone,IsRingPhone:Boolean;

function GetDataAMKsys:Boolean;
procedure UpDateAMKsys;

implementation
uses UnAMKini,ApllDlgs;

//*****************************************
function WriteSysFile:Boolean;
label 11;
var
  i,k,N:Integer;
  b1:Word;
  Pch:array[0..255] of Char;
begin
  try
  try
  Result:=false;
  Screen.Cursor:=crHourGlass;
  Rewrite(SysFile,1);
  if SysData.Count > 0 then Begin
  with SysData do Begin
  for k:=0 to Count-1 do Begin
    StrPCopy(PCh,Strings[k]);
    for i:=0 to Length(Strings[k])-1 do begin
      b1:=CharToWord(PCh[i],StrToInt(Copy(regNumber,Length(regNumber)-2,3)));
      BlockWrite(SysFile,b1,SizeOf(b1),N);
      if N <> SizeOf(b1) then GOTO 11;
      end;
    b1:=64531;
    BlockWrite(SysFile,b1,SizeOf(b1),N);
    if N <> SizeOf(b1) then GOTO 11;
    End;
  End;
  Result:=true;
  End;
  except
  Result:=false;
  end;
  finally
11:System.CloseFile(SysFile);
   Screen.Cursor:=crDefault;
  end;
end;

{*****************************************************************************}
function ReadSysFile:Boolean;
var
  S:String;
  J00:Integer;
  Ch:Char;
  s1:String[1];
  NumR:Integer;
  par:Word;
  st:String;
begin
  try
```

```
    try
    Result:=true;
    Screen.Cursor:=crHourGlass;
    Reset(SysFile,1);
    if FileSize(SysFile) > 0 then Begin
    with SysData do Begin
    NumR:=SizeOf(Word);
    While (NumR=SizeOf(Word)) do Begin
    S:='';  par:=0;
    While (par <> 64531) do begin
      BlockRead(SysFile,par,SizeOf(Word),NumR);
      if NumR=SizeOf(Word) then begin
        if par <> 64531 then begin
          Ch:=WordToChar(par,StrToInt(Copy(regNumber,Length(regNumber)-2,3)));
          s1:=Ch;
          S:=Format('%s%s',[S,s1]);
          end;
        end else begin
        par:=64531; S:='';
        end;
      end;
    if S<>'' then Add(S);
    End;
    J00:=IndexOf(StrSysDevice);
    if J00=-1 then Result:=false
      else begin
      st:=SysData.Strings[J00+1];
      if Copy(st,Length(st)-2,3)<>regNumber then Result:=false
      else regNumber:=st;
      end
    End;
    End else Result:=false;
    except
    Result:=false;
    end;
    finally
    System.CloseFile(SysFile);
    Screen.Cursor:=crDefault;
    end;
  end;

  //*********************************************
  function RealTo_MyStr(Par:Real; W,D:Byte):String;
  const
  s0='0000000000';
  var
    S1,S2:String;
  begin
    str(Par:W:D,S1);
    S2:=Format('%s%s',[S1,S0]);
    Result:=Copy(S2,1,10);
  end;

  //*******************************************************
  function GetSysDataStr1:String;
  var
    S,S1:String;
    par:Real;
    i:byte;
  begin
```

```
  S:='';
  par:=UGI_Fkg/1.e6;
  S1:=RealTo_MyStr(par,6,3);
  S:=Format('%s%s',[S,S1]);
  for i:=0 to 3 do begin
    S1:=RealTo_MyStr(UGI_Dela[i],7,1);
    S:=Format('%s%s',[S,S1]);
    end;
  for i:=0 to 3 do begin
    S1:=RealTo_MyStr(RastMM[i],8,4);
    S:=Format('%s%s',[S,S1]);
    end;
  S1:=RealTo_MyStr(Tcalibr,8,4);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Tcoef,8,4);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_hor,6,1);
  S:=Format('%s%s',[S,S1]);
  Result:=S;
end;

//*****************************************************
function GetSysDataStr2:String;
var
  S,S1:String;
begin
  S:='';
  S1:=RealTo_MyStr(P_P0,7,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(P_x0,9,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(P_ln,10,7);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(P_T0,6,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(PT_ln,9,6);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(PT_kv,9,6);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(PT_kb,9,6);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(P_delta,7,2);
  S:=Format('%s%s',[S,S1]);
  Result:=S;
end;

//*********************************************
function GetSysDataStr3:String;
var
  S,S1:String;
  par:Real;
begin
  S:='';
  S1:=RealTo_MyStr(r_r0,7,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(r_X0,9,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(r_ln,10,7);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(r_kv,10,6);
```

```
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(r_kb,10,7);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(r_delta,7,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(Tr_T0,6,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(Tr_x0,9,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(Tr_ln,10,7);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(Tr_kv,10,6);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(Tr_delta,6,2);
  S:=Format('%s%s',[S,S1]);
  par:=IsDCB_15;
  S1:=RealTo_MyStr(par,3,1);
  S:=Format('%s%s',[S,S1]);
  Result:=S;
end;

//*****************************************
function GetSysDataStr4:String;
var
  S,S1:String;
begin
  S:='';
  S1:=RealTo_MyStr(G_x0,6,4);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(G_y0,6,4);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(G_xd,6,4);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(G_yd,6,4);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(G_hor,6,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(G_xdelta,6,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(G_ydelta,6,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Z,5,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(SetMinVolt,4,1);
  S:=Format('%s%s',[S,S1]);
  Result:=S;
end;

//*****************************************
function GetSysDataStr5:String;
var
  S,S1:String;
begin
  S:='';
  S1:=RealTo_MyStr(UGI_Fiz,7,2);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Az0,10,5);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Az1,10,5);
  S:=Format('%s%s',[S,S1]);
```

```
  S1:=RealTo_MyStr(UGI_Az2,10,5);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Az3,10,5);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Az4,10,5);
  S:=Format('%s%s',[S,S1]);
  S1:=RealTo_MyStr(UGI_Coefz,7,3);
  S:=Format('%s%s',[S,S1]);
  Result:=S;
end;

//*****************************************************
function MyStr_ToReal(const MyStr:String; var D1:Real):Boolean;
var
  code:Integer;
begin
  try
  Val(MyStr,D1,code);
  if code = 0 then Result:=true
            else Result:=false;
  except
  raise
  end;
end;

//**************************************************
function ReadSysDataStr1(const S:String):Boolean;
var
  S1:String;
  par:Real;
begin
  try
  Result:=true;
  S1:=Copy(S,1,10);
  if MyStr_ToReal(S1,par) then UGI_Fkg:=round(1.e6*par)
                    else Result:=false;
  S1:=Copy(S,11,10);
  if MyStr_ToReal(S1,par) then UGI_Dela[0]:=round(par)
                    else Result:=false;
  S1:=Copy(S,21,10);
  if MyStr_ToReal(S1,par) then UGI_Dela[1]:=round(par)
                    else Result:=false;
  S1:=Copy(S,31,10);
  if MyStr_ToReal(S1,par) then UGI_Dela[2]:=round(par)
                    else Result:=false;
  S1:=Copy(S,41,10);
  if MyStr_ToReal(S1,par) then UGI_Dela[3]:=round(par)
                    else Result:=false;
  S1:=Copy(S,51,10);
  if MyStr_ToReal(S1,par) then RastMM[0]:=par
                    else Result:=false;
  S1:=Copy(S,61,10);
  if MyStr_ToReal(S1,par) then RastMM[1]:=par
                    else Result:=false;
  S1:=Copy(S,71,10);
  if MyStr_ToReal(S1,par) then RastMM[2]:=par
                    else Result:=false;
  S1:=Copy(S,81,10);
  if MyStr_ToReal(S1,par) then RastMM[3]:=par
                    else Result:=false;
```

```
  S1:=Copy(S,91,10);
  if MyStr_ToReal(S1,par) then Tcalibr:=par
                  else Result:=false;
  S1:=Copy(S,101,10);
  if MyStr_ToReal(S1,par) then UGI_Tcoef:=par
                  else Result:=false;
  S1:=Copy(S,111,10);
  if MyStr_ToReal(S1,par) then UGI_hor:=par
                  else Result:=false;
  except
  raise
  end;
end;

//*****************************************************
function ReadSysDataStr2(const S:String):Boolean;
var
  S1:String;
  par:Real;
begin
  try
  Result:=true;
  S1:=Copy(S,1,10);
  if MyStr_ToReal(S1,par) then P_P0:=par
                  else Result:=false;
  S1:=Copy(S,11,10);
  if MyStr_ToReal(S1,par) then P_x0:=par
                  else Result:=false;
  S1:=Copy(S,21,10);
  if MyStr_ToReal(S1,par) then P_ln:=par
                  else Result:=false;
  S1:=Copy(S,31,10);
  if MyStr_ToReal(S1,par) then P_T0:=par
                  else Result:=false;
  S1:=Copy(S,41,10);
  if MyStr_ToReal(S1,par) then PT_ln:=par
                  else Result:=false;
  S1:=Copy(S,51,10);
  if MyStr_ToReal(S1,par) then PT_kv:=par
                  else Result:=false;
  S1:=Copy(S,61,10);
  if MyStr_ToReal(S1,par) then PT_kb:=par
                  else Result:=false;
  S1:=Copy(S,71,10);
  if MyStr_ToReal(S1,par) then P_delta:=par
                  else Result:=false;
  except
  raise
  end;
end;

//*********************************************************
function ReadSysDataStr3(const S:String):Boolean;
var
  S1:String;
  par:Real;
begin
  try
  Result:=true;
  S1:=Copy(S,1,10);
```

```
  if MyStr_ToReal(S1,par) then r_r0:=par
                    else Result:=false;
  S1:=Copy(S,11,10);
  if MyStr_ToReal(S1,par) then r_X0:=par
                    else Result:=false;
  S1:=Copy(S,21,10);
  if MyStr_ToReal(S1,par) then r_ln:=par
                    else Result:=false;
  S1:=Copy(S,31,10);
  if MyStr_ToReal(S1,par) then r_kv:=par
                    else Result:=false;
  S1:=Copy(S,41,10);
  if MyStr_ToReal(S1,par) then r_kb:=par
                    else Result:=false;
  S1:=Copy(S,51,10);
  if MyStr_ToReal(S1,par) then r_delta:=par
                    else Result:=false;
  S1:=Copy(S,61,10);
  if MyStr_ToReal(S1,par) then Tr_T0:=par
                    else Result:=false;
  S1:=Copy(S,71,10);
  if MyStr_ToReal(S1,par) then Tr_x0:=par
                    else Result:=false;
  S1:=Copy(S,81,10);
  if MyStr_ToReal(S1,par) then Tr_ln:=par
                    else Result:=false;
  S1:=Copy(S,91,10);
  if MyStr_ToReal(S1,par) then Tr_kv:=par
                    else Result:=false;
  S1:=Copy(S,101,10);
  if MyStr_ToReal(S1,par) then Tr_delta:=par
                    else Result:=false;
  S1:=Copy(S,111,10);
  if MyStr_ToReal(S1,par) then IsDCB_15:=round(par)
                    else Result:=false;
  except
  raise
  end;
end;

//*****************************************************
function ReadSysDataStr4(const S:String):Boolean;
var
  S1:String;
  par:Real;
begin
  try
  Result:=true;
  S1:=Copy(S,1,10);
  if MyStr_ToReal(S1,par) then G_x0:=par
                    else Result:=false;
  S1:=Copy(S,11,10);
  if MyStr_ToReal(S1,par) then G_y0:=par
                    else Result:=false;
  S1:=Copy(S,21,10);
  if MyStr_ToReal(S1,par) then G_xd:=par
                    else Result:=false;
  S1:=Copy(S,31,10);
  if MyStr_ToReal(S1,par) then G_yd:=par
                    else Result:=false;
```

```
  S1:=Copy(S,41,10);
  if MyStr_ToReal(S1,par) then G_hor:=par
                  else Result:=false;
  S1:=Copy(S,51,10);
  if MyStr_ToReal(S1,par) then G_xdelta:=par
                  else Result:=false;
  S1:=Copy(S,61,10);
  if MyStr_ToReal(S1,par) then G_ydelta:=par
                  else Result:=false;
  S1:=Copy(S,71,10);
  if MyStr_ToReal(S1,par) then UGI_Z:=par
                  else Result:=false;
  S1:=Copy(S,81,10);
  if MyStr_ToReal(S1,par) then SetMinVolt:=par
                  else Result:=false;
  except
  raise
  end;
end;

//*****************************************************
function ReadSysDataStr5(const S:String):Boolean;
var
  S1:String;
  par:Real;
begin
  try
  Result:=true;
  S1:=Copy(S,1,10);
  if MyStr_ToReal(S1,par) then UGI_Fiz:=par
                  else Result:=false;
  S1:=Copy(S,11,10);
  if MyStr_ToReal(S1,par) then UGI_Az0:=par
                  else Result:=false;
  S1:=Copy(S,21,10);
  if MyStr_ToReal(S1,par) then UGI_Az1:=par
                  else Result:=false;
  S1:=Copy(S,31,10);
  if MyStr_ToReal(S1,par) then UGI_Az2:=par
                  else Result:=false;
  S1:=Copy(S,41,10);
  if MyStr_ToReal(S1,par) then UGI_Az3:=par
                  else Result:=false;
  S1:=Copy(S,51,10);
  if MyStr_ToReal(S1,par) then UGI_Az4:=par
                  else Result:=false;
  S1:=Copy(S,61,10);
  if MyStr_ToReal(S1,par) then UGI_Coefz:=par
                  else Result:=false;
  except
  raise
  end;
end;

//***********************************************
function ReadSysData:Boolean;
var
  J0:Integer;
  k1:array[0..4] of Integer;
begin
```

```
  try
  with SysData do Begin
  st_FullNamePr:=Copy(Strings[0],1,27);
  st_BriefNamePr:=Copy(Strings[0],28,11);
  Par_Standart:=Copy(Strings[1],1,12);
  Par_System:=Copy(Strings[1],13,12);
  st_Copyr:=Strings[2];
  J0:=SysData.IndexOf(StrSysDevice);
  if J0<>-1 then Result:=true else Result:=false;
  if J0<>-1 then Begin
  if ReadSysDataStr1(Strings[J0+2]) then k1[0]:=1 else k1[0]:=0;
  if ReadSysDataStr2(Strings[J0+3]) then k1[1]:=1 else k1[1]:=0;
  if ReadSysDataStr3(Strings[J0+4]) then k1[2]:=1 else k1[2]:=0;
  if ReadSysDataStr4(Strings[J0+5]) then k1[3]:=1 else k1[3]:=0;
  if ReadSysDataStr5(Strings[J0+6]) then k1[4]:=1 else k1[4]:=0;
  J0:=k1[0]*k1[1]*k1[2]*k1[3]*k1[4];
  if J0=1 then Result:=true else Result:=false;
  End; End;
  except
  raise
  end;
end;

//*******************************************
function GetDataAMKsys:Boolean;
begin
  try
  Result:=false;
  if GetParamProgram then Begin
    AssignFile(SysFile,SysFileName);
    SysData:=TStringList.Create;
    if FileExists(SysFileName) then Begin
     if not ReadSysFile then GetMyErrorDlg(Msg_File_0+SysFileName+Msg_File_Format+NameProgram)
     else begin
       if not ReadSysData then GetMyErrorDlg(Msg_Modify+SysFileName)
       else Result:=true;
       end;
    End else GetMyErrorDlg(Format('%s%s%s',[Msg_File_0,SysFileName,Msg_File_NoFind]));
  End;
  except
  raise
  end;
end;

//****************************************
procedure UpDateAMKsys;
var
  J0:Integer;
begin
  with SysData do Begin
  J0:=SysData.IndexOf(StrSysDevice);
  if J0<>-1 then Begin
  Delete(J0+2);
  Insert(J0+2,GetSysDataStr1);
  Delete(J0+3);
  Insert(J0+3,GetSysDataStr2);
  Delete(J0+4);
  Insert(J0+4,GetSysDataStr3);
  Delete(J0+5);
  Insert(J0+5,GetSysDataStr4);
```

```
  WriteSysFile;
  End;
  End;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT UnAMKport;
interface
uses
  WinProcs, Classes, Controls, Forms, Dialogs;

const
 CountByteData_UGI = 23;
 CountConstant_PPU=53;

 MsgErr1 =  'Не совпадает контрольная сумма байт.';
 MsgErr4 =  'Ультразвуковые данные блока УГИ вне диапазона разрешенных значений.';
 MsgErr5 =  'Отсутствуют данные об атмосферном давлении.';
 MsgErr6 =  'Отсутствуют данные о влажности воздуха.';
 MsgErr7 =  'Отсутствуют данные об угле наклона метеомачты.';
 MsgErr8 =  'Отсутствуют данные о температуре из датчика влажности.';
 MsgErr9_0 =  'Нет связи с устройством.';
 MsgUGIErr9_0 =  'Отсутствует связь с блоком УГИ - не поступают его данные.';
 MsgErr9_1 =  ' Проверьте его включение и/или подключение к компьютеру.';
 MsgErr101 = 'Принимаемые данные по формату не соответствуют устройству. ';
 MsgErr10_1='Возможно неверно установлены параметры порта или данные искажаются из-за значительных помех в
линии связи.';
 MsgErr11 = 'Задержка поступления данных из устройства. ';

 MsgPPUErr_Kod = 'Код ошибки=';
 MsgPPUErr0 = ' Команда выполнена успешно.';
 MsgPPUErr1 = ' В ответе не совпадает контрольная сумма байт.';
 MsgPPUErr2 = ' Исполнение команды отвергнуто.';
 MsgPPUErr3 = ' Ответ не соответствует операции с градуировочным параметром заданного номера.';
 MsgPPUErr4 = ' Ответ не совпадает по номеру с выданной ему командой.';
 MsgPPUErr5 = ' В ответе нарушена структура информационного пакета байтов.';
 MsgPPUErr9_0 =  'Нет связи с блоком ВиИ (пультом).';

 MsgPPU_999 = 'Сообщение блока ВиИ "Отсутствует связь с блоком УГИ" - не поступают его данные.';
 MsgPPU_998 = 'Сообщение блока ВиИ "С момента начала измерений прошло времени меньше запрошенного для
усреднения".';
 MsgPPU_997 = 'Сообщение блока ВиИ "Первичные данные ультразвукового термоанемометра вне диапазона раз-
решенных значений".';

var
  Handle_AMK:THandle;
  OverLap_AMK:tOverLapped;
  IsOpenCOMport:Boolean;
  InQue:Cardinal;
  BufData_UGI: array of Byte;

procedure Close_AMKport(var Handle1:THandle; var IsOpen1:Boolean);far; external 'AMKport.DLL';
procedure ClearRxBuf_AMKport(Handle1:THandle; var InQue1:Cardinal);far; external 'AMKport.DLL';
function ReadUGI_Data(Handle1:THandle; Count1:Byte; var Buf1:array of Byte;
                var OverLap1:tOverLapped):Boolean;far; external 'AMKport.DLL';
procedure SetPPU_inWork(Handle1:THandle; var OverLap1:tOverLapped);far; external 'AMKport.DLL';
```

```
function GetPPU_MeteoParam(Handle1:THandle;TypeMeteo1,Time1:Byte; var Value1:Single;
                    var OverLap1:tOverLapped):Byte;far; external 'AMKport.DLL';
function ReadPPU_OneConstant(Handle1:THandle; Num1:Byte; var Value1:Single;
                     var OverLap1:tOverLapped):Byte;far; external 'AMKport.DLL';
function WritePPU_OneConstant(Handle1:THandle; Num1:Byte; var Value1:Single;
                      var OverLap1:tOverLapped):Byte;far; external 'AMKport.DLL';
function ReadPPU_ks(Handle1:THandle; Count1:Word; var ks1:Word;
                var OverLap1:tOverLapped):Byte;far; external 'AMKport.DLL';
function WritePPU_ks(Handle1:THandle; Count1:Word; var ks1:Word;
                 var OverLap1:tOverLapped):Byte;far; external 'AMKport.DLL';
procedure StartAMKport_UGI;
function StartAMKport_PPU(Rate1:DWord; ShowMSG1:Boolean):Byte;
function  GetStrRepOtkazPPU(Value1:Single):String;

implementation
uses UnAMKini;

procedure Open_AMKport(Num1:Byte; Rate1,RxBuf1,TxBuf1:DWord; var Handle1:THandle;
                  var IsOpen1:Boolean);far; external 'AMKport.DLL';
procedure SetPPU_inCalibr(Handle1:THandle; var OverLap1:tOverLapped);far; external 'AMKport.DLL';

//***********************************
procedure StartAMKport_UGI;
label 1;
var
  Start1,Stop1,Chec:Longint;
  Errors:DWord;
  FlagPPU:Boolean;
  COM_Stat:tCOMStat;
begin
  try
  Open_AMKport(COM_Number,COM_BaudRate,COM_SizeRxBuf,COM_SizeTxBuf,Handle_AMK,IsOpenCOMport);
  if not IsOpenCOMport then Exit;
  Screen.Cursor:=crHourGlass;
  SetLength(BufData_UGI,CountByteData_UGI+1);
  FlagPPU:=false;
1:ClearRxBuf_AMKport(Handle_AMK,InQue);
  Start1:=GetTickCount;
  Stop1:=Start1;
  while Stop1 < (Start1+1000) do Stop1:=GetTickCount;
  ClearCommError(Handle_AMK,Errors,@COM_Stat);
  if COM_Stat.cbInQue >=3 then begin
    Chec:=0;
    while COM_Stat.cbInQue div 3 >1 do begin
      if ReadUGI_Data(Handle_AMK,CountByteData_UGI+1,BufData_UGI,OverLap_AMK) then Inc(Chec);
      ClearCommError(Handle_AMK,Errors,@COM_Stat);
      end;{while}
    if Chec=0 then begin
      ShowMessage(MsgErr101+MsgErr10_1);
      Close_AMKport(Handle_AMK,IsOpenCOMport);
      end;
    end else begin
    if not FlagPPU then begin
      SetPPU_inCalibr(Handle_AMK,OverLap_AMK);
      FlagPPU:=true;
      GOTO 1;
      end else begin
      ShowMessage(MsgUGIErr9_0+MsgErr9_1);
      Close_AMKport(Handle_AMK,IsOpenCOMport);
      BufData_UGI:=nil;
```

```
      end;
    end;
  finally
  Screen.Cursor:=crDefault;
  end;
end;

//************************************
function StartAMKport_PPU(Rate1:DWord; ShowMSG1:Boolean):Byte;
var
  Value1:Single;
begin
  try
  Result:=9;
  Open_AMKport(COM_Number,Rate1,COM_SizeRxBuf,COM_SizeTxBuf,Handle_AMK,IsOpenCOMport);
  if not IsOpenCOMport then Exit;
  SetPPU_inWork(Handle_AMK,OverLap_AMK);
  Screen.Cursor:=crHourGlass;
  Result:=ReadPPU_OneConstant(Handle_AMK,1,Value1,OverLap_AMK);
  if Result>4 then begin
    if ShowMSG1 then ShowMessage(MsgPPUErr9_0+MsgErr9_1);
    Close_AMKport(Handle_AMK,IsOpenCOMport);
    end;
  finally
  Screen.Cursor:=crDefault;
  end;
end;

//************************************}
function GetStrRepOtkazPPU(Value1:Single):String;
var
  Kod1:Integer;
begin
  Kod1:=round(Value1);
case Kod1 of
  -999:Result:=MsgPPU_999;
  -998:Result:=MsgPPU_998;
  -997:Result:=MsgPPU_997;
  end;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT UnData;
interface
uses
  Windows, SysUtils, UnAMKport;

const
  MaxLengthBufBinFile = 2048;
  CountParam_UGI = CountByteData_UGI div 2;

type
  TData_UGI = array[0..CountParam_UGI] of Word;
  TInputData = array[0..CountParam_UGI] of Real;

  TDataRecord = array [0..8] of Real;
```

```
  TLineRec = record
    mas:TDataRecord;
    Sboj:Byte;
    TDr:TDateTime;
  end;
  TGlobData = array of TLineRec;

var
  Data_UGI,OldData_UGI,DelSrData_UGI: TData_UGI;
  InputData: TInputData;
  Met_Data,Cal_Data,Con_Data,Graf_Data:TGlobData;
  Met_SrVar,Met_Sred:TDataRecord;
  Cal_SrVar,Cal_Sred:TDataRecord;
  Con_SrVar,Con_Sred:TDataRecord;
  isYesAllGlobData:Boolean;
  NumPosGlobData:Integer;
  CountInData,CountCenzu,SbojCenzu:Integer;
  MasFiltr1:array [0..3,0..99] of Word;
  CountNumFiltr1:array [0..3] of Byte;
  FileNameBinF:String;
  Handle_BinF,CountBinDataWriten:Integer;
  BufBinFile:Array of Byte;
  CheckTimeCreateBinF,CheckTimeSaveBinF:TDateTime;

procedure GetInputData;
procedure ClearGlobData(var Data1:TGlobData);
function ValidGlobData(const Data1:TGlobData):Integer;
procedure AddRecGlobData(Mas1:TDataRecord; Sboj1:Byte;TD1:TDateTime;Usredn1:Integer;
        var Data1:TGlobData; var SrVar1:TDataRecord);
function GetSredGlobData(var Data1:TGlobData; var Sred1:TDataRecord):Boolean;
procedure CalcHorWind(Vx1,Vy1:Real; var Vg1,Fi1:Real);far; external 'Meteo.DLL';
function GetStrDirWindDYg(Dir1:Real):ShortString;far; external 'Meteo.DLL';
procedure StartCenzuir;
procedure MakeCikleCenzuir(Fkg:Integer);
procedure FiltrateGlobData(var Data1:TGlobData);
procedure FiltrateInputDataUgi;
procedure Filtr1AddNumData(Num:Byte; A1:Word; Min1,Max1,MaxMasFIn1:Integer);
procedure GetNewBinFile;
procedure Close_BinF;
procedure WriteFromBufToBinF;
procedure AddDataBufBin(er1:Byte; TC1,Vx1,Vy1,Vz1,MMdavl1,Otvlag1:Real);

implementation
uses UnAMKini, UnAMKsys, ApllDlgs;

procedure Filtr2Execute(N:Integer; var X:array of Real);far; external 'Meteo.DLL';
procedure SortirArray(N:Integer; var X:array of Word);far; external 'Meteo.DLL';

//********************************
procedure GetInputData;
const
  MaxCountPrTrData=16;
  MaxCountGxGy=9;
var
  i:Integer;
begin
  for i:=0 to 4 do
    InputData[i]:=Data_Ugi[i];
  if Data_UGI[0]>=5 then Exit;
    Inc(CountInData);
```

```
    for i:=5 to 6 do begin
      if CountInData<MaxCountPrTrData then begin
       if CountInData<=1 then InputData[i]:=Data_Ugi[i]
       else InputData[i]:=(Data_Ugi[i]+CountInData*InputData[i])/(CountInData+1);
    end else
    InputData[i]:=(Data_Ugi[i]+MaxCountPrTrData*InputData[i])/(MaxCountPrTrData+1);
    end;
  for i:=7 to CountParam_UGI do begin
      if CountInData<MaxCountGxGy then begin
       if CountInData<=1 then InputData[i]:=Data_Ugi[i]
       else InputData[i]:=(Data_Ugi[i]+CountInData*InputData[i])/(CountInData+1);
    end else
    InputData[i]:=(Data_Ugi[i]+MaxCountGxGy*InputData[i])/(MaxCountGxGy+1);
    end;
end;

//****************************************
procedure ClearGlobData(var Data1:TGlobData);
begin
  try
  if Data1<>nil then Data1:=nil;
  if Graf_Data<>nil then Graf_Data:=nil;
  isYesAllGlobData:=false;
  NumPosGlobData:=0;
  except
  raise
  end;
end;

//****************************************
function ValidGlobData(const Data1:TGlobData):Integer;
var
  i,Ng:Integer;
begin
  try
  if Data1=nil then Result:=0 else begin
  if Length(Data1)<3 then Result:=0 else begin
  Ng:=0;
  for i:=0 to High(Data1) do
    if Data1[i].Sboj<=4 then Inc(Ng);
  Result:=round(Ng/Length(Data1)*100);
  end;
  end;
  except
  raise
  end;
end;

//****************************************
procedure AddRecGlobData(Mas1:TDataRecord; Sboj1:Byte;TD1:TDateTime;Usredn1:Integer;
        var Data1:TGlobData; var SrVar1:TDataRecord);
var
  NewRec:TLineRec;
  i,L1:Integer;
begin
  try
  with NewRec do begin
  for i:=0 to 8 do
    mas[i]:=mas1[i];
  TDr:=TD1;
```

```
    Sboj:=Sboj1;
    end; {with NewRec^}
    SetLength(Graf_Data,Length(Graf_Data)+1);
    Graf_Data[High(Graf_Data)]:=NewRec;
    if Length(Graf_Data)>round(15/TimeCikle)+2 then Graf_Data:=nil;
    if Data1=nil then Begin
      isYesAllGlobData:=false;
      NumPosGlobData:=0;
      SetLength(Data1,1);
      Data1[0]:=NewRec;
      if NewRec.sboj<4 then
        for i:=0 to 8 do
          SrVar1[i]:=NewRec.mas[i];
      End else Begin
        if (not isYesAllGlobData) and
        (NewRec.TDr>Data1[0].TDr+EncodeTime(0,Usredn1 div 60,0,0)) then begin
          isYesAllGlobData:=true;
          NumPosGlobData:=-1;
          end;
        if not isYesAllGlobData then begin
          SetLength(Data1,Length(Data1)+1);
          NumPosGlobData:=High(Data1);
          Data1[NumPosGlobData]:=NewRec;
          end else begin
          if NumPosGlobData<High(Data1) then Inc(NumPosGlobData)
                                  else NumPosGlobData:=0;
          Data1[NumPosGlobData]:=NewRec;
          end;
      L1:=Length(Data1);
      if NewRec.sboj<4 then
      for i:=0 to 8 do
        SrVar1[i]:=(NewRec.mas[i]+L1*SrVar1[i])/(L1+1);
      End; {if Data1=nil}
    except
    raise;
    end;
  end;

  {*****************************************************************************}
  function GetSredGlobData(var Data1:TGlobData; var Sred1:TDataRecord):Boolean;
  const
    P_intMeteo = 2.5;
  var
    i,J,N,K:Integer;
    su1,su2,md,sd,min1,max1:Real;
  begin
    try
    Result:=false;
    if Data1=nil then Exit;
    if Length(Data1)<=1 then begin
      if Data1[High(Data1)].Sboj<=4 then begin
        for i:=0 to 8 do
          Sred1[i]:=Data1[High(Data1)].Mas[i];
        Result:=true;
        end;
      Exit;
      end;
    for i:=0 to 8 do Begin
    N:=0; su1:=0; su2:=0;
    for J:=0 to High(Data1) do Begin
```

```
    if Data1[J].Sboj<=4 then begin
      su1:=su1+Data1[J].Mas[i];
      su2:=su2+sqr(Data1[J].Mas[i]);
      Inc(N);
      end; {if Line.Sboj<=4}
    End; {J}
  if N>0 then md:=su1/N
        else md:=Data1[High(Data1)].Mas[i];
  if N>10 then Begin
  sd:=sqrt(abs((su2/N-sqr(md))*N/(N-1)))*P_intMeteo+0.01;
  Min1:=md-sd;
  Max1:=md+sd;
  K:=0; su1:=0;
  for J:=0 to High(Data1) do Begin
   if Data1[J].Sboj<=4 then begin
   if not ((Data1[J].Mas[i]<Min1) or (Data1[J].Mas[i]>Max1)) then begin
     su1:=su1+Data1[J].Mas[i];
     Inc(K);
     end; {if}
    end; {if Line.Sboj<=4}
   End; {j}
  if K>0 then md:=su1/K;
  End;{if N>10}
  Sred1[i]:=md;
  End; {i}
  Result:=true;
  except;
 raise;
 end;
end;

//************************************
procedure StartCenzuir;
begin
  CountCenzu:=0;
end;

//************************************
procedure MakeCikleCenzuir(Fkg:Integer);
const
  UGI_Fs = 100000;
var
  DelVar:Array [1..4] of Integer;
  i,Col,Period:Integer;
  FlagOut:Boolean;
begin
  if Data_UGI[0]>=4 then Exit;
  if CountCenzu=0 then SbojCenzu:=0;
  if SbojCenzu > VolumeCenzuir then begin
     CountCenzu:=0;
     SbojCenzu:=0;
     end;
  if (CountCenzu=0) and (Data_UGI[0]<=1) then begin
     for i:=1 to 4 do
        OldData_UGI[i]:=Data_Ugi[i];
     Inc(CountCenzu);
     Exit;
  end;
  FlagOut:=false;
  Period:=round(0.85*Fkg/UGI_Fs);
```

```
  for i:=1 to 4 do begin
    Delvar[i]:=Abs(Data_Ugi[i]-OldData_UGI[i])+1;
    if Delvar[i]>Period then begin
      Data_UGI[i]:=OldData_UGI[i];
      Data_UGI[0]:=4;
      FlagOut:=true;
      end;
    end;
  if not FlagOut then SbojCenzu:=0
  else begin
    Inc(SbojCenzu);
    Exit;
    end;
  if CountCenzu-1<VolumeCenzuir then Col:=CountCenzu-1
                          else Col:=VolumeCenzuir;
  if Col<VolumeCenzuir then begin
     if Col=0 then begin
       for i:=1 to 4 do begin
         DelSrData_UGI[i]:=DelVar[i];
         OldData_UGI[i]:=Data_Ugi[i];
         end
       end else begin
       for i:=1 to 4 do begin
         DelSrData_UGI[i]:=round((DelVar[i]+Col*DelSrData_UGI[i])/(Col+1));
         OldData_UGI[i]:=Data_Ugi[i];
         end;
       end; {if Col}
     Inc(CountCenzu);
     end else begin
     for i:=1 to 4 do
     if DelVar[i] div DelSrData_UGI[i] > BoundsCenzuir then Data_UGI[0]:=4;
     if Data_UGI[0]<4 then begin
       for i:=1 to 4 do begin
         DelSrData_UGI[i]:=round((DelVar[i]+Col*DelSrData_UGI[i])/(Col+1));
         OldData_UGI[i]:=Data_Ugi[i];
         end;
       Inc(CountCenzu);
       end;
     end;
end;

//**************************************************
procedure FiltrateGlobData(var Data1:TGlobData);
var
  X:array of Real;
  i,j,N:Integer;
begin
  try
  if Data1=nil then Exit;
  N:=Length(Data1);
  SetLength(X,N);
  for i:=0 to 8 do Begin
  for J:=0 to High(Data1) do Begin
    X[J]:=Data1[J].Mas[i];
    End; {j}
  Filtr2Execute(N,X);
  for J:=0 to High(Data1) do Begin
    Data1[J].Mas[i]:=X[J];
    End; {j}
  End;{i}
```

```
  finally
  X:=nil;
  end;
end;

//*********************************************************
procedure Filtr1AddNumData(Num:Byte; A1:Word; Min1,Max1,MaxMasFIn1:Integer);
var
  j:Byte;
begin
  if (A1 < min1) or (A1 > max1)  then Exit;
  if CountNumFiltr1[Num]<MaxMasFIn1 then begin
  MasFiltr1[Num,CountNumFiltr1[Num]]:=A1;
  Inc(CountNumFiltr1[Num]);
  end else begin
  for j:=1 to MaxMasFIn1-1 do
  MasFiltr1[Num,j-1]:=MasFiltr1[Num,j];
  MasFiltr1[Num,MaxMasFIn1-1]:=A1;
  end;
end;

//*******************************************
function Filtr1GetNumData(Num:Byte; var A1:Word):Boolean;
var
  X:array of Word;
  j:Integer;
begin
  Result:=false;
  if CountNumFiltr1[Num]<3 then Exit;
  try
  SetLength(X,CountNumFiltr1[Num]);
  for j:=0 to CountNumFiltr1[Num]-1 do
    X[j]:=MasFiltr1[Num,j];
  SortirArray(CountNumFiltr1[Num],X);
  A1:=X[CountNumFiltr1[Num] div 2];
  Result:=true;
  finally
  X:=nil;
  end;
end;

//**********************************
procedure FiltrateInputDataUgi;
var
  k:Byte;
  A1:Word;
begin
  for k:=0 to 3 do begin
  if Filtr1GetNumData(k,A1) then begin
  if abs(Data_Ugi[k+1]-A1)>IntWinFiltrInput then begin
  Data_Ugi[k+1]:=MasFiltr1[k,CountNumFiltr1[k]-1];
  Data_Ugi[0]:=1;
  end else
  Filtr1AddNumData(k,Data_Ugi[k+1],UGI_Amin,UGI_Amax,MaxMasFiltrInput); {if abs}
  end else
  Filtr1AddNumData(k,Data_Ugi[k+1],UGI_Amin,UGI_Amax,MaxMasFiltrInput); {if Filtr1GetNumData(k,A1)}
  end; {k}
end;

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

```
procedure AddByteBufBin(b1:Byte);
begin
  try
  SetLength(BufBinFile,Length(BufBinFile)+1);
  BufBinFile[High(BufBinFile)]:=b1;
  except
  raise
  end;
end;

{*********************************}
procedure AddInt2BufBin(X1:SmallInt);
var
  IS1:array [0..1] of Byte ABSOLUTE X1;
begin
  try
  AddByteBufBin(IS1[0]);
  AddByteBufBin(IS1[1]);
  except
  raise
  end;
end;

{*******************************}
procedure AddDataBufBin(er1:Byte; TC1,Vx1,Vy1,Vz1,MMdavl1,Otvlag1:Real);
var
  a1:SmallInt;
begin
  try
  if abs(TC1) < 327.67 then a1:=round(100*TC1)
                  else a1:=32767;
  AddInt2BufBin(a1);
  if abs(Vx1) < 327.67 then a1:=round(100*Vx1)
                  else a1:=32767;
  AddInt2BufBin(a1);
  if abs(Vy1) < 327.67 then a1:=round(100*Vy1)
                  else a1:=32767;
  AddInt2BufBin(a1);
  if abs(Vz1) < 327.67 then a1:=round(100*Vz1)
                  else a1:=32767;
  AddInt2BufBin(a1);
  if abs(MMdavl1) < 3276.7 then a1:=round(10*MMdavl1)
                    else a1:=32767;
  AddInt2BufBin(a1);
  if abs(Otvlag1) < 327.67 then a1:=round(100*Otvlag1)
                    else a1:=32767;
  AddInt2BufBin(a1);
  AddByteBufBin(er1);
  Inc(CountBinDataWriten,13);
  except
  raise
  end;
end;

{************************************}
procedure WriteFromBufToBinF;
var
  i,j,m:Integer;
  mas:array[0..12] of Byte;
begin
```

```
  try
  if BufBinFile=nil then Exit;
  m:=Length(BufBinFile) div 13;
  if m<=0 then Exit;
  Handle_BinF:=FileOpen(FileNameBinF,fmOpenWrite);
  FileSeek(Handle_BinF,0,2);
  for j:=0 to m-1 do begin
    for i:=0 to 12 do
      mas[i]:=BufBinFile[13*j+i];
    FileWrite(Handle_BinF,mas,13);
    end; {for j}
  FlushFileBuffers(Handle_BinF);
  BufBinFile:=nil;
  finally
  FileClose(Handle_BinF);
  end;
end;

{*********************************}
procedure WriteDateTimeToBinF(DT1:tDateTime);
var
  a1,a2,a3,a4:Word;
  b1:array [0..1] of Byte ABSOLUTE a1;
  mas:array[0..13] of Byte;
begin
  try
  DecodeDate(Dt1,a1,a2,a3);
  mas[0]:=b1[0]; mas[1]:=b1[1];
  a1:=a2; mas[2]:=b1[0]; mas[3]:=b1[1];
  a1:=a3; mas[4]:=b1[0]; mas[5]:=b1[1];
  DecodeTime(Dt1,a1,a2,a3,a4);
  mas[6]:=b1[0]; mas[7]:=b1[1];
  a1:=a2; mas[8]:=b1[0]; mas[9]:=b1[1];
  a1:=a3; mas[10]:=b1[0]; mas[11]:=b1[1];
  a1:=a4; mas[12]:=b1[0]; mas[13]:=b1[1];
  FileWrite(Handle_BinF,mas,14);
  except
  raise
  end;
end;

{***********************************}
procedure Close_BinF;
begin
  try
  Handle_BinF:=FileOpen(FileNameBinF,fmOpenWrite);
  FileSeek(Handle_BinF,0,2);
  WriteDateTimeToBinF(Now);
  FlushFileBuffers(Handle_BinF);
  FileClose(Handle_BinF);
  if BufBinFile<>nil then BufBinFile:=nil;
  except
  raise
  end;
end;

{*********************************}
procedure GetNewBinFile;
Label 1;
var
```

```
  FileName1:String;
  a1:Word;
  b2:array [0..1] of Byte ABSOLUTE a1;
begin
  if not IsSaveBinFile then Exit;
  try
  FileName1:=GetNameFileDef(Now);
  FileName1:=Format('%s%s%s%s',[DataDir,'\',FileName1,'B']);
1:if FileExists(FileName1) then begin
    FileName1:=FileName1+'A';
    GOTO 1;
    end;
  FileNameBinF:=FileName1;
  Handle_BinF:=FileCreate(FileNameBinF);
  FileSeek(Handle_BinF,0,0);
  CheckTimeCreateBinF:=Now;
  WriteDateTimeToBinF(CheckTimeCreateBinF);
  if UGI_Z < 655.35 then a1:=round(100*UGI_Z)
                 else a1:=65535;
  FileWrite(Handle_BinF,b2[0],1);
  FileWrite(Handle_BinF,b2[1],1);
  FileWrite(Handle_BinF,TypeSurface,1);
  FlushFileBuffers(Handle_BinF);
  CheckTimeSaveBinF:=CheckTimeCreateBinF+EncodeTime(0,0,30,0);
  CountBinDataWriten:=17;
  finally
  FileClose(Handle_BinF);
  end;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT UnMeteo;
interface
uses
  SysUtils, Classes;

const
  Title_Data = ' - дата;';
  Title_Time = ' - срок наблюдения;';
  Title_TimeUsred = ' - время усреднения, минуты;';
  Title_TimeCikle = ' - частота измерений, Гц;';
  Title_CountData = ' - количество выполненных циклов измерений;';
  LineTitle_R =   '*********************************';
  LineLider_R =   '*  УЛЬТРАЗВУКОВАЯ МЕТЕОСТАНЦИЯ  *';
  TitleReport_R = '*            Отчет            *';

  LineIch_R =  '_________________________';
  LineI_R =    'I.ПАРАМЕТРЫ ИЗМЕРЕНИЙ:';
  LineI_R2 =   'I.УСЛОВИЯ ИЗМЕРЕНИЙ:';
  LineIIch_R = '_____________________________';
  LineIII_R   = 'МГНОВЕННЫЕ ЗНАЧЕНИЯ:';

type
  TSrednie = record
    T,Vs,Ve,Vh,Fh,Vmod,w,MMd,Otvl,Gx,Gy,Tr:Real;
  end;
```

```
  TSrKvOt = record
    T,Vs,Ve,Vh,Fh,Vmod,u,v,w,dTC,dVmod:Real;
  end;

  TAsimetr = record
    T,Vs,Ve,Vh,Fh,Vmod,u,v,w:Real;
  end;

  TExcess = record
    T,Vs,Ve,Vh,Fh,Vmod,u,v,w:Real;
  end;

  TKorrel = record
    T_w,T_u,T_v,T_Vh,T_Vmod,w_Vh,u_w,v_w,u_v:Real;
  end;

  TTurbulent = record
    Energ,taux,H,Cd,Vmo,Tmo,Lmo,Ct2,Cv2,Cn2,Cn2opt,gradTN,Hprizem,gradTeta0,
    gradT0,gradU0,Kh,Km,Loturb,Ri,Edesip,Ndesip:Real;
  end;

  TMeteoData = record
      Sred:TSrednie;
      SKO:TSrKvOt;
      Asim:TAsimetr;
      Exc:TExcess;
      KorM:TKorrel;
      Turb:TTurbulent;
      Trmin,Trmax,Tmin,Tmax,Vmin,Vmax,VminSrok,VmaxSrok:Real;
  end;

var
  MeteoData:TMeteoData;

procedure MeteoData_CalcWindMgnoven(Nstop:Integer; var V1:Real);
function  MeteoData_CalcStatistic:Boolean;
function  MeteoData_GetReportStrList(var Data1:TStringList):Boolean;
function  GetMeteoInformStr(var St1:String):Boolean;
function  MeteoData_GetDataStrList(var Data1:TStringList):Boolean;
procedure Calc_EQTrosa(t1,r1:Real; var e1,q1,td1:Real);far; external 'Meteo.DLL';

implementation
uses UnAMKini,ApllDlgs,UnAMKsys,UnData;

const
TitleData_R =   '*      Данные измерений        *';
LineII_R2  = 'II.МЕТЕОРОЛОГИЧЕСКИЕ ПАРАМЕТРЫ:';
LineStch_R = '_____________________________';
LineSt_R   = 'III.СТАТИСТИЧЕСКИЕ ПАРАМЕТРЫ:';
LineTurbch_R = '_____________________________';
LineTurb_R   = 'IV.ПАРАМЕТРЫ ТУРБУЛЕНТНОСТИ:';
LineIIIch_R = '_____________________';
LineIII_R1   = 'III.МГНОВЕННЫЕ ЗНАЧЕНИЯ:';
LineIVch_R = '_____________________________';
LineIV_R   = 'IV.ПРИЗНАКИ КАЧЕСТВА ДАННЫХ:';
LineIIIval= '    N   T     Vs    Ve    w     P     r   err';
St_err0 = 'err = 0 - идеальные данные;';
St_err1 = 'err = 1 - использовалась входная фильтрация первичных данных устройства;';
st_err2 = 'err = 2 - использовалась медианная фильтрация выходных данных программы;';
```

```
st_err3 = 'err = 3 - не совпала контрольная сумма для переданных с устройства байтов';
st_err3d = '         (т.е.существует вероятность сбоя при передаче данных);';
st_err4 = 'err = 4 - первичные данные термоанемометра были вне диапазона разрешенных значений;';
st_err5 = 'err = 5 - не поступили данные датчика атмосферного давления;';
st_err6 = 'err = 6 - не поступили данные датчика влажности воздуха;';
st_err7 = 'err = 7 - не поступили данные датчика угла наклона метеомачты вдоль оси X;';
st_err8 = 'err = 8 - не поступили данные датчика угла наклона метеомачты вдоль оси Y;';
st_erra = 'ПРИ СОБЫТИЯХ err=4,5,6,7 и 8 В НАСТОЯЩИЙ ФАЙЛ ВМЕСТО ОТБРАКОВАННЫХ ДАННЫХ';
st_errb ='ЗАПИСЫВАЮТСЯ ИХ ПОСЛЕДНИЕ ДОСТОВЕРНЫЕ ЗНАЧЕНИЯ.';
Msg_CalcStat = ' Отсутствуют или недостоверные данные! Расчет статистических параметров не возможен. ';
Title_Zmeteo = ' - высота термоанемометра, м;';
Title_SP = ' (тип подстилающей поверхности).';
St_sneg = ' - снег';
St_trava = ' - трава до 0,2 м';
St_rastit = ' - растительность от 0,2 до 1 м';
St_les = ' - лес';
St_pesok = ' - песок';
St_pashnja = ' - пашня';
St_grunt = ' - замороженный грунт';
St_voda = ' - вода, лед';
St_gorod = ' - город';
st_MET_a = '  а) для срока наблюдения';
st_MET_ach='  ____________________________';
st_MET_air=  '                    характеристики воздуха';
st_MET_airch='                    ____________________________';
st_TC15 =     ' <= T - средняя температура, градусы Цельсия;';
st_RMSTC =  ' <= RMS(T) - стандартное отклонение температуры, градусы Цельсия;';
st_MMdavl1 = ' <= P - атмосферное давление, мм.рт.ст.;';
st_MMdavl2 = ' <= P - атмосферное давление, гПа;';
st_rHu =    ' <= r - относительная влажность воздуха, %;';
st_eHu =    ' <= e - упругость (давление) водяного пара, гПа;';
st_EdHu =   ' <= Ed - дефицит влажности, гПа;';
st_TCd =    ' <= Td - температура точки росы, градусы Цельсия;';
st_AbsVlag =' <= q - абсолютная влажность, г/м^3;';
st_moo =    ' <= m - массовая концентрация влаги, о/оо;';
st_Plotnoct = ' <= ro - плотность воздуха, кг/м^3.';
st_MET_wind=  '                    характеристики ветра';
st_MET_windch='                    ____________________';
st_Vh = ' <= V - средняя скорость горизонтального ветра, м/с;';
st_VminSrok = ' <= Vmin - минимальная скорость мгновенного горизонтального ветра, м/с;';
st_VmaxSrok = ' <= Vmax - максимальная скорость мгновенного горизонтального ветра, м/с;';
st_RMSVh = ' <= RMS(V) - стандартное отклонение скорости горизонтального ветра, м/с;';
st_Fh1 = ' <= D - среднее направление горизонтального ветра, градусы;';
st_Fh2 = ' <= D - среднее направление горизонтального ветра, д.у.;';
st_RMSFh = ' <= RMS(D) - стандартное отклонение направления горизонтального ветра, градусы;';
st_wert = ' <= w - средняя скорость вертикального ветра, м/с;';
st_RMSwert = ' <= RMS(w) - стандартное отклонение скорости вертикального ветра, м/с;';
st_Vm = ' <= Vm - модуль среднего вектора скорости ветра, м/с;';
st_Alfa = ' <= Av - угол наклона к горизонту среднего вектора скорости ветра, градусы;';
st_Vx = ' <= Vs - среднее значение южной компоненты горизонтального ветра, м/с;';
st_Vy = ' <= Ve - среднее значение восточной компоненты горизонтального ветра, м/с.';
st_MET_b = '  б) для интервала между сроками наблюдения';
st_MET_bch='  ____________________________________________';
st_TCmin2 = ' <= Trmin - минимальная средняя температура воздуха (ДДВ-12), градусы Цельсия;';
st_TCmax2 = ' <= Trmax - максимальная средняя температура воздуха (ДДВ-12), градусы Цельсия;';
st_TCmin15 = ' <= Tmin - минимальная средняя температура воздуха, градусы Цельсия;';
st_TCmax15 = ' <= Tmax - максимальная средняя температура воздуха, градусы Цельсия;';
st_TCmin16 = ' <= Tmin - минимальная средняя температура воздуха (ДСВ-16), градусы Цельсия;';
st_TCmax16 = ' <= Tmax - максимальная средняя температура воздуха (ДСВ-16), градусы Цельсия;';
st_Vmin = ' <= Vmin - минимальная скорость мгновенного горизонтального ветра, м/с;';
```

```
st_Vmax = ' <= Vmax - максимальная скорость мгновенного горизонтального ветра, м/с.';
st_SKO = '  а) среднеквадратичные отклонения RMS(x)';
st_DivSKO = '  б) среднеквадратичные разности значений за интервал между циклами';
st_Assim = '  в) коэффициенты асиметрии Ass(x)';
st_Exces = '  г) коэффициенты эксцесса Ex(x)';
st_Correl = '  д) коэффициенты корреляции R(x,y)';
st_com_uv1 =' где u и v - продольная и поперечная компоненты флуктуаций скорости ветра';
st_com_uv2 = ' (относительно среднего направления D горизонтального ветра).';
st_forTc = ' для T - ';
st_forVx = ' для Vs - ';
st_forVy = 'для Ve - ';
st_forw = ' для w - ';
st_foru = ' для u -  ';
st_forv = 'для v -  ';
st_forVh = ' для V - ';
st_forFh = 'для D -  ';
st_forVm = 'для Vm - ';
st_edinTC = ' град Ц;';
st_edinV = ' м/с;';
st_edinF = ' град;   ';
st_beetTw = ' между T и w - ';
st_beetTu = 'между T и u - ';
st_beetTv = ' между T и v - ';
st_beetTVh = 'между T и V - ';
st_beetTVm = ' между T и Vm -';
st_beetwVh = 'между w и V - ';
st_beetuw = ' между u и w - ';
st_beetvw = 'между v и w - ';
st_beetuv = ' между u и v - ';
st_Turb_a = '  а) измеряемые значения';
st_TurbE =' <= Ev - полная энергия турбулентных движений, (м^2)/(с^2);';
st_TurbIV=' <= Iv - относительная интенсивность флуктуаций скорости ветра;';
st_TurbIT=' <= Et - энергия температурных флуктуаций, град^2';
st_TurbMomentTau =' <= <u*w> - момент потока импульса, (м^2)/(с^2);';
st_TurbMomentH =' <= <w*T> - момент потока тепла, (м^2)/(с^2);';
st_TurbTau =' <= Tau - вертикальный поток импульса;';
st_TurbH =' <= H - вертикальный поток тепла;';
st_TurbCd =' <= Cd - коэффициент сопротивления потоку;';
st_TurbTmo =' <= T* - масштаб температуры, град;';
st_RurbVmo = ' <= u* - масштаб скорости ветра, м/с;';
st_TurbLmo =' <= L* - масштаб Монина-Обухова, м;';
st_TurbCt2 =' <= Ct2 - структ. постоянная флукт. температуры, град^2/(см^2/3);';
st_TurbCv2 =' <= Cv2 - структ. постоянная флуктуаций ветра, (м/с)^2/(см^2/3);';
st_TurbCn2 =' <= Cn2a - структ. пост. для акустического показат. преломл., 1/(м^2/3);';
st_TurbCn2opt =' <= Cn2o - структ. пост. для оптического показат. преломл., 1/(см^2/3).';

procedure CalcPlotnost(TC1,MMd1,Otvl1:Real; var Plo1:Real);far; external 'Meteo.DLL';
procedure CalcTurbVeter(vx1,vy1,vs1,ve1,vh1:Real;var u1,v1:Real);far; external 'Meteo.DLL';
function GetTurbEnerg(su1,sv1,sw1:Real):Real;far; external 'Meteo.DLL';
function GetTurbtaux(uw1:Real):Real;far; external 'Meteo.DLL';
function GetTurbH(Tw1:Real):Real;far; external 'Meteo.DLL';
function GetTurbVmo(taux1:Real):Real;far; external 'Meteo.DLL';
function GetTurbTmo(H1,Vmo1:Real):Real;far; external 'Meteo.DLL';
function GetTurbLmo(Vmo1,Tmo1,T1:Real):Real;far; external 'Meteo.DLL';
function GetTurbCd(Vh1,Vmo1:Real):Real;far; external 'Meteo.DLL';
function GetTurbCT2(sdT1,Vm1,dt1:Real):Real;far; external 'Meteo.DLL';
function GetTurbCV2(sdV1,Vm1,dt1:Real):Real;far; external 'Meteo.DLL';
function GetTurbCn2acust(CT21,CV21,T1:Real):Real;far; external 'Meteo.DLL';
function GetTurbCn2opt(CT21,Pmm1,T1:Real):Real;far; external 'Meteo.DLL';
```

```
{*************************************************************************}
function SetSurfaceParam(NumberSurface:Byte):Real;
Begin
  case NumberSurface of
   0: Result:=0.25/100;
   1: Result:=2/100;
   2: Result:=20/100;
   3: Result:=250/100;
   4: Result:=0.25/100;
   5: Result:=0.25/100;
   6: Result:=0.25/100;
   7: Result:=0.001/100;
   8: Result:=300/100;
   else Result:=2/100;
  end;
End;

//*****************************************
function MeteoData_CetSrednie:Boolean;
begin
  Result:=false;
  if Met_Data=nil then Exit;
  if GetSredGlobData(Met_Data,Met_Sred) then begin
  with  MeteoData do begin
  Sred.T:=Met_Sred[0];
  Sred.Vs:=Met_Sred[1];
  Sred.Ve:=Met_Sred[2];
  Sred.w:=Met_Sred[3];
  Sred.MMd:=Met_Sred[4];
  Sred.Otvl:=Met_Sred[5];
  Sred.Gx:=Met_Sred[6];
  Sred.Gy:=Met_Sred[7];
  Sred.Tr:=Met_Sred[8];
  CalcHorWind(Sred.Vs,Sred.Ve,Sred.Vh,Sred.Fh);
  Sred.Vmod:=sqrt(sqr(Sred.Vh)+sqr(Sred.w));
  Result:=true;
  end; end;
end;

//*********************************
procedure MeteoData_CalcWindMgnoven(Nstop:Integer; var V1:Real);
var
  k,i:Integer;
  S,Vper:Real;
  td,td0,tdi:TDateTime;
begin
  try
  V1:=-100;
  if Nstop > High(Met_Data) then Exit;
  td:=EncodeTime(0,0,2,0);
  i:=Nstop;
  tdi:=Met_Data[i].TDr;
  if tdi - Met_Data[0].TDr < td then Exit;
  k:=0; S:=0;
  td0:=tdi-td;
  while tdi > td0 do Begin
    if Met_Data[i].Sboj<4 then begin
      Vper:=sqrt(sqr(Met_Data[i].Mas[1])+sqr(Met_Data[i].Mas[2]));
      S:=S+Vper;
      inc(k);
```

```
      end;
     dec(i);
     if i>=0 then tdi:=Met_Data[i].TDr else tdi:=td0 - td;
     End; {while}
   if k > 0 then V1:=S/k;
   except
   V1:=-100;
   end;
 end;

 //*********************************
 procedure MeteoData_GetVminmaxSrok(var Vmin1,Vmax1:Real);
 var
   i,n1:Integer;
   V1:Real;
 begin
   try
   Vmin1:=100;  Vmax1:=-100;
   n1:=Trunc(2*TimeCikle)+1;
   if High(Met_Data)<n1 then Exit;
   for i:=n1 to High(Met_Data) do begin
     MeteoData_CalcWindMgnoven(i,V1);
     if V1>=0 then begin
       if V1<Vmin1 then Vmin1:=V1;
       if V1>Vmax1 then Vmax1:=V1;
       end;
     end;
   except
   Vmin1:=100;  Vmax1:=-100;
   end;
 end;

 //*************************************************
 function MeteoData_CalcStatistic:Boolean;
 var
   j,Col,k0:Integer;
   Ti,Vpri,Vpri_1,Ti_1,Vxi,Vyi,ui,vi,wi,Vhi,FiVhi,Vmodi,per,fiper,
   M3_T,M3_Vx,M3_Vy,M3_Vh,M3_Fi,M3_Vmod,M3_u,M3_v,M3_w,
   M4_T,M4_Vx,M4_Vy,M4_Vh,M4_Fi,M4_Vmod,M4_u,M4_v,M4_w:Real;
 procedure CalcStandart;
 begin
   if Met_Data[j].Sboj<=1 then begin
   Vpri_1:=Vpri; Ti_1:=Ti;
   Ti:=Met_Data[j].Mas[0]-MeteoData.Sred.T;
   Vxi:=Met_Data[j].Mas[1]-MeteoData.Sred.Vs;
   Vyi:=Met_Data[j].Mas[2]-MeteoData.Sred.Ve;
   wi:=Met_Data[j].Mas[3]-MeteoData.Sred.w;
   CalcTurbVeter(Vxi,Vyi,MeteoData.Sred.Vs,MeteoData.Sred.Ve,MeteoData.Sred.Vh,ui,vi);
   CalcHorWind(Met_Data[j].Mas[1],Met_Data[j].Mas[2],Vhi,Fiper);
   Vmodi:=sqrt(sqr(Vhi)+sqr(Met_Data[j].Mas[3]))-MeteoData.Sred.Vmod;
   Vhi:=Vhi-MeteoData.Sred.Vh;
   Fiper:=Fiper - MeteoData.Sred.Fh;
   if fiper > 180 then FiVhi:=180 - fiper
                 else
   if fiper < -180 then FiVhi:=180 + fiper
                  else FiVhi:=fiper;
   if abs(MeteoData.Sred.Vmod) > 1.0e-37 then
   Vpri:=(ui*MeteoData.Sred.Vh + wi*MeteoData.Sred.w)/MeteoData.Sred.Vmod
                                 else Vpri:=0;
    with MeteoData.SKO do begin
```

```
      per:=sqr(Ti);
      T:=T+per;
      M3_T:=M3_T + Ti*per;
      M4_T:=M4_T + sqr(per);
      per:=sqr(Vxi);
      Vs:=Vs + per;
      M3_Vx:=M3_Vx + Vxi*per;
      M4_Vx:=M4_Vx + sqr(per);
      per:=sqr(Vyi);
      Ve:=Ve + per;
      M3_Vy:=M3_Vy + Vyi*per;
      M4_Vy:=M4_Vy + sqr(per);
      per:=sqr(wi);
      w:=w + per;
      M3_w:=M3_w + wi*per;
      M4_w:=M4_w + sqr(per);
      per:=sqr(Vhi);
      Vh:=Vh + per;
      M3_Vh:=M3_Vh + Vhi*per;
      M4_Vh:=M4_Vh + sqr(per);
      per:=sqr(Vmodi);
      Vmod:=Vmod + per;
      M3_Vmod:=M3_Vmod + Vmodi*per;
      M4_Vmod:=M4_Vmod + sqr(per);
      per:=sqr(ui);
      u:=u + per;
      M3_u:=M3_u + ui*per;
      M4_u:=M4_u + sqr(per);
      per:=sqr(vi);
      v:=v + per;
      M3_v:=M3_v + vi*per;
      M4_v:=M4_v + sqr(per);
      per:=sqr(FiVhi);
      Fh:=Fh + per;
      M3_Fi:=M3_Fi + FiVhi*per;
      M4_Fi:=M4_Fi + sqr(per);
      if j>k0 then begin
        dTC:=dTC + sqr(Ti - Ti_1);
        dVmod:=dVmod + sqr(Vpri - Vpri_1);
        end;
      end;
      with MeteoData.KorM do begin
      T_w:=T_w + wi*Ti;
      T_u:=T_u + ui*Ti;
      T_v:=T_v + vi*Ti;
      T_Vh:=T_Vh + Vhi*Ti;
      T_Vmod:=T_Vmod + Vmodi*Ti;
      w_Vh:=w_Vh + wi*Vhi;
      u_w:=u_w + ui*wi;
      v_w:=v_w + vi*wi;
      u_v:=u_v + ui*vi;
      end;
      Inc(Col);
      end;
    end;
    begin
      try
      Result:=false;
      if ValidGlobData(Met_Data)<60 then begin
      Result:=false;
```

```
GetMyErrorDlg(Msg_CalcStat);
Exit;
End;
if not MeteoData_CetSrednie then Exit;
MeteoData_GetVminmaxSrok(MeteoData.VminSrok,MeteoData.VmaxSrok);
with MeteoData.SKO do Begin
T:=0; Vs:=0; Ve:=0; Vh:=0; Fh:=0; Vmod:=0;
u:=0; v:=0; w:=0; dTC:=0; dVmod:=0;
End;
with MeteoData.Asim do Begin
T:=0; Vs:=0; Ve:=0; Vh:=0; Fh:=0; Vmod:=0;
u:=0; v:=0; w:=0;
End;
with MeteoData.Exc do Begin
T:=0; Vs:=0; Ve:=0; Vh:=0; Fh:=0; Vmod:=0;
u:=0; v:=0; w:=0;
End;
with MeteoData.KorM do Begin
T_w:=0; T_u:=0; T_v:=0; T_Vh:=0; T_Vmod:=0;
w_Vh:=0; u_w:=0; v_w:=0; u_v:=0;
End;
M3_T:=0; M3_Vx:=0; M3_Vy:=0; M3_Vh:=0; M3_Fi:=0; M3_Vmod:=0;
M3_u:=0; M3_v:=0; M3_w:=0;
M4_T:=0; M4_Vx:=0; M4_Vy:=0; M4_Vh:=0; M4_Fi:=0; M4_Vmod:=0;
M4_u:=0; M4_v:=0; M4_w:=0;
Vpri:=0; Ti:=0; Col:=0;
if NumPosGlobData=High(Met_Data) then k0:=0
                          else k0:=NumPosGlobData+1;
for j:=k0 to High(Met_Data) do CalcStandart;
if k0<>0 then begin
  for j:=0 to NumPosGlobData do CalcStandart;
  end;
 if Col<2 then Begin
 Result:=false;
 Exit;
 End;
 Dec(Col);
 with MeteoData.SKO do Begin
 T:=sqrt(T/Col);
 Vs:=sqrt(Vs/Col);
 Ve:=sqrt(Ve/Col);
 w:=sqrt(w/Col);
 u:=sqrt(u/Col);
 v:=sqrt(v/Col);
 Vh:=sqrt(Vh/Col);
 Fh:=sqrt(Fh/Col);
 Vmod:=sqrt(Vmod/Col);
 dTC:=dTC/(Col-1);
 dVmod:=dVmod/(Col-1);
 End;
 with MeteoData do begin
 if abs(SKO.T) > 1e-20 then begin
 per:=sqr(SKO.T);
 Asim.T:=M3_T/Col/SKO.T/per;
 Exc.T:=M4_T/Col/per/per - 3;
 end else begin
 Asim.T:=0;
 Exc.T:=0;
 end; {SKO.T<>0}
 if abs(SKO.Vs)> 1e-20 then begin
```

```
per:=sqr(SKO.Vs);
Asim.Vs:=M3_Vx/Col/SKO.Vs/per;
Exc.Vs:=M4_Vx/Col/per/per - 3;
end else begin
Asim.Vs:=0;
Exc.Vs:=0;
end; {SKO.Vs<>0}
if abs(SKO.Ve) > 1e-20 then begin
per:=sqr(SKO.Ve);
Asim.Ve:=M3_Vy/Col/SKO.Ve/per;
Exc.Ve:=M4_Vy/Col/per/per - 3;
end else begin
Asim.Ve:=0;
Exc.Ve:=0;
end; {SKO.Ve<>0}
if abs(SKO.Vh) > 1e-20 then begin
per:=sqr(SKO.Vh);
Asim.Vh:=M3_Vh/Col/SKO.Vh/per;
Exc.Vh:=M4_Vh/Col/per/per - 3;
end else begin
Asim.Vh:=0;
Exc.Vh:=0;
end; {SKO.Vh<>0}
if abs(SKO.Fh) > 1e-20 then begin
per:=sqr(SKO.Fh);
Asim.Fh:=M3_Fi/Col/SKO.Fh/per;
Exc.Fh:=M4_Fi/Col/per/per - 3;
end else begin
Asim.Fh:=0;
Exc.Fh:=0;
end; {SKO.Fh<>0}
if abs(SKO.Vmod) > 1e-20 then begin
per:=sqr(SKO.Vmod);
Asim.Vmod:=M3_Vmod/Col/SKO.Vmod/per;
Exc.Vmod:=M4_Vmod/Col/per/per - 3;
end else begin
Asim.Vmod:=0;
Exc.Vmod:=0;
end; {SKO.Vmod<>0}
if abs(SKO.u) > 1e-20 then begin
per:=sqr(SKO.u);
Asim.u:=M3_u/Col/SKO.u/per;
Exc.u:=M4_u/Col/per/per - 3;
end else begin
Asim.u:=0;
Exc.u:=0;
end; {SKO.u<>0}
if abs(SKO.v) > 1e-20 then begin
per:=sqr(SKO.v);
Asim.v:=M3_v/Col/SKO.v/per;
Exc.v:=M4_v/Col/per/per - 3;
end else begin
Asim.v:=0;
Exc.v:=0;
end; {SKO.v<>0}
if abs(SKO.w) > 1e-20 then begin
per:=sqr(SKO.w);
Asim.w:=M3_w/Col/SKO.w/per;
Exc.w:=M4_w/Col/per/per - 3;
end else begin
```

```
    Asim.w:=0;
    Exc.w:=0;
    end; {SKO.w<>0}
    with KorM do begin
    T_w:=T_w/Col;
    T_u:=T_u/Col;
    T_v:=T_v/Col;
    T_Vh:=T_Vh/Col;
    T_Vmod:=T_Vmod/Col;
    w_Vh:=w_Vh/Col;
    u_w:=u_w/Col;
    v_w:=v_w/Col;
    u_v:=u_v/Col;
    end;
    end;
   with MeteoData do Begin
    Turb.Energ:=GetTurbEnerg(SKO.u,SKO.v,SKO.w);
    CalcPlotnost(Sred.T,Sred.MMd,Sred.OtVl,per);
    Turb.taux:=-per*GetTurbtaux(KorM.u_w);
    Turb.H:=1005*per*GetTurbH(Korm.T_w);
    Turb.Vmo:=GetTurbVmo(Turb.taux);
    Turb.Tmo:=GetTurbTmo(Turb.H,Turb.Vmo);
    Turb.Lmo:=GetTurbLmo(Turb.Vmo,Turb.Tmo,Sred.T);
    Turb.Cd:=GetTurbCd(Sred.Vh,Turb.Vmo);
    Turb.CT2:=GetTurbCT2(SKO.dTC,Sred.Vmod,TimeCikle);
    Turb.CV2:=GetTurbCV2(SKO.dVmod,Sred.Vmod,TimeCikle);
    Turb.Cn2:=GetTurbCn2acust(Turb.CT2,Turb.Cv2,Sred.T);
    Turb.Cn2opt:=GetTurbCn2opt(Turb.CT2,Sred.MMd,Sred.T);
    End;
   Result:=true;
   except;
   raise;
   end;
  end;

  {*************************************************************************}
  function GetMeteoInformStr(var St1:String):Boolean;
  var
   per:String;
   sm,Evl,AbsVl:Real;
  begin
   try
   Result:=false;
   if ValidGlobData(Met_Data)<60 then Exit;
   st1:=LogNumServer;
   per:='|';
   st1:=Format('%s%s',[st1,per]);
   DateTimeToString(per,'dd.mm.yy',Met_Data[NumPosGlobData].TDr);
   st1:=Format('%s%s',[st1,per]);
   per:=' ';
   st1:=Format('%s%s',[st1,per]);
   DateTimeToString(per,'hh:nn',Met_Data[NumPosGlobData].TDr);
   st1:=Format('%s%s',[st1,per]);
   per:=' ';
   st1:=Format('%s%s',[st1,per]);
   per:='|';
   st1:=Format('%s%s',[st1,per]);
   with MeteoData do begin
   per:=Format('%6.2f',[Sred.MMd]);
   st1:=Format('%s%s',[st1,per]);
```

```
    per:='|';
    st1:=Format('%s%s',[st1,per]);
    per:=Format('%6.2f',[Sred.T]);
    st1:=Format('%s%s',[st1,per]);
    if isDCB_15=0 then begin
      per:='|';
      st1:=Format('%s%s',[st1,per]);
      per:=Format('%6.2f',[Sred.Tr]);
      st1:=Format('%s%s',[st1,per]);
      end;
    per:='|';
    st1:=Format('%s%s',[st1,per]);
    per:=Format('%6.2f',[Sred.Vh]);
    st1:=Format('%s%s',[st1,per]);
    per:='|';
    st1:=Format('%s%s',[st1,per]);
    per:=Format('%6.2f',[Sred.Fh]);
    st1:=Format('%s%s',[st1,per]);
    per:='|';
    st1:=Format('%s%s',[st1,per]);
    per:=Format('%6.2f',[Sred.OtVl]);
    st1:=Format('%s%s',[st1,per]);
    per:='|';
    st1:=Format('%s%s',[st1,per]);
    sm:=7.665*Sred.T/(243.33+Sred.T);
    EVl:=Sred.OtVl*0.06107*exp(2.3025851*sm);
    AbsVl:=0.622*EVl/(0.78376 - 0.0007735*EVl);
    per:=Format('%6.2f',[AbsVl]);
    st1:=Format('%s%s',[st1,per]);
    end;
    Result:=true;
    except;
    raise;
    end;
  end;

  //*****************************************************************
  procedure GetTitleStrList(const Title:String; IsReport:Boolean;
                            var Data1:TStringList);
  var
    st:String;
    Evl,AbsVl,UdVlag,Td1,per:Real;
    Hour1,Min1,Sec1,MSec1:Word;
    Flag1,Flag2:Boolean;
    k0:Integer;
  begin
    try
    if IsReport then begin
      Flag1:=IsStatistica;
      Flag2:=IsTurbulent;
      end else begin
      Flag1:=true;
      Flag2:=true;
      end;
    with Data1 do Begin
    Add(LineTitle_R);
    Add(LineLider_R);
    Add(LineCaption_R);
    Add(Title);
    Add(LineTitle_R);
```

```
Add(LineIch_R);
Add(LineI_R2);
Add('');
Add(Format('%s%s',[DateTo_MyStr(Met_Data[NumPosGlobData].TDr),Title_Data]));
Add(Format('%s%s',[TimeTo_MyStr(Met_Data[NumPosGlobData].TDr),Title_Time]));
if NumPosGlobData=High(Met_Data) then k0:=0
                          else k0:=NumPosGlobData;
DecodeTime(Met_Data[NumPosGlobData].TDr-Met_Data[k0].TDr,Hour1,Min1,Sec1,MSec1);
per:=(3600*Hour1+60*Min1+Sec1+MSec1/1000)/60;
str(per:8:5,st);
Add(Format('%s%s',[st,Title_TimeUsred]));
if Flag1 or Flag2 then Begin
  str(1/TimeCikle:8:3,st);
  Add(Format('%s%s',[st,Title_TimeCikle]));
  End; {if Flag1 or Flag2}
if Flag1 or Flag2 then Begin
  st:=Format('%8d',[Length(Met_Data)]);
  Add(Format('%s%s',[st,Title_CountData]));
  End; {if Flag1 or Flag2}
if Flag2 then begin
  str(UGI_Z:8:2,st);
  Add(Format('%s%s',[st,Title_Zmeteo]));
  end;
if Flag2 then Begin
  st:=Format('%8d',[TypeSurface]);
  case TypeSurface of
   0: st:=Format('%s%s%s',[st,St_sneg,Title_SP]);
   1: st:=Format('%s%s%s',[st,St_trava,Title_SP]);
   2: st:=Format('%s%s%s',[st,St_rastit,Title_SP]);
   3: st:=Format('%s%s%s',[st,St_les,Title_SP]);
   4: st:=Format('%s%s%s',[st,St_pesok,Title_SP]);
   5: st:=Format('%s%s%s',[st,St_pashnja,Title_SP]);
   6: st:=Format('%s%s%s',[st,St_grunt,Title_SP]);
   7: st:=Format('%s%s%s',[st,St_voda,Title_SP]);
   8: st:=Format('%s%s%s',[st,St_gorod,Title_SP]);
  end;
  Add(st);
  End; {if Flag2}
Add(LineIIch_R);
Add(LineII_R2);
Add(st_MET_ach);
Add(st_MET_a);
Add(st_MET_airch);
Add(st_MET_air);
Add('');
with MeteoData do begin
str(Sred.T:8:2,st);
Add(Format('%s%s',[st,st_TC15]));
str(SKO.T:8:2,st);
Add(Format('%s%s',[st,st_RMSTC]));
if SizeDavlen=1 then begin
  str(Sred.MMd:8:2,st);
  Add(Format('%s%s',[st,st_MMdavl1]));
  end else begin
  str(Sred.MMd/0.75:8:2,st);
  Add(Format('%s%s',[st,st_MMdavl2]));
  end;
str(Sred.OtVl:8:2,st);
Add(Format('%s%s',[st,st_rHu]));
Calc_EQTrosa(Sred.T,Sred.OtVl,Evl,Absvl,Td1);
```

```
str(Evl:8:2,st);
Add(Format('%s%s',[st,st_eHu]));
per:=Evl/Sred.OtVl*100-Evl;
str(per:8:2,st);
Add(Format('%s%s',[st,st_EdHu]));
str(Td1:8:2,st);
Add(Format('%s%s',[st,st_TCd]));
str(AbsVl:8:2,st);
Add(Format('%s%s',[st,st_AbsVlag]));
UdVlag:=622*EVl/(Sred.MMd*1.33322 - 0.378*EVl);
str(UdVlag:8:2,st);
Add(Format('%s%s',[st,st_moo]));
CalcPlotnost(Sred.T,Sred.MMd,Sred.OtVl,per);
str(per:8:4,st);
Add(Format('%s%s',[st,st_Plotnoct]));
Add(st_MET_windch);
Add(st_MET_wind);
Add('');
str(Sred.Vh:8:2,st);
Add(Format('%s%s',[st,st_Vh]));
if VminSrok<99 then begin
  str(VminSrok:8:2,st);
  Add(Format('%s%s',[st,st_VminSrok]));
  end;
if VmaxSrok>-99 then begin
  str(VmaxSrok:8:2,st);
  Add(Format('%s%s',[st,st_VmaxSrok]));
  end;
str(SKO.Vh:8:2,st);
Add(Format('%s%s',[st,st_RMSVh]));
if SizeDirWind=1 then begin
  str(Sred.Fh:8:2,st);
  Add(Format('%s%s',[st,st_Fh1]));
   end else begin
   st:=GetStrDirWindDYg(Sred.Fh);
   Add(Format('%s%s',[st,st_Fh2]));
   end;
str(SKO.Fh:8:2,st);
Add(Format('%s%s',[st,st_RMSFh]));
str(Sred.w:8:2,st);
Add(Format('%s%s',[st,st_wert]));
str(SKO.w:8:2,st);
Add(Format('%s%s',[st,st_RMSwert]));
str(Sred.Vmod:8:2,st);
Add(Format('%s%s',[st,st_Vm]));
if abs(Sred.Vh)<1.e-20 then per:=0 else
per:=180/pi*arctan(Sred.w/Sred.Vh);
str(per:8:2,st);
Add(Format('%s%s',[st,st_Alfa]));
str(Sred.Vs:8:2,st);
Add(Format('%s%s',[st,st_Vx]));
str(Sred.Ve:8:2,st);
Add(Format('%s%s',[st,st_Vy]));
if Met_Data[High(Met_Data)].TDr>=
 Met_Data[0].TDr+EncodeTime(0,TimeUsredn div 60,0,0)- EncodeTime(0,0,2,0) then begin
Add(st_MET_bch);
Add(st_MET_b);
Add('');
if Trmin<99 then begin
  str(Trmin:8:2,st);
```

```
      Add(Format('%s%s',[st,st_TCmin2]));
      end;
    if Trmax>-99 then begin
      str(Trmax:8:2,st);
      Add(Format('%s%s',[st,st_TCmax2]));
      end;
    end;
    if Tmin<99 then begin
      str(Tmin:8:2,st);
      if isDCB_15=0 then Add(Format('%s%s',[st,st_TCmin16]))
                else Add(Format('%s%s',[st,st_TCmin15]));
      end;
    if Tmax>-99 then begin
      str(Tmax:8:2,st);
      if isDCB_15=0 then Add(Format('%s%s',[st,st_TCmax16]))
                else Add(Format('%s%s',[st,st_TCmax15]));
      end;
    if Vmin<99 then begin
      str(Vmin:8:2,st);
      Add(Format('%s%s',[st,st_Vmin]));
      end;
    if Vmax>-99 then begin
      str(Vmax:8:2,st);
      Add(Format('%s%s',[st,st_Vmax]));
      end;
    end; {with MeteoData}
    End; {Data1}
    except
    raise;
    end;
  end;

  {*************************************}
  procedure CommentStrList(var Data1:TStringList);
  begin
    try
    with Data1 do Begin
    Add(LineIVch_R);
    Add(LineIV_R);
    Add(St_err0);
    Add(St_err1);
    Add(St_err2);
    Add(St_err3);
    Add(St_err3d);
    Add(St_err4);
    Add(St_err5);
    Add(St_err6);
    Add(St_err7);
    Add(St_err8);
    Add(St_erra);
    ADD(St_errb);
    End; {Data1}
    except
    raise;
    end;
  end;

  {*************************************************************************}
  function MeteoData_GetDataStrList(var Data1:TStringList):Boolean;
  var
```

```
  st,st2:String;
  j,k0,Num1:Integer;
procedure OutDataStr(var Data1:TStringList);
begin
  inc(Num1);
  st:=Format('%6d',[Num1]);
  str(Met_Data[j].Mas[0]:6:2,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  str(Met_Data[j].Mas[1]:6:2,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  str(Met_Data[j].Mas[2]:6:2,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  str(Met_Data[j].Mas[3]:6:2,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  str(Met_Data[j].Mas[4]:6:1,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  str(Met_Data[j].Mas[5]:6:1,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  st2:=Format('%3d',[Met_Data[j].Sboj]);
  st:=Format('%s%s%s',[st,' ',st2]);
  Data1.Add(st);
end;
begin
  try
  Result:=false;
  if ValidGlobData(Met_Data)<60 then Exit;
  GetTitleStrList(TitleData_R,false,Data1);
  with Data1 do Begin
  Add(LineIIIch_R);
  Add(LineIII_R1);
  Add(LineIIIval);
  if NumPosGlobData=High(Met_Data) then k0:=0
                          else k0:=NumPosGlobData+1;
  Num1:=0;
  for j:=k0 to High(Met_Data) do OutDataStr(Data1);
  if k0<>0 then begin
    for j:=0 to NumPosGlobData do OutDataStr(Data1);
    end;
  End; {Data1}
  CommentStrList(Data1);
  Result:=true;
  except
  raise;
  end;
end;

//*****************************************************************
function MeteoData_GetReportStrList(var Data1:TStringList):Boolean;
var
  st,st1,st2:String;
  per,per2:Real;
begin
  try
  Result:=false;
  if ValidGlobData(Met_Data)<60 then Exit;
  GetTitleStrList(TitleReport_R,true,Data1);
  with Data1 do Begin
  with MeteoData do begin
  if IsStatistica then Begin
  Add(LineStch_R);
```

```
Add(LineSt_R);
Add('');
Add(st_SKO);
str(SKO.T:6:2,st);
st:=Format('%s%s%s%s',[st_forTc,st,st_edinTC,'   ']);
str(SKO.Vs:6:2,st1);
st1:=Format('%s%s%s%s',[st_forVx,st1,st_edinV,'    ']);
str(SKO.Ve:6:2,st2);
st2:=Format('%s%s%s',[st_forVy,st2,st_edinV]);
Add(Format('%s%s%s',[st,st1,st2]));
str(SKO.w:6:2,st);
st:=Format('%s%s%s%s',[st_forw,st,st_edinV,'      ']);
str(SKO.u:6:2,st1);
st1:=Format('%s%s%s%s',[st_foru,st1,st_edinV,'    ']);
str(SKO.v:6:2,st2);
st2:=Format('%s%s%s',[st_forv,st2,st_edinV]);
Add(Format('%s%s%s',[st,st1,st2]));
str(SKO.Vh:6:2,st);
st:=Format('%s%s%s%s',[st_forVh,st,st_edinV,'       ']);
str(SKO.Fh:6:2,st1);
st1:=Format('%s%s%s',[st_forFh,st1,st_edinF]);
str(SKO.Vmod:6:2,st2);
st2:=Format('%s%s%s',[st_forVm,st2,st_edinV]);
Add(Format('%s%s%s',[st,st1,st2]));
Add('');
Add(st_DivSKO);
str(SKO.dTC:6:2,st);
st:=Format('%s%s%s%s',[st_forTc,st,st_edinTC,'   ']);
str(SKO.Vmod:6:2,st1);
st1:=Format('%s%s%s%s',[st_forVm,st1,st_edinV,'    ']);
Add(Format('%s%s',[st,st1]));
Add('');
Add(st_Assim);
str(Asim.T:6:2,st);
st:=Format('%s%s%s',[st_forTc,st,';        ']);
str(Asim.Vs:6:2,st1);
st1:=Format('%s%s%s',[st_forVx,st1,';       ']);
str(Asim.Ve:6:2,st2);
st2:=Format('%s%s%s',[st_forVy,st2,';']);
Add(Format('%s%s%s',[st,st1,st2]));
str(Asim.w:6:2,st);
st:=Format('%s%s%s',[st_forw,st,';        ']);
str(Asim.u:6:2,st1);
st1:=Format('%s%s%s',[st_foru,st1,';       ']);
str(Asim.v:6:2,st2);
st2:=Format('%s%s%s',[st_forv,st2,';']);
Add(Format('%s%s%s',[st,st1,st2]));
str(Asim.Vh:6:2,st);
st:=Format('%s%s%s',[st_forVh,st,';         ']);
str(Asim.Fh:6:2,st1);
st1:=Format('%s%s%s',[st_forFh,st1,';       ']);
str(Asim.Vmod:6:2,st2);
st2:=Format('%s%s%s',[st_forVm,st2,';']);
Add(Format('%s%s%s',[st,st1,st2]));
Add('');
Add(st_Exces);
str(Exc.T:6:2,st);
st:=Format('%s%s%s',[st_forTc,st,';        ']);
str(Exc.Vs:6:2,st1);
st1:=Format('%s%s%s',[st_forVx,st1,';       ']);
```

```
str(Exc.Ve:6:2,st2);
st2:=Format('%s%s%s',[st_forVy,st2,';']);
Add(Format('%s%s%s',[st,st1,st2]));
str(Exc.w:6:2,st);
st:=Format('%s%s%s',[st_forw,st,';        ']);
str(Exc.u:6:2,st1);
st1:=Format('%s%s%s',[st_foru,st1,';       ']);
str(Exc.v:6:2,st2);
st2:=Format('%s%s%s',[st_forv,st2,';']);
Add(Format('%s%s%s',[st,st1,st2]));
str(Exc.Vh:6:2,st);
st:=Format('%s%s%s',[st_forVh,st,';         ']);
str(Exc.Fh:6:2,st1);
st1:=Format('%s%s%s',[st_forFh,st1,';       ']);
str(Exc.Vmod:6:2,st2);
st2:=Format('%s%s%s',[st_forVm,st2,';']);
Add(Format('%s%s%s',[st,st1,st2]));
Add('');
Add(st_Correl);
if (SKO.T>1e-20) and (SKO.w>1e-20) then
per:=KorM.T_w/SKO.T/SKO.w else per:=0;
str(per:6:2,st);
st:=Format('%s%s%s',[st_beetTw,st,';        ']);
if (SKO.T>1e-20) and (SKO.u>1e-20) then
per:=KorM.T_u/SKO.T/SKO.u else per:=0;
str(per:6:2,st1);
st1:=Format('%s%s%s',[st_beetTu,st1,';']);
Add(Format('%s%s',[st,st1]));
if (SKO.T>1e-20) and (SKO.v>1e-20) then
per:=KorM.T_v/SKO.T/SKO.v else per:=0;
str(per:6:2,st);
st:=Format('%s%s%s',[st_beetTv,st,';        ']);
if (SKO.T>1e-20) and (SKO.Vh>1e-20) then
per:=KorM.T_Vh/SKO.T/SKO.Vh else per:=0;
str(per:6:2,st1);
st1:=Format('%s%s%s',[st_beetTVh,st1,';']);
Add(Format('%s%s',[st,st1]));
if (SKO.T>1e-20) and (SKO.Vmod>1e-20) then
per:=KorM.T_Vmod/SKO.T/SKO.Vmod else per:=0;
str(per:6:2,st);
st:=Format('%s%s%s',[st_beetTVm,st,';        ']);
if (SKO.w>1e-20) and (SKO.Vh>1e-20) then
per:=KorM.w_Vh/SKO.w/SKO.Vh else per:=0;
str(per:6:2,st1);
st1:=Format('%s%s%s',[st_beetwVh,st1,';']);
Add(Format('%s%s',[st,st1]));
if (SKO.u>1e-20) and (SKO.w>1e-20) then
per:=KorM.u_w/SKO.u/SKO.w else per:=0;
str(per:6:2,st);
st:=Format('%s%s%s',[st_beetuw,st,';        ']);
if (SKO.v>1e-20) and (SKO.w>1e-20) then
per:=KorM.v_w/SKO.v/SKO.w else per:=0;
str(per:6:2,st1);
st1:=Format('%s%s%s',[st_beetvw,st1,';']);
Add(Format('%s%s',[st,st1]));
if (SKO.u>1e-20) and (SKO.v>1e-20) then
per:=KorM.u_v/SKO.u/SKO.v else per:=0;
str(per:6:2,st);
Add(Format('%s%s%s',[st_beetuv,st,';']));
Add(st_com_uv1);
```

```
 Add(st_com_uv2);
 End; {if IsStatistica}
 if IsTurbulent then Begin
 Add(LineTurbch_R);
 Add(LineTurb_R);
 Add('');
// Add(st_Turb_a);
 str(Turb.Energ:12,st);
 Add(Format('%s%s',[st,st_TurbE]));
 if Sred.Vmod> 1e-20 then per:=Turb.Energ/sqr(Sred.Vmod)
                   else per:=0;
 str(per:12,st);
 Add(Format('%s%s',[st,st_TurbIV]));
 per:=sqr(SKO.T)/2;
  str(per:12,st);
 Add(Format('%s%s',[st,st_TurbIT]));
 str(Turb.Taux:12,st);
 Add(Format('%s%s',[st,st_TurbMomentTau]));
 str(Turb.H:12,st);
 Add(Format('%s%s',[st,st_TurbMomentH]));
 CalcPlotnost(Sred.T,Sred.MMd,Sred.OtVl,per2);
 per:=-per2*Turb.Taux;
 str(per:12,st);
 Add(Format('%s%s',[st,st_TurbTau]));
 per:=1005*per2*Turb.H;
 str(per:12,st);
 Add(Format('%s%s',[st,st_TurbH]));
 str(Turb.Cd:12,st);
 Add(Format('%s%s',[st,st_TurbCd]));
 Add('');
 str(Turb.Tmo:12,st);
 Add(Format('%s%s',[st,st_TurbTmo]));
 str(Turb.Vmo:12,st);
 Add(Format('%s%s',[st,st_RurbVmo]));
 str(Turb.Lmo:12,st);
 Add(Format('%s%s',[st,st_TurbLmo]));
 per2:=exp(-4./3*ln(10));
 per:=Turb.Ct2*per2;
 str(per:12,st);
 Add(Format('%s%s',[st,st_TurbCt2]));
 per:=Turb.Cv2*per2;
 str(per:12,st);
 Add(Format('%s%s',[st,st_TurbCv2]));
 str(Turb.Cn2:12,st);
 Add(Format('%s%s',[st,st_TurbCn2]));
 per:=Turb.Cn2opt*per2;
 str(per:12,st);
 Add(Format('%s%s',[st,st_TurbCn2opt]));
 End; {if IsTurbulent}
 end;
 End; {Data1}
 Result:=true;
 except
 raise;
 end;
end;

END.

{8888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
```

Изм	Лист	№ докум	Подп.	Дата	АМЯ4.079.001 12	Лист 110

Инв. № подл. | Подп. и дата | Взам. инв.№ | Инв. № дубл. | Подп. и дата

```
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT UnDevice;
interface
uses
  Controls, Forms;

var
  Speed,MMdavl,OtVlag,AVlag,Evlag,Trosa,GammaX,GammaY,DatTC,TC,Vx,Vy,Vz,Vhor,Fhor:Real;
  TkalibrNew,Speed_C,MMdavl_C,OtVlag_C,AbsVlag_C:Real;
  CosTilt,SinTilt,CosUgiTilt,SinUgiTilt:Real;
  Rast,DelayTime,Smm_C,tmks_C:Array [0..3] of Real;

procedure InitDefault;
procedure AddDefault;
procedure CalcCikleMeteo;
function CalcStopMeteo:Boolean;
function CalcStopControl:Boolean;
procedure CalcCikleCalibr;
function CalcStopCalibr:Boolean;

implementation
uses UnAMKsys,UnAMKini,UnData,UnMeteo;

procedure Recalc_R(Tc1,Tr1:Real; var r1:Real);far; external 'Meteo.DLL';
procedure Calc_TC(c1,r1,p1:Real; var t1:Real);far; external 'Meteo.DLL';
procedure Calc_C(t1,r1,p1:Real; var c1:Real);far; external 'Meteo.DLL';
procedure CalcKvadro(var vx1,vy1,vz1,c1:Real);far; external 'Meteo.DLL';
procedure CorrectWind0(k1,u12,u34,du12,du34:Real; var vx1,vy1,vz1,c1:Real);far; external 'Meteo.DLL';
procedure CorrectWind1(d1,h1:Real; var vx1,vy1:Real);far; external 'Meteo.DLL';
procedure CorrectWind2(gx1,gy1,cos1,sin1:Real; var vx1,vy1,vz1:Real);far; external 'Meteo.DLL';
procedure CorrectWind3(a0,a1,a2,a3,a4,b0:Real; var vx1,vy1:Real);far; external 'Meteo.DLL';

//**********************************
procedure InitDefault;
begin
  try
  ParolCurrent:='';
  OtVlag:=OtVlagKl;
  MMdavl:=MMdavlKl;
  GammaX:=45;
  GammaY:=45;
  TC:=TCalibr;
  Vx:=0;
  Vy:=0;
  Vz:=0;
  Vhor:=0;
  Fhor:=0;
  OtVlag_C:=OtVlagKl;
  MMdavl_C:=MMdavlKl;
  DatTC:=TC;
  Calc_EQTrosa(TC,Otvlag,Evlag,Avlag,Trosa);
  AddDefault;
  except;
  raise;
  end;
end;

//***********************************
procedure AddDefault;
```

```
var
  i:Integer;
begin
  try
  CosTilt:=cos(pi/180*G_hor);
  SinTilt:=-sin(pi/180*G_hor);
  CosUgiTilt:=cos(pi/180*(Ugi_hor+G_hor));
  SinUgiTilt:=sin(pi/180*(Ugi_hor+G_hor));
  for i:=0 to 3 do begin
   Rast[i]:=RastMM[i]/1000;
   Smm_C[i]:=RastMM[i];
   end;
  except;
  raise;
  end;
end;

//**********************************
procedure CalcSonic;
const
  b1=1.4142134;
var
   Ugr:Array [0..3] of Real;
   Ugr12,Ugr34,dUgr12,dUgr34,v2:Real;
   i:Byte;
begin
  for i:=0 to 3 do
    Ugr[i]:=Rast[i]/DelayTime[i];
  Ugr12:=Ugr[0] + Ugr[1];
  Ugr34:=Ugr[2] + Ugr[3];
  dUgr12:=Ugr[0] - Ugr[1];
  dUgr34:=Ugr[2] - Ugr[3];
  Vx:=dUgr12/b1;
  Vy:=dUgr34/b1;
  Vz:=(Ugr12 - Ugr34)/(2*b1);
  Speed:=(Ugr12 + Ugr34)/4;
  v2:=Vx*Vx+Vy*Vy+Vz*Vz;
  if IsCorrectZ and (V2 > 0.01) then
     CorrectWind0(UGI_Coefz,ugr12,ugr34,dugr12,dugr34,Vx,Vy,Vz,Speed);
  if V2>10 then CalcKvadro(Vx,Vy,Vz,Speed);
 end;

//**********************************
procedure GetOtnVlag(X6:Real; Err1:Word; var Otvl1:Real);
var
  x1:Real;
begin
  if not (IsDeviceHu and (Err1<6)) then
    OtVl1:=OtVlagKl
    else begin
  x1:=X6-r_X0;
  Otvl1:=r_r0 + r_ln*x1 + r_kv*x1*x1*1.e-7+r_delta;
  if Otvl1>129 then Otvl1:=129;
  if Otvl1<0.001 then Otvl1:=0.001;
  if IsCutHu_100 and (Otvl1>100) then Otvl1:=100;
  end;
end;

//**********************************
procedure GetTempFromVlag(X10,Tc1:Real; Err1:Word; var Tr1:Real);
```

```
var
  x1:Real;
begin
  if not (IsDeviceHu and (Err1<6)) then
    Tr1:=Tc1
    else begin
  x1:=X10-Tr_x0;
  Tr1:=Tr_T0 + Tr_ln*x1 + Tr_kv*x1*x1*1.e-7+Tr_delta;
  if Tr1>70 then Tr1:=70;
  if Tr1<-70 then Tr1:=-70;
  end;
end;

//**********************************
procedure GetDavlenie(X5:Real; Err1:Word; var Pmm1:Real);
begin
  if not (IsDeviceP and (Err1<5)) then
    Pmm1:=MMdavlKl
  else begin
  Pmm1:=P_P0+P_ln*(X5-P_x0)+P_delta;
  if Pmm1<450 then Pmm1:=450;
  if Pmm1>850 then Pmm1:=850;
  end;
end;

//**********************************
procedure RecalcDavlenOnTemp(Tc1:Real; var Pmm1:Real);
var
  dx1,PT:Real;
begin
  dx1:=Tc1-P_T0;
  PT:=PT_ln*dx1 + 1.e-3*PT_kv*sqr(dx1) + 1.e-5*PT_kb*dx1*sqr(dx1);
  Pmm1:=Pmm1-PT;
end;

//**********************************
procedure GetTitles(X7,X8,X9:Real; Err1:Word; var Gx,Gy:Real);
var
  a1,tgx1,tgy1:Real;
begin
  try
  if not Err1<7 then begin
  Gx:=45; Gy:=45;
  end else begin
  a1:=(X7/X8+G_x0)/G_xd;
  tgx1:=a1/sqrt(abs(1-sqr(a1)));
  a1:=(X9/X8-G_y0)/G_yd;
  tgy1:=a1/sqrt(abs(1+sqr(a1)));
  Gx:=180/pi*arctan(tgx1*CosTilt+tgy1*SinTilt)+ G_xdelta;
  Gy:=180/pi*arctan(tgy1*CosTilt-tgx1*SinTilt) + G_ydelta;
  if Gx>90 then Gx:=90;  if Gx<-90 then Gx:=-90;
  if Gy>90 then Gy:=90;  if Gy<-90 then Gy:=-90;
  end;
  except;
  raise;
  end;
end;

//**********************************
procedure CalcRast(TC1:Real);
```

```
var
  i:Byte;
begin
  for i:=0 to 3 do
    Rast[i]:=RastMM[i]/1000*(1 + 1.e-5*UGI_Tcoef*(TC1-Tcalibr));
end;

//**************************************************
procedure CalcDelayTime(const Data1:TInputData);
var
  i:Byte;
begin
  for i:=0 to 3 do Begin
  DelayTime[i]:=(Data1[i+1] + UGI_Dela[i])/UGI_Fkg;
  end
end;

//*******************************************************
procedure CalcCikleMeteo;
var
  TCold:Real;
  k:Byte;
begin
  try
  GetDavlenie(InputData[5],Data_UGI[0],MMdavl);
  GetOtnVlag(InputData[6],Data_UGI[0],OtVlag);
  GetTempFromVlag(InputData[10],TC,Data_UGI[0],DatTC);
  CalcDelayTime(InputData);
  k:=0;
  TCold:=-100;
    while (k<7) and (abs(TC-TCold)>0.01) do begin
      Inc(k);
      TCold:=TC;
      CalcSonic;
      Calc_TC(Speed,OtVlag,MMdavl,TC);
      if (IsDCB_15=1) and IsDeviceHu and (Data_UGI[0]<6) then begin
        Recalc_R(TC,DatTC,OtVlag);
        if IsCutHu_100 and (OtVlag>100) then OtVlag:=100;
        Calc_TC(Speed,OtVlag,MMdavl,TC);
        end;
      CalcRast(TC);
    end; {while}
  if IsDeviceP and (Data_UGI[0]<5) then RecalcDavlenOnTemp(DatTC,MMdavl);
  GetTitles(InputData[7],InputData[8],InputData[9],Data_UGI[0],GammaX,GammaY);
  CorrectWind2(GammaX,GammaY,CosUgiTilt,SinUgiTilt,Vx,Vy,Vz);
  if IsCorrectZ then
    CorrectWind3(UGI_Az0,UGI_Az1,UGI_Az2,UGI_Az3,UGI_Az4,UGI_Fiz,Vx,Vy);
  CorrectWind1(DeviceOrient,UGI_hor,Vx,Vy);
  CalcHorWind(Vx,Vy,Vhor,Fhor);
  Calc_EQTrosa(TC,Otvlag,Evlag,Avlag,Trosa);
  except;
  raise;
  end;
end;

//***********************************
procedure UpdateNewMeteo_List;
begin
  with MeteoData do begin
  TC:=Sred.T;
```

```
  Vx:=Sred.Vs;
  Vy:=Sred.Ve;
  Vz:=Sred.w;
  Vhor:=Sred.Vh;
  Fhor:=Sred.Fh;
  MMdavl:=Sred.MMd;
  OtVlag:=Sred.Otvl;
  GammaX:=Sred.Gx;
  GammaY:=Sred.Gy;
  end;
  Calc_EQTrosa(TC,Otvlag,Evlag,Avlag,Trosa);
  Calc_C(TC,OtVlag,MMdavl,Speed);
  CalcRast(TC);
end;

//*************************************
function CalcStopMeteo:Boolean;
begin
  try
  try
  Screen.Cursor:=crHourGlass;
  if IsFiltrOutput then FiltrateGlobData(Met_Data);
  if MeteoData_CalcStatistic then Begin
  Result:=true;
  UpdateNewMeteo_List;
  End else Result:=false;
  finally
  Screen.Cursor:=crDefault;
  end;
  except;
    raise;
    end;
end;

//*************************************
function CalcStopControl:Boolean;
begin
  try
  if IsFiltrOutput then
  FiltrateGlobData(Con_Data);
  Result:=GetSredGlobData(Con_Data,Con_Sred);
  except;
  raise;
  end;
end;

//*************************************
procedure PrepeCikleCalibr;
var
  e1,td1:Real;
begin
  try
  GetDavlenie(InputData[5],Data_UGI[0],MMdavl_C);
  RecalcDavlenOnTemp(TkalibrNew,MMdavl_C);
  GetOtnVlag(InputData[6],Data_UGI[0],OtVlag_C);
  GetTempFromVlag(InputData[10],TkalibrNew,Data_UGI[0],DatTC);
  if (IsDCB_15=1) and IsDeviceHu and (Data_UGI[0]<6) then begin
       Recalc_R(TkalibrNew,DatTC,OtVlag_C);
       if IsCutHu_100 and (OtVlag>100) then OtVlag:=100;
       end;
```

```
  Calc_EQTrosa(TkalibrNew,OtVlag_C,e1,Absvlag_C,td1);
  Calc_C(TkalibrNew,OtVlag_C,MMdavl_C,Speed_C);
  GetTitles(InputData[7],InputData[8],InputData[9],Data_UGI[0],GammaX,GammaY);
  except;
  raise;
  end;
end;

//*************************************
procedure CalcCikleCalibr;
var
  i:Byte;
begin
  try
  PrepeCikleCalibr;
  CalcDelayTime(InputData);
  for i:=0 to 3 do Begin
  Rast[i]:=Speed_C*DelayTime[i];
  Smm_C[i]:=Rast[i]*1000;
  tmks_C[i]:=1.e6*DelayTime[i];
  End;
  except;
  raise;
  end;
end;

//*************************************
function CalcStopCalibr:Boolean;
var
  i:Byte;
begin
  try
  try
  Screen.Cursor:=crHourGlass;
  if IsFiltrOutput then FiltrateGlobData(Cal_Data);
  if GetSredGlobData(Cal_Data,Cal_Sred) then Begin
  Result:=true;;
  MMdavl_C:=Cal_Sred[4];
  OtVlag_C:=Cal_Sred[5];
  GammaX:=Cal_Sred[6];
  GammaY:=Cal_Sred[7];
  Calc_C(TkalibrNew,OtVlag_C,MMdavl_C,Speed_C);
  for i:=0 to 3 do Begin
   Smm_C[i]:=Cal_Sred[i];
   Rast[i]:=Smm_C[i]/1000;
   DelayTime[i]:=Rast[i]/Speed_C;
   tmks_C[i]:=DelayTime[i]*1.e6;
  End;
  End else Result:=false;
  finally
  Screen.Cursor:=crDefault;
  end;
  except;
    raise;
    end;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
```

```
{888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT GraphLib;
interface
uses
  SysUtils, WinProcs, Classes, Graphics;

  function GetColorFromIndex(Index:Word):TColor;
  function GetStyleLineFromIndex(Index1:Word):TPenStyle;
  procedure ClearGraphWindow(TheCanvas:TCanvas; Width,Height:Integer);
  procedure DrawWinDirWind(Canvas1:TCanvas; Height1,Width1:Integer; isSmall1:Boolean);
  procedure DrawStrelkaDirWind(Fi:Real;Col:TColor;Canvas1:TCanvas;
                         Height1,Width1:Integer);
  procedure DrawWinAmpl(a0,ad,xd:Integer;Canvas1:TCanvas; Height1,Width1:Integer);
  procedure DrawStrelkaAmpl(Am:Real;a0,xd:Integer;Col:TColor;
                       Canvas1:TCanvas;Height1,Width1:Integer);

implementation
uses UnAMKini;

const
Malrad1=4;

{********************************************************}
function GetColorFromIndex(Index:Word):TColor;
Begin
  case Index of
   0:Result:=clBlack;
   1:Result:=clMaroon;
   2:Result:=clGreen;
   3:Result:=clOlive;
   4:Result:=clNavy;
   5:Result:=clPurple;
   6:Result:=clTeal;
   7:Result:=clGray;
   8:Result:=clSilver;
   9:Result:=clRed;
   10:Result:=clLime;
   11:Result:=clYellow;
   12:Result:=clBlue;
   13:Result:=clFuchsia;
   14:Result:=clAqua;
   15:Result:=clWhite;
   16:Result:=clMoneyGreen;
   17:Result:=clSkyBlue;
   18:Result:=clCream;
   else Result:=clBlack;
  end;
End;

{**********************************************************}
function GetStyleLineFromIndex(Index1:Word):TPenStyle;
Begin
  case Index1 of
   0:Result:=psSolid;
   1:Result:=psDash;
   2:Result:=psDot;
   3:Result:=psDashDot;
   4:Result:=psDashDotDot;
   else Result:=psSolid;
```

Изм	Лист	№ докум	Подп.	Дата	АМЯ4.079.001 12	Лист 117

Подп. и дата | Инв. № дубл. | Взам. инв.№ | Подп. и дата | Инв. № подл.

```
  end;
End;

{*************************************************************************}
procedure ClearGraphWindow(TheCanvas:TCanvas; Width,Height:Integer);
var
  R:TRect;
begin
  With TheCanvas do Begin
  Brush.Color:=GetColorFromIndex(WindowColor);
  R:=Rect(0,0,Width,Height);
  FillRect(R);
  end;
end;

{*************************************************************************}
procedure DrawWinDirWind(Canvas1:TCanvas; Height1,Width1:Integer; isSmall1:Boolean);
var
  i,step,centrex,centrey,x,y:Integer;
  xper,yper,deg:Real;
begin
 ClearGraphWindow(Canvas1,Width1,Height1);
 With Canvas1 do Begin
 yper:=(Height1/2-TextHeight('S')-2)/1.2;
 xper:=(Width1/2-TextWidth('W')-5)/1.2;
 if xper<=yper then step:=round(xper)
            else step:=round(yper);
 centrex := Width1 div 2;
 centrey := Height1 div 2;
    Pen.Color:=GetColorFromIndex(LabelColor);
    Pen.Width:=1;
    Brush.Color:=GetColorFromIndex(DiagColor);
 Ellipse(centrex-step,centrey-step,centrex+step,centrey+step);
 Brush.Color:=GetColorFromIndex(ArrowColor);
 Ellipse(centrex-Malrad1,centrey-Malrad1,centrex+Malrad1,centrey+Malrad1);
 Brush.Color:=GetColorFromIndex(WindowColor);
 deg:=pi*10/180;
 for i := 0 to 35 do
 begin
  yper:=sin(i*deg)*step;
  xper:=cos(i*deg)*step;
  MoveTo(centrex+round(1.1*xper),centrey+round(1.1*yper));
  LineTo(centrex+round(xper),centrey+round(yper));
 end;
 deg:=pi*30/180;
 for i := 0 to 11 do
 begin
  yper:=sin(i*deg)*step;
  xper:=cos(i*deg)*step;
  MoveTo(centrex+round(1.2*xper),centrey+round(1.2*yper));
  LineTo(centrex+round(xper),centrey+round(yper));
 end;
 Font.Color:=GetColorFromIndex(LineColor);
 Font.Size:=8;
 TextOut(centrex-TextWidth('S') div 2,centrey+round(1.2*step)+1,'S');
 TextOut(centrex-TextWidth('N') div 2,centrey-round(1.2*step)-TextHeight('N')-1,'N');
 TextOut(centrex+round(1.2*step)+1,centrey-TextHeight('E') div 2,'E');
 TextOut(centrex-round(1.2*step)-TextWidth('W')-1,centrey-TextHeight('W') div 2,'W');
 Font.Color:=GetColorFromIndex(LabelColor);
 deg:=pi*30/180;
```

```
if not isSmall1 then Begin
y :=centrey+round(1.2*sin(deg-pi/2)*step)-TextHeight('30')-1;
x :=centrex+round(1.2*cos(deg-pi/2)*step)+1;
Font.Size:=8;
TextOut(x,y,'30');
Font.Size:=4;
TextOut(x+TextWidth('30'),y-TextHeight('30')+5,'o');
end;
y :=centrey+round(1.2*sin(2*deg-pi/2)*step)-TextHeight('60')-1;
x :=centrex+round(1.2*cos(2*deg-pi/2)*step)+1;
Font.Size:=8;
TextOut(x,y,'60');
Font.Size:=4;
TextOut(x+TextWidth('60'),y-TextHeight('60')+5,'o');
if not isSmall1 then Begin
y :=centrey+round(1.2*sin(4*deg-pi/2)*step)+1;
x :=centrex+round(1.2*cos(4*deg-pi/2)*step)+1;
Font.Size:=8;
TextOut(x,y,'120');
Font.Size:=4;
TextOut(x+TextWidth('120'),y-TextHeight('120')+5,'o');
end;
y :=centrey+round(1.2*sin(5*deg-pi/2)*step)+2;
x :=centrex+round(1.2*cos(5*deg-pi/2)*step)+1;
Font.Size:=8;
TextOut(x,y,'150');
Font.Size:=4;
TextOut(x+TextWidth('150'),y-TextHeight('150')+5,'o');
y :=centrey+round(1.2*sin(7*deg-pi/2)*step)+4;
x :=centrex+round(1.2*cos(7*deg-pi/2)*step)-TextWidth('210')-1;
Font.Size:=8;
TextOut(x,y,'210');
Font.Size:=4;
TextOut(x+TextWidth('210'),y-TextHeight('210')+5,'o');
if not isSmall1 then Begin
y :=centrey+round(1.2*sin(8*deg-pi/2)*step)+1;
x :=centrex+round(1.2*cos(8*deg-pi/2)*step)-TextWidth('240');
Font.Size:=8;
TextOut(x,y,'240');
Font.Size:=4;
TextOut(x+TextWidth('240'),y-TextHeight('240')+5,'o');
end;
y :=centrey+round(1.2*sin(10*deg-pi/2)*step)-TextHeight('300')-1;
x :=centrex+round(1.2*cos(10*deg-pi/2)*step)-TextWidth('300');
Font.Size:=8;
TextOut(x,y,'300');
Font.Size:=4;
TextOut(x+TextWidth('300'),y-TextHeight('300')+5,'o');
if not isSmall1 then Begin
y :=centrey+round(1.2*sin(11*deg-pi/2)*step)-TextHeight('330')-1;
x :=centrex+round(1.2*cos(11*deg-pi/2)*step)-TextWidth('330');
Font.Size:=8;
TextOut(x,y,'330');
Font.Size:=4;
TextOut(x+TextWidth('330'),y-TextHeight('330')+5,'o');
end;
Brush.Color:=GetColorFromIndex(DiagColor);
End;
end;
```

```
{*****************************************************************************}
procedure DrawStrelkaDirWind(Fi:Real;Col:TColor;Canvas1:TCanvas;Height1,Width1:Integer);
var
  step,centrex,centrey,si,co,si0,co0:Integer;
  xper,yper:Real;
begin
 With Canvas1 do Begin
 yper:=(Height1/2-TextHeight('S')-2)/1.2;
 xper:=(Width1/2-TextWidth('W')-4)/1.2;
 if xper<=yper then step:=round(xper)
            else step:=round(yper);
 centrex := Width1 div 2;
 centrey := Height1 div 2;
    Pen.Color:=Col;
    Pen.Width:=2;
    Brush.Color:=GetColorFromIndex(DiagColor);
    si0 := centrey+round(sin((Fi - 90) * pi / 180) *(Malrad1+2));
    co0 := centrex+round(cos((Fi - 90) * pi / 180) *(Malrad1+2));
    MoveTo(co0,si0);
    si := round(sin((Fi - 90) * pi / 180) *(step-2));
    co := round(cos((Fi - 90) * pi / 180) *(step-2));
    LineTo(centrex+co,centrey+si);
    si := round(sin((Fi - 80) * pi / 180) *step/4);
    co := round(cos((Fi - 80) * pi / 180) *step/4);
    MoveTo(co0,si0);
    LineTo(centrex+co,centrey+si);
    si := round(sin((Fi - 100) * pi / 180) * step/4);
    co := round(cos((Fi - 100) * pi / 180) * step/4);
    MoveTo(co0,si0);
    LineTo(centrex+co,centrey+si);
 End;
end;

{*****************************************************************************}
procedure DrawWinAmpl(a0,ad,xd:Integer;Canvas1:TCanvas; Height1,Width1:Integer);
var
  i,step,centrex,centrey,x,y,th,tw,am:Integer;
  xper,yper,fi,si,co:Real;
  R:TRect;
  st:String;
begin
 ClearGraphWindow(Canvas1,Width1,Height1);
 With Canvas1 do Begin
 th:=TextHeight('8');
 tw:=TextWidth('888');
 yper:=(Height1-Th-2)/1.2-Th-2;
 xper:=(Width1/2-TW-4)/1.2;
 if xper<=yper then step:=round(xper)
            else step:=round(yper);
 centrex := Width1 div 2;
 centrey := Height1 - Th-2;
 Pen.Color:=GetColorFromIndex(LabelColor);
    Pen.Width:=1;
    Brush.Color:=GetColorFromIndex(DiagColor);
 Ellipse(centrex-step,centrey-step,centrex+step,centrey+step);
  Brush.Color:=GetColorFromIndex(WindowColor);
    R:=Rect(0,centrey+1,Width1,Height1);
    FillRect(R);
 Brush.Color:=GetColorFromIndex(ArrowColor);
 Ellipse(centrex-Malrad1,centrey-Malrad1,centrex+Malrad1,centrey+Malrad1);
```

```
 Brush.Color:=GetColorFromIndex(WindowColor);
  for i:=0 to 20{6} do Begin
   fi:=(i-10)*30/xd*ad;
   if (fi>-90.5) and (fi<90.5) then begin
   si := sin((Fi - 90) * pi / 180)*step;
   co := cos((Fi - 90) * pi / 180)*step;
   MoveTo(centrex+round(1.1*co),centrey+round(1.1*si));
   LineTo(centrex+round(co),centrey+round(si));
   end;
   End;
  Font.Color:=GetColorFromIndex(LabelColor);
  for i:=0 to 6 do Begin
   Am:=a0+(i-3)*ad;
   fi:=(i-3)*90/xd*ad;
   si := sin((Fi - 90) * pi / 180)*step;
   co := cos((Fi - 90) * pi / 180)*step;
   MoveTo(centrex+round(1.2*co),centrey+round(1.2*si));
   LineTo(centrex+round(co),centrey+round(si));
   st:=IntToStr(Am);
   th:=TextHeight(st);
   tw:=TextWidth(st);
   case i of
    0:begin x:=-tw-1; y:=-th div 2; end;
    1:begin x:=-tw; y:=-th-1; end;
    2:begin x:=-tw; y:=-th-1; end;
    3:begin x:=-tw div 2; y:=-th-1; end;
    4:begin x:=1; y:=-th-1; end;
    5:begin x:=1; y:=-th-1; end;
    6:begin x:=1; y:=-th div 2; end;
    else begin x:=0; y:=0; end;
   end;
   TextOut(centrex+round(1.2*co)+x,centrey+round(1.2*si)+y,st);
  End;
 Brush.Color:=GetColorFromIndex(DiagColor);
 End;
end;

{*****************************************************************************}
procedure DrawStrelkaAmpl(Am:Real;a0,xd:Integer;Col:TColor;
                          Canvas1:TCanvas;Height1,Width1:Integer);
var
  step,centrex,centrey,si,co,th,tw,si0,co0:Integer;
  xper,yper,Fi:Real;
begin
 With Canvas1 do Begin
 th:=TextHeight('8');
 tw:=TextWidth('888');
 yper:=(Height1-Th-2)/1.2-Th-2;
 xper:=(Width1/2-TW-4)/1.2;
 if xper<=yper then step:=round(xper)
              else step:=round(yper);
 centrex := Width1 div 2;
 centrey := Height1 - Th-2;
    Pen.Color:=Col;
    Pen.Width:=2;
    fi:=90*(Am-a0)/xd;
    si0 := centrey+round(sin((Fi - 90) * pi / 180) *Malrad1);
    co0 := centrex+round(cos((Fi - 90) * pi / 180) *Malrad1);
    MoveTo(co0,si0);
    si0 :=centrey+round(sin((Fi - 90) * pi / 180) *(step-2));
```

```
    co0 :=centrex+round(cos((Fi - 90) * pi / 180) *(step-2));
    LineTo(co0,si0);
    xper:=0.9*(step-2);
    si := round(sin((Fi - 87) * pi / 180) *xper);
    co := round(cos((Fi - 87) * pi / 180) *xper);
    MoveTo(co0,si0);
    LineTo(centrex+co,centrey+si);
    si := round(sin((Fi - 93) * pi / 180) *xper);
    co := round(cos((Fi - 93) * pi / 180) *xper);
    MoveTo(co0,si0);
    LineTo(centrex+co,centrey+si);
  End;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT Izmeren;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, Main;

type
  TIzmerenForm = class(TForm)
    AnalDatchBox: TGroupBox;
    LabPmm: TLabel;
    LabHu: TLabel;
    EditP: TEdit;
    EditU: TEdit;
    HorWindBox: TGroupBox;
    FiPanel: TPanel;
    PaintFi: TPaintBox;
    SonicBox: TGroupBox;
    LabT: TLabel;
    LabVh: TLabel;
    LabF: TLabel;
    LabVz: TLabel;
    PanelT: TPanel;
    PaintT: TPaintBox;
    EditT: TEdit;
    PanelVh: TPanel;
    PaintVh: TPaintBox;
    PanelF: TPanel;
    PaintF: TPaintBox;
    PanelVz: TPanel;
    PaintVz: TPaintBox;
    EditVh: TEdit;
    EditF: TEdit;
    EditVz: TEdit;
    Label4: TLabel;
    LabFdop: TLabel;
    DopolnBox: TGroupBox;
    LabQ: TLabel;
    LabQdop: TLabel;
    LabVx: TLabel;
    LabVy: TLabel;
    EditQ: TEdit;
```

```
    EditVx: TEdit;
    EditVy: TEdit;
    EditGx: TEdit;
    EditGy: TEdit;
    LabGx: TLabel;
    LabGxdop: TLabel;
    LabGy: TLabel;
    Label1: TLabel;
    CloseBtn: TBitBtn;
    EditTr: TEdit;
    LabTr: TLabel;
    Label2: TLabel;
    procedure UpdateVidGraph(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure CloseBtnClick(Sender: TObject);
    procedure PaintTPaint(Sender: TObject);
    procedure PaintVhPaint(Sender: TObject);
    procedure PaintFPaint(Sender: TObject);
    procedure PaintVzPaint(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure PaintFiPaint(Sender: TObject);
  private
    { Private declarations }
    Xstart,Ystart:Integer;
    MashX,FhorOld:Real;
    MashY:array [0..3] of Real;
    AmStart,RayM:array [0..3] of Integer;
    UserLineColor:TColor;
    procedure ClearDigits;
    procedure OutDigits;
    procedure DrawDirWind(Fi:Real);
    procedure UpdateEditFont;
    procedure UpdateGraphics;
    procedure DrawSetka(TheCanvas:TCanvas; Width,Height:Integer);
    procedure DrawMashtab(Num1:Byte; TheCanvas:TCanvas;Width,Height:Integer);
    procedure DrawLine(Num1:Byte;k:Integer;TheCanvas:TCanvas);
    procedure Redraw(Num1:Byte;TheCanvas:TCanvas);
    procedure FormClear;
  public
    { Public declarations }
    procedure DrawLines(Line1:Integer);
  end;

var
  IzmerenForm: TIzmerenForm;

implementation
{$R *.DFM}
uses UnAMKini,UnData,UnDevice,GraphLib;

{*********************************}
procedure TIzmerenForm.FormResize(Sender: TObject);
begin
  SonicBox.Height:=Height-38;
  SonicBox.Width:=Width-146;
  PanelT.Width:=SonicBox.Width-64;
```

```
  PanelVh.Width:=PanelT.Width;
  PanelF.Width:=PanelT.Width;
  PanelVz.Width:=PanelT.Width;
  PanelT.Height:=(SonicBox.Height-38) div 4;
  PanelVh.Height:=PanelT.Height;
  PanelF.Height:=PanelT.Height;
  PanelVz.Height:=PanelT.Height;
  PanelVh.Top:=PanelT.Top+PanelT.Height+5;
  PanelF.Top:=PanelVh.Top+PanelVh.Height+5;
  PanelVz.Top:=PanelF.Top+PanelF.Height+5;
  EditT.Top:=PanelT.Top+37;
  EditVh.Top:=PanelVh.Top+37;
  EditF.Top:=PanelF.Top+37;
  EditVz.Top:=PanelVz.Top+37;
  LabT.Top:=EditT.Top-19;
  Label4.Left:=LabT.Left+12;
  Label4.Top:=LabT.Top-5;
  LabVh.Top:=EditVh.Top-19;
  LabF.Top:=EditF.Top-19;
  LabFdop.Left:=LabF.Left+10;
  LabFdop.Top:=LabF.Top-5;
  LabVz.Top:=EditVz.Top-19;
end;

{*********************************}
procedure TIzmerenForm.UpdateEditFont;
var
  Col1,Col2:TColor;
begin
  Col1:=MainForm.Edit1.Color;
  Col2:=MainForm.Edit1.Font.Color;
  EditVx.Color:=Col1;
  EditVx.Font.Color:=Col2;
  EditVy.Color:=Col1;
  EditVy.Font.Color:=Col2;
  EditQ.Color:=Col1;
  EditQ.Font.Color:=Col2;
  EditT.Color:=Col1;
  EditT.Font.Color:=Col2;
  EditVh.Color:=Col1;
  EditVh.Font.Color:=Col2;
  EditF.Color:=Col1;
  EditF.Font.Color:=Col2;
  EditVz.Color:=Col1;
  EditVz.Font.Color:=Col2;
  if IsDeviceP then begin
  EditP.Color:=Col1;
  EditP.Font.Color:=Col2;
  end else begin
  EditP.Color:=clSilver;
  EditP.Font.Color:=clBlack;
  end;
  if IsDeviceHu then begin
  EditU.Color:=Col1;
  EditU.Font.Color:=Col2;
  EditTr.Color:=Col1;
  EditTr.Font.Color:=Col2;
  end else begin
  EditU.Color:=clSilver;
  EditU.Font.Color:=clBlack;
```

```
  EditTr.Color:=clSilver;
  EditTr.Font.Color:=clBlack;
  end;
  EditGx.Color:=Col1;
  EditGx.Font.Color:=Col2;
  EditGy.Color:=Col1;
  EditGy.Font.Color:=Col2;
end;

{*********************************}
procedure TIzmerenForm.UpdateGraphics;
var
  Col1:TColor;
begin
  Col1:=GetColorFromIndex(WindowColor);
  PanelT.Color:=Col1;
  PaintT.Color:=Col1;
  PanelVh.Color:=Col1;
  PaintVh.Color:=Col1;
  PanelF.Color:=Col1;
  PaintF.Color:=Col1;
  PanelVz.Color:=Col1;
  PaintVz.Color:=Col1;
  FiPanel.Color:=Col1;
  PaintFi.Color:=Col1;
  UserLineColor:=GetColorFromIndex(LineColor);
  RayM[0]:=rayTC; RayM[1]:=rayVg;
  RayM[2]:=rayFi; RayM[3]:=rayVz;
  UpdateEditFont;
  labT.Caption:=MainForm.label3b.Caption;
  labVh.Caption:=MainForm.label1b.Caption;
  labF.Caption:=MainForm.label2b.Caption;
  labVz.Caption:=MainForm.label4b.Caption;
  LabPmm.Caption:=MainForm.label5b.Caption;
  if SizeDavlen<>1 then LabPmm.Caption:='     '+LabPmm.Caption;
  labHu.Caption:=MainForm.label6b.Caption;
end;

{*********************************}
procedure TIzmerenForm.FormCreate(Sender: TObject);
begin
  Top:=0;
  Height:=Screen.Height-32;
  Left:=round(0.33*Screen.Width);
  Width:=Screen.Width-Left;
  Xstart:=22;
  FhorOld:=0;
  UpdateGraphics;
end;

{*********************************}
procedure TIzmerenForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  with MainForm do
  if RegimVabor.ItemIndex=0 then ViewBtn.Enabled:=true;
  Action:=caHide;
end;

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

```
procedure TIzmerenForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then CloseBtnClick(Sender);
end;

{*********************************}
procedure TIzmerenForm.CloseBtnClick(Sender: TObject);
begin
  Close;
end;

{*********************************}
procedure TIzmerenForm.ClearDigits;
begin
  EditT.Text:='';
  EditVh.Text:='';
  EditF.Text:='';
  EditVz.Text:='';
  EditP.Text:='';
  EditU.Text:='';
  EditQ.Text:='';
  EditVx.Text:='';
  EditVy.Text:='';
  EditGx.Text:='';
  EditGy.Text:='';
  EditTr.Text:='';
end;

{*********************************}
procedure TIzmerenForm.OutDigits;
begin
  if Met_Data=nil then
    ClearDigits else Begin
  EditT.Text:=Format('%6.2f',[TC]);
  EditVh.Text:=Format('%6.2f',[Vhor]);
  EditF.Text:=Format('%6.1f',[Fhor]);
  EditVz.Text:=Format('%6.2f',[Vz]);
  if SizeDavlen=1 then EditP.Text:=Format('%6.1f',[MMdavl])
              else EditP.Text:=Format('%6.1f',[MMdavl/0.75]);
  EditU.Text:=Format('%6.1f',[OtVlag]);
  EditQ.Text:=Format('%6.2f',[AVlag]);
  EditVx.Text:=Format('%6.2f',[Vx]);
  EditVy.Text:=Format('%6.2f',[Vy]);
  EditGx.Text:=Format('%6.1f',[GammaX]);
  EditGy.Text:=Format('%6.1f',[GammaY]);
  if IsDeviceHu then EditTr.Text:=Format('%5.1f',[DatTC])
            else EditTr.Text:='';
  end;
end;

{*********************************}
procedure TIzmerenForm.DrawDirWind(Fi:Real);
begin
  try
  if Met_Data<>nil then begin
  if Length(Met_Data)>1 then
  DrawStrelkaDirWind(Fhorold,GetColorFromIndex(DiagColor),
                PaintFi.Canvas,PaintFi.Height,PaintFi.Width);
  DrawStrelkaDirWind(Fi,GetColorFromIndex(ArrowColor),
```

```
                    PaintFi.Canvas,PaintFi.Height,PaintFi.Width);
  FhorOld:=Fi;
  end;
  except
  end;
end;

{*****************************************************************************}
procedure TIzmerenForm.DrawSetka(TheCanvas:TCanvas; Width,Height:Integer);
var
  i:Byte;
  xx:Real;
  x:Integer;
begin
  With TheCanvas do Begin
  Pen.Color:=GetColorFromIndex(LabelColor);
  Brush.Color:=GetColorFromIndex(WindowColor);
  Pen.Width:=1;
  Pen.Style:=psSolid;
  Pen.Style:=psDot;
  Pen.Color:=GetColorFromIndex(GridColor);
  Ystart:=Height div 2;
  MoveTo(Xstart,0);
  LineTo(Width,0);
  MoveTo(Xstart,Height div 4);
  LineTo(Width,Height div 4);
  MoveTo(Xstart,Ystart);
  LineTo(Width,Ystart);
  MoveTo(Xstart,round(3*Height/4));
  LineTo(Width,round(3*Height/4));
  xx:=(Width-Xstart)/10;
  for i:=1 to 10 do Begin
  if i=10 then x:=Width-1
        else x:=Xstart+round(i*xx);
  MoveTo(x,0);
  LineTo(x,Height);
  End; {i}
  Pen.Color:=GetColorFromIndex(LabelColor);
  Pen.Style:=psSolid;
  MoveTo(Xstart,0);
  LineTo(Xstart,Height);
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
  End;
end;

{*****************************************************************************}
procedure TIzmerenForm.DrawMashtab(Num1:Byte; TheCanvas:TCanvas;
                          Width,Height:Integer);
var
  st3:string[3];
  per,MaxColPoi:Integer;
begin
  try
  MaxColPoi:=round(15/TimeCikle)+2;
  if Met_Data<>nil then begin
  if (Length(Met_Data)>=3) and MainForm.BreakReadDataUGI then
    MaxColPoi:=Length(Met_Data);
    end;
  MashX:=(Width-Xstart)/(MaxColPoi-1);
```

```
  MashY[Num1]:=Height/RayM[Num1];
  case Num1 of
   0:AmStart[Num1]:=round(TC);
   1:begin
       AmStart[Num1]:=round(Vhor);
       per:=RayM[Num1] div 2;
       if AmStart[Num1]-per<0 then AmStart[Num1]:=per;
     end;
   2:begin
       AmStart[Num1]:=round(Fhor);
       per:=RayM[Num1] div 2;
       if (AmStart[Num1]+per>360) or (AmStart[Num1]-per<0)
       then AmStart[Num1]:=per;
     end;
   3:AmStart[Num1]:=round(Vz);
  end;
  With TheCanvas do Begin
  Pen.Color:=GetColorFromIndex(LabelColor);
  Brush.Color:=GetColorFromIndex(WindowColor);
  Pen.Width:=1;
  Pen.Style:=psSolid;
  Font.Size:=8;
  Font.Color:=GetColorFromIndex(LabelColor);
  MoveTo(Xstart,0);
  LineTo(Xstart,Height);
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
  MoveTo(Xstart,0);
  LineTo(Xstart-4,0);
  MoveTo(Xstart,Height div 4);
  LineTo(Xstart-4,Height div 4);
  MoveTo(Xstart,Ystart);
  LineTo(Xstart-4,Ystart);
  MoveTo(Xstart,round(3*Height/4));
  LineTo(Xstart-4,round(3*Height/4));
  MoveTo(Xstart,Height-1);
  LineTo(Xstart-4,Height-1);
  str(AmStart[Num1]:3,st3);
  TextOut(1,Ystart - 6,st3);
  per:=AmStart[Num1]-RayM[Num1] div 2;
  str(per:3,st3);
  TextOut(1,Height-13,st3);
  per:=AmStart[Num1]+RayM[Num1] div 2;
  str(per:3,st3);
  TextOut(1,0,st3);
  End;
  except
  raise;
  end;
end;

{*********************************}
procedure TIzmerenForm.DrawLine(Num1:Byte;k:Integer;TheCanvas:TCanvas);
var
  y1,y2,F1,F2:Real;
  MyColor:TColor;
  er1:Byte;
begin
  if Met_Data=nil then Exit;
  if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
```

```
if MainForm.BreakReadDataUGI then begin
  er1:=Met_Data[k].Sboj;
  case Num1 of
  0:begin
      y2:=Met_Data[k].Mas[Num1]-AmStart[Num1];
      if k>0 then y1:=Met_Data[k-1].Mas[Num1]-AmStart[Num1];
    end;
  1:begin
      CalcHorWind(Met_Data[k].Mas[1],Met_Data[k].Mas[2],y2,F2);
      y2:=y2-AmStart[Num1];
      if k>0 then begin
        CalcHorWind(Met_Data[k-1].Mas[1],Met_Data[k-1].Mas[2],y1,F1);
        y1:=y1-AmStart[Num1];
        end;
    end;
  2:begin
      CalcHorWind(Met_Data[k].Mas[1],Met_Data[k].Mas[2],y2,F2);
      y2:=F2-AmStart[Num1];
      F1:=F2;
      if k>0 then begin
        CalcHorWind(Met_Data[k-1].Mas[1],Met_Data[k-1].Mas[2],y1,F1);
        y1:=F1-AmStart[Num1];
        end;
    end;
  3:begin
      y2:=Met_Data[k].Mas[Num1]-AmStart[Num1];
      if k>0 then y1:=Met_Data[k-1].Mas[Num1]-AmStart[Num1];
    end;
   end;//case Num1
  end else begin
  er1:=Graf_Data[k].Sboj;
  case Num1 of
  0:begin
      y2:=Graf_Data[k].Mas[Num1]-AmStart[Num1];
      if k>0 then y1:=Graf_Data[k-1].Mas[Num1]-AmStart[Num1];
    end;
  1:begin
      CalcHorWind(Graf_Data[k].Mas[1],Graf_Data[k].Mas[2],y2,F2);
      y2:=y2-AmStart[Num1];
      if k>0 then begin
        CalcHorWind(Graf_Data[k-1].Mas[1],Graf_Data[k-1].Mas[2],y1,F1);
        y1:=y1-AmStart[Num1];
        end;
    end;
  2:begin
      CalcHorWind(Graf_Data[k].Mas[1],Graf_Data[k].Mas[2],y2,F2);
      y2:=F2-AmStart[Num1];
      F1:=F2;
      if k>0 then begin
        CalcHorWind(Graf_Data[k-1].Mas[1],Graf_Data[k-1].Mas[2],y1,F1);
        y1:=F1-AmStart[Num1];
        end;
    end;
  3:begin
      y2:=Graf_Data[k].Mas[Num1]-AmStart[Num1];
      if k>0 then y1:=Graf_Data[k-1].Mas[Num1]-AmStart[Num1];
    end;
   end;//case Num1
  end;
 case er1 of
```

```
    0:MyColor:=UserLineColor;
    1:MyColor:=UserLineColor;
    2:MyColor:=ClBlue;
    else MyColor:=UserLineColor+1;
   end;
  with TheCanvas do Begin
   if (LineStyle>4) or (k<2) then
   Pixels[Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1])]:=MyColor
 (**)  else begin
   Pen.Style:=GetStyleLineFromIndex(LineStyle);
   Pen.Color:=MyColor;
   Pen.Width:=LineTick;
   if (Num1=2) and (abs(F2-F1)>270) then begin
   Pixels[Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1])]:=MyColor;
   end else begin
   MoveTo(Xstart+round((k-1)*MashX),Ystart-round(y1*MashY[Num1]));
   LineTo(Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1]));
   end; {if (Num1=2) and (abs(F2-F1)>270)}
   end; {if (k<2)}(**)
  End; {with TheCanvas}
 end;

 {*********************************}
 procedure TIzmerenForm.DrawLines(Line1:Integer);
 begin
   try
   if Met_Data<>nil then begin
   OutDigits;
   DrawDirWind(Fhor);
   if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
   DrawLine(0,Line1,PaintT.Canvas);
   DrawLine(1,Line1,PaintVh.Canvas);
   DrawLine(2,Line1,PaintF.Canvas);
   DrawLine(3,Line1,PaintVz.Canvas);
   end;
   except
   raise
   end;
 end;

 {*************************************}
 procedure TIzmerenForm.Redraw(Num1:Byte;TheCanvas:TCanvas);
 var
  k:Integer;
 Begin
  try
  if Met_Data=nil then Exit;
  if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
  k:=0;
  With TheCanvas do Begin
    if not MainForm.BreakReadDataUGI then begin
     while k<=High(Graf_Data) do
      begin
        DrawLine(Num1,k,TheCanvas);
        Inc(k);
      end;
    end else begin
     while k<=High(Met_Data) do
      begin
        DrawLine(Num1,k,TheCanvas);
```

```
        Inc(k);
      end;
    end;
  End;
  except;
  raise;
  end;
 End;

 {*********************************}
 procedure TIzmerenForm.PaintTPaint(Sender: TObject);
 begin
   ClearGraphWindow(PaintT.Canvas,PaintT.Width,PaintT.Height);
   if IsOutGrid then DrawSetka(PaintT.Canvas,PaintT.Width,PaintT.Height);
   DrawMashtab(0,PaintT.Canvas,PaintT.Width,PaintT.Height);
   Redraw(0,PaintT.Canvas);
 end;

 {*********************************}
 procedure TIzmerenForm.PaintVhPaint(Sender: TObject);
 begin
   ClearGraphWindow(PaintVh.Canvas,PaintVh.Width,PaintVh.Height);
   if IsOutGrid then DrawSetka(PaintVh.Canvas,PaintVh.Width,PaintVh.Height);
   DrawMashtab(1,PaintVh.Canvas,PaintVh.Width,PaintVh.Height);
   Redraw(1,PaintVh.Canvas);
 end;

 {*********************************}
 procedure TIzmerenForm.PaintFPaint(Sender: TObject);
 begin
   ClearGraphWindow(PaintF.Canvas,PaintF.Width,PaintF.Height);
   if IsOutGrid then DrawSetka(PaintF.Canvas,PaintF.Width,PaintF.Height);
   DrawMashtab(2,PaintF.Canvas,PaintF.Width,PaintF.Height);
   Redraw(2,PaintF.Canvas);
 end;

 {*********************************}
 procedure TIzmerenForm.PaintVzPaint(Sender: TObject);
 begin
   ClearGraphWindow(PaintVz.Canvas,PaintVz.Width,PaintVz.Height);
   if IsOutGrid then DrawSetka(PaintVz.Canvas,PaintVz.Width,PaintVz.Height);
   DrawMashtab(3,PaintVz.Canvas,PaintVz.Width,PaintVz.Height);
   Redraw(3,PaintVz.Canvas);
 end;

 {*********************************}
 procedure TIzmerenForm.FormClear;
 begin
   ClearGraphWindow(PaintT.Canvas,PaintT.Width,PaintT.Height);
   ClearGraphWindow(PaintVh.Canvas,PaintVh.Width,PaintVh.Height);
   ClearGraphWindow(PaintF.Canvas,PaintF.Width,PaintF.Height);
   ClearGraphWindow(PaintVz.Canvas,PaintVz.Width,PaintVz.Height);
   ClearGraphWindow(PaintFi.Canvas,PaintFi.Width,PaintFi.Height);
 end;

 {*********************************}
 procedure TIzmerenForm.FormPaint(Sender: TObject);
 begin
   try
   UpdateGraphics;
```

```
  OutDigits;
  PaintTPaint(Sender);
  PaintVhPaint(Sender);
  PaintFPaint(Sender);
  PaintVzPaint(Sender);
  PaintFiPaint(Sender);
  except
  raise
  end;
end;

{*********************************}
procedure TIzmerenForm.PaintFiPaint(Sender: TObject);
begin
  try
  ClearGraphWindow(PaintFi.Canvas,PaintFi.Width,PaintFi.Height);
  DrawWinDirWind(PaintFi.Canvas,PaintFi.Height,PaintFi.Width,true);
  DrawDirWind(Fhor);
  except
  end;
end;

{****************************************}
procedure TIzmerenForm.UpdateVidGraph(Sender: TObject);
begin
  UpdateGraphics;
  FormClear;
  FormPaint(Sender);
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT Calibrat;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, Main;

type
  TCalibUGIForm = class(TForm)
    SonicBox: TGroupBox;
    LabT: TLabel;
    LabVh: TLabel;
    LabF: TLabel;
    LabVz: TLabel;
    PanelT: TPanel;
    PaintS1: TPaintBox;
    EditS1: TEdit;
    PanelVh: TPanel;
    PaintS2: TPaintBox;
    PanelF: TPanel;
    PaintS3: TPaintBox;
    PanelVz: TPanel;
    PaintS4: TPaintBox;
    EditS2: TEdit;
    EditS3: TEdit;
    EditS4: TEdit;
```

```
    AcousticBox: TGroupBox;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    EditSpeed: TEdit;
    EditT1: TEdit;
    EditT2: TEdit;
    EditT3: TEdit;
    EditT4: TEdit;
    DopolnBox: TGroupBox;
    LabTcalibr: TLabel;
    EditQ: TEdit;
    EditTcalib: TEdit;
    Label2: TLabel;
    CloseBtn: TBitBtn;
    LabQ: TLabel;
    LabQdop: TLabel;
    AnalDatchBox: TGroupBox;
    LabPmm: TLabel;
    LabHu: TLabel;
    LabGx: TLabel;
    LabGxdop: TLabel;
    LabGy: TLabel;
    Label1: TLabel;
    LabTr: TLabel;
    Label3: TLabel;
    EditP: TEdit;
    EditU: TEdit;
    EditGx: TEdit;
    EditGy: TEdit;
    EditTr: TEdit;
    procedure UpdateVidGraph(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormPaint(Sender: TObject);
    procedure PaintS1Paint(Sender: TObject);
    procedure PaintS2Paint(Sender: TObject);
    procedure PaintS3Paint(Sender: TObject);
    procedure PaintS4Paint(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure CloseBtnClick(Sender: TObject);
  private
    { Private declarations }
    Xstart,Ystart:Integer;
    MashX:Real;
    MashY:array [0..3] of Real;
    AmStart,RayM:array [0..3] of Integer;
    UserLineColor:TColor;
    procedure ClearDigits;
    procedure OutDigits;
    procedure UpDateForm;
    procedure UpdateEditFont;
    procedure DrawSetka(TheCanvas:TCanvas; Width,Height:Integer);
    procedure DrawMashtab(Num1:Byte; TheCanvas:TCanvas;Width,Height:Integer);
    procedure DrawLine(Num1:Byte;k:Integer;TheCanvas:TCanvas);
    procedure Redraw(Num1:Byte;TheCanvas:TCanvas);
```

```
    procedure FormClear;
  public
    { Public declarations }
    procedure UpdateGraphics;
    procedure DrawLines(Line1:Integer);
  end;

var
  CalibUGIForm: TCalibUGIForm;

implementation
{$R *.DFM}
uses UnAMKini,UnData,UnDevice,GraphLib;

{*********************************}
procedure TCalibUGIForm.UpDateForm;
begin
    SonicBox.Caption:=Capt_SonicBox_1;
    LabT.Caption:=MainForm.label1b.Caption;
    labVh.Caption:=MainForm.label2b.Caption;
    labF.Caption:=MainForm.label3b.Caption;
    labVz.Caption:=MainForm.label4b.Caption;
    LabPmm.Caption:=MainForm.label5b.Caption;
    if SizeDavlen<>1 then LabPmm.Caption:='     '+LabPmm.Caption;
end;

{*********************************}
procedure TCalibUGIForm.UpdateEditFont;
var
  Col1,Col2:TColor;
begin
  Col1:=GetColorFromIndex(EditColor);
  Col2:=GetColorFromIndex(DigitColor);
  if IsDeviceP then begin
  EditP.Color:=Col1;
  EditP.Font.Color:=Col2;
  end else begin
  EditP.Color:=clSilver;
  EditP.Font.Color:=clBlack;
  end;
  if IsDeviceHu then begin
  EditU.Color:=Col1;
  EditU.Font.Color:=Col2;
  EditTr.Color:=Col1;
  EditTr.Font.Color:=Col2;
  end else begin
  EditU.Color:=clSilver;
  EditU.Font.Color:=clBlack;
  EditTr.Color:=clSilver;
  EditTr.Font.Color:=clBlack;
  end;
  EditGx.Color:=Col1;
  EditGx.Font.Color:=Col2;
  EditGy.Color:=Col1;
  EditGy.Font.Color:=Col2;
  EditS1.Color:=Col1;
  EditS1.Font.Color:=Col2;
  EditS2.Color:=Col1;
  EditS2.Font.Color:=Col2;
  EditS3.Color:=Col1;
```

```
  EditS3.Font.Color:=Col2;
  EditS4.Color:=Col1;
  EditS4.Font.Color:=Col2;
  EditQ.Color:=Col1;
  EditQ.Font.Color:=Col2;
  EditSpeed.Color:=Col1;
  EditSpeed.Font.Color:=Col2;
  EditT1.Color:=Col1;
  EditT1.Font.Color:=Col2;
  EditT2.Color:=Col1;
  EditT2.Font.Color:=Col2;
  EditT3.Color:=Col1;
  EditT3.Font.Color:=Col2;
  EditT4.Color:=Col1;
  EditT4.Font.Color:=Col2;
end;

{*********************************}
procedure TCalibUGIForm.UpdateGraphics;
var
  Col1:TColor;
  i:Byte;
begin
  Col1:=GetColorFromIndex(WindowColor);
  PanelT.Color:=Col1;
  PaintS1.Color:=Col1;
  PanelVh.Color:=Col1;
  PaintS2.Color:=Col1;
  PanelF.Color:=Col1;
  PaintS3.Color:=Col1;
  PanelVz.Color:=Col1;
  PaintS4.Color:=Col1;
  UserLineColor:=GetColorFromIndex(LineColor);
  for i:=0 to 3 do
    RayM[i]:=RaySi;
  UpDateForm;
  UpdateEditFont;
end;

{*********************************}
procedure TCalibUGIForm.FormCreate(Sender: TObject);
begin
  Top:=0;
  Height:=Screen.Height-32;
  Left:=round(0.33*Screen.Width);
  Width:=Screen.Width-Left;
  Xstart:=36;
  UpdateGraphics;
end;

{*********************************}
procedure TCalibUGIForm.FormResize(Sender: TObject);
begin
  SonicBox.Height:=Height-38;
  SonicBox.Width:=Width-146;
  PanelT.Width:=SonicBox.Width-64;
  PanelVh.Width:=PanelT.Width;
  PanelF.Width:=PanelT.Width;
  PanelVz.Width:=PanelT.Width;
  PanelT.Height:=(SonicBox.Height-38) div 4;
```

```
  PanelVh.Height:=PanelT.Height;
  PanelF.Height:=PanelT.Height;
  PanelVz.Height:=PanelT.Height;
  PanelVh.Top:=PanelT.Top+PanelT.Height+5;
  PanelF.Top:=PanelVh.Top+PanelVh.Height+5;
  PanelVz.Top:=PanelF.Top+PanelF.Height+5;
  EditS1.Top:=PanelT.Top+37;
  EditS2.Top:=PanelVh.Top+37;
  EditS3.Top:=PanelF.Top+37;
  EditS4.Top:=PanelVz.Top+37;
  LabT.Top:=EditS1.Top-19;
  LabVh.Top:=EditS2.Top-19;
  LabF.Top:=EditS3.Top-19;
  LabVz.Top:=EditS4.Top-19;
end;

{*********************************}
procedure TCalibUGIForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  with MainForm do
  if RegimVabor.ItemIndex=1 then ViewBtn.Enabled:=true;
  Action:=caHide;
end;

{*********************************}
procedure TCalibUGIForm.ClearDigits;
begin
  EditS1.Text:='';
  EditS2.Text:='';
  EditS3.Text:='';
  EditS4.Text:='';
  EditP.Text:='';
  EditU.Text:='';
  EditGx.Text:='';
  EditGy.Text:='';
  EditQ.Text:='';
  EditSpeed.Text:='';
  EditT1.Text:='';
  EditT2.Text:='';
  EditT3.Text:='';
  EditT4.Text:='';
  EditTr.Text:='';
end;

{*********************************}
procedure TCalibUGIForm.OutDigits;
begin
  if Cal_Data=nil then ClearDigits
  else begin
  EditS1.Text:=Format('%7.3f',[Smm_C[0]]);
  EditS2.Text:=Format('%7.3f',[Smm_C[1]]);
  EditS3.Text:=Format('%7.3f',[Smm_C[2]]);
  EditS4.Text:=Format('%7.3f',[Smm_C[3]]);
  if SizeDavlen=1 then EditP.Text:=Format('%6.1f',[MMdavl_C])
              else EditP.Text:=Format('%6.1f',[MMdavl_C/0.75]);
  EditU.Text:=Format('%6.1f',[OtVlag_C]);
  if IsDeviceHu then EditTr.Text:=Format('%5.1f',[DatTC])
             else EditTr.Text:='';
  EditGx.Text:=Format('%6.1f',[GammaX]);
```

```
  EditGy.Text:=Format('%6.1f',[GammaY]);
  EditQ.Text:=Format('%6.2f',[AbsVlag_C]);
  EditTcalib.Text:=Format('%6.2f',[TkalibrNew]);
  EditSpeed.Text:=Format('%9.5f',[Speed_C]);
  EditT1.Text:=Format('%9.5f',[tmks_C[0]]);
  EditT2.Text:=Format('%9.5f',[tmks_C[1]]);
  EditT3.Text:=Format('%9.5f',[tmks_C[2]]);
  EditT4.Text:=Format('%9.5f',[tmks_C[3]]);
  end;
end;

{*****************************************************************************}
procedure TCalibUGIForm.DrawSetka(TheCanvas:TCanvas; Width,Height:Integer);
var
  i:Byte;
  xx:Real;
  x:Integer;
begin
  With TheCanvas do Begin
  Pen.Color:=GetColorFromIndex(GridColor);
  Brush.Color:=GetColorFromIndex(WindowColor);
  Pen.Width:=1;
  Pen.Style:=psDot;
  Ystart:=Height div 2;
  MoveTo(Xstart,0);
  LineTo(Width,0);
  MoveTo(Xstart,Height div 4);
  LineTo(Width,Height div 4);
  MoveTo(Xstart,Ystart);
  LineTo(Width,Ystart);
  MoveTo(Xstart,round(3*Height/4));
  LineTo(Width,round(3*Height/4));
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
  xx:=(Width-Xstart)/10;
  for i:=1 to 10 do Begin
  if i=10 then x:=Width-1
        else x:=Xstart+round(i*xx);
  MoveTo(x,0);
  LineTo(x,Height);
  End; {i}
  Pen.Color:=GetColorFromIndex(LabelColor);
  Pen.Style:=psSolid;
  MoveTo(Xstart,0);
  LineTo(Xstart,Height);
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
  End;
end;

{*****************************************************************************}
procedure TCalibUGIForm.DrawMashtab(Num1:Byte; TheCanvas:TCanvas;
                          Width,Height:Integer);
var
  st5:string[5];
  per,MaxColPoi:Integer;
begin
  try
  MaxColPoi:=round(15/TimeCikle)+2;
  if Cal_Data<>nil then begin
```

```
  if (Length(Cal_Data)>=3) and MainForm.BreakReadDataUGI then
    MaxColPoi:=Length(Cal_Data);
    end;
  MashX:=(Width-Xstart)/(MaxColPoi-1);
  MashY[Num1]:=Height/RayM[Num1];
  AmStart[Num1]:=round(Smm_C[Num1]);
  With TheCanvas do Begin
  Pen.Color:=GetColorFromIndex(LabelColor);
  Brush.Color:=GetColorFromIndex(WindowColor);
  Pen.Width:=1;
  Pen.Style:=psSolid;
  Font.Size:=8;
  Font.Color:=GetColorFromIndex(LabelColor);
  MoveTo(Xstart,0);
  LineTo(Xstart,Height);
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
  MoveTo(Xstart,0);
  LineTo(Xstart-4,0);
  MoveTo(Xstart,Height div 4);
  LineTo(Xstart-4,Height div 4);
  MoveTo(Xstart,Ystart);
  LineTo(Xstart-4,Ystart);
  MoveTo(Xstart,round(3*Height/4));
  LineTo(Xstart-4,round(3*Height/4));
  MoveTo(Xstart,Height-1);
  LineTo(Xstart-4,Height-1);
  str(AmStart[Num1]:5,st5);
  TextOut(1,Ystart - 6,st5);
  per:=AmStart[Num1]-RayM[Num1] div 2;
  str(per:5,st5);
  TextOut(1,Height-13,st5);
  per:=AmStart[Num1]+RayM[Num1] div 2;
  str(per:5,st5);
  TextOut(1,0,st5);
  End;
  except
  raise;
  end;
end;

{*********************************}
procedure TCalibUGIForm.DrawLine(Num1:Byte;k:Integer;TheCanvas:TCanvas);
var
  a1,a2,y1,y2:Real;
  er1:Byte;
  MyColor:TColor;
begin
  if Cal_Data=nil then Exit;
  if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
  if MainForm.BreakReadDataUGI then begin
    er1:=Cal_Data[k].Sboj;
    a2:=Cal_Data[k].Mas[Num1];
    if k>0 then a1:=Cal_Data[k-1].Mas[Num1] else a1:=0;
    end else begin
    er1:=Graf_Data[k].Sboj;
    a2:=Graf_Data[k].Mas[Num1];
    if k>0 then a1:=Graf_Data[k-1].Mas[Num1] else a1:=0;
    end;
  case er1 of
```

```
    0:MyColor:=UserLineColor;
    1:MyColor:=UserLineColor;
    2:MyColor:=ClBlue;
    else MyColor:=UserLineColor+1;
   end;
  with TheCanvas do Begin
   y2:=a2-AmStart[Num1];
   if (LineStyle>4) or (k=0) then
   Pixels[Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1])]:=MyColor
   else begin
   Pen.Style:=GetStyleLineFromIndex(LineStyle);
   Pen.Color:=MyColor;
   Pen.Width:=LineTick;
   y1:=a1-AmStart[Num1];
   MoveTo(Xstart+round((k-1)*MashX),Ystart-round(y1*MashY[Num1]));
   LineTo(Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1]));
   end; {if (k<2)}
  End; {with TheCanvas}
 end;

 {*********************************}
 procedure TCalibUGIForm.DrawLines(Line1:Integer);
 begin
   try
   if Cal_Data<>nil then begin
   OutDigits;
   if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
   DrawLine(0,Line1,PaintS1.Canvas);
   DrawLine(1,Line1,PaintS2.Canvas);
   DrawLine(2,Line1,PaintS3.Canvas);
   DrawLine(3,Line1,PaintS4.Canvas);
   end;
   except
   raise
   end;
 end;

 {*************************************}
 procedure TCalibUGIForm.Redraw(Num1:Byte;TheCanvas:TCanvas);
 var
  k:Integer;
 Begin
  try
  if Cal_Data=nil then Exit;
  if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
  k:=0;
  With TheCanvas do Begin
    if not MainForm.BreakReadDataUGI then begin
     while k<=High(Graf_Data) do
      begin
        DrawLine(Num1,k,TheCanvas);
        Inc(k);
      end;
    end else begin
     while k<=High(Cal_Data) do
      begin
        DrawLine(Num1,k,TheCanvas);
        Inc(k);
      end;
    end;
```

```
 End;
 except;
 raise;
 end;
End;

{**************************************************}
procedure TCalibUGIForm.PaintS1Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  if IsOutGrid then DrawSetka(PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  DrawMashtab(0,PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  Redraw(0,PaintS1.Canvas);
end;

{**************************************************}
procedure TCalibUGIForm.PaintS2Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  if IsOutGrid then DrawSetka(PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  DrawMashtab(1,PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  Redraw(1,PaintS2.Canvas);
end;

{**************************************************}
procedure TCalibUGIForm.PaintS3Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  if IsOutGrid then DrawSetka(PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  DrawMashtab(2,PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  Redraw(2,PaintS3.Canvas);
end;

{**************************************************}
procedure TCalibUGIForm.PaintS4Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
  if IsOutGrid then DrawSetka(PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
  DrawMashtab(3,PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
  Redraw(3,PaintS4.Canvas);
end;

{*********************************}
procedure TCalibUGIForm.FormClear;
begin
  ClearGraphWindow(PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  ClearGraphWindow(PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  ClearGraphWindow(PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  ClearGraphWindow(PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
end;

{**************************************************}
procedure TCalibUGIForm.FormPaint(Sender: TObject);
begin
  try
  UpdateGraphics;
  OutDigits;
  PaintS1Paint(Sender);
  PaintS2Paint(Sender);
  PaintS3Paint(Sender);
```

```
  PaintS4Paint(Sender);
  except
  raise
  end;
end;

{****************************************}
procedure TCalibUGIForm.UpdateVidGraph(Sender: TObject);
begin
  UpdateGraphics;
  FormClear;
  UpDateForm;
  FormPaint(Sender);
end;

{*****************************************}
procedure TCalibUGIForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then CloseBtnClick(Sender);
end;

{******************************************}
procedure TCalibUGIForm.CloseBtnClick(Sender: TObject);
begin
  Close;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit Control;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, Main;

type
  TControlUGIForm = class(TForm)
    SonicBox: TGroupBox;
    LabT: TLabel;
    LabVh: TLabel;
    LabF: TLabel;
    LabVz: TLabel;
    PanelT: TPanel;
    PaintS1: TPaintBox;
    EditS1: TEdit;
    PanelVh: TPanel;
    PaintS2: TPaintBox;
    PanelF: TPanel;
    PaintS3: TPaintBox;
    PanelVz: TPanel;
    PaintS4: TPaintBox;
    EditS2: TEdit;
    EditS3: TEdit;
    EditS4: TEdit;
    AnalDatchBox: TGroupBox;
    LabX5: TLabel;
```

```
    LabX6: TLabel;
    LabGx: TLabel;
    LabGy: TLabel;
    EditX5: TEdit;
    EditX6: TEdit;
    EditGx: TEdit;
    EditGy: TEdit;
    SaveAsBtn: TBitBtn;
    SaveDialog1: TSaveDialog;
    EditX7: TEdit;
    EditX8: TEdit;
    EditX9: TEdit;
    EditX10: TEdit;
    LabX7: TLabel;
    LabX8: TLabel;
    LabX9: TLabel;
    LabX10: TLabel;
    CloseBtn: TBitBtn;
    procedure UpdateVidGraph(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormPaint(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure PaintS1Paint(Sender: TObject);
    procedure PaintS2Paint(Sender: TObject);
    procedure PaintS3Paint(Sender: TObject);
    procedure PaintS4Paint(Sender: TObject);
    procedure CloseBtnClick(Sender: TObject);
    procedure SaveAsBtnClick(Sender: TObject);
  private
    { Private declarations }
    Xstart,Ystart:Integer;
    MashX:Real;
    MashY:array [0..3] of Real;
    AmStart,RayM:array [0..3] of Integer;
    UserLineColor:TColor;
    procedure ClearDigits;
    procedure OutDigits;
    procedure UpdateEditFont;
    procedure UpdateGraphics;
    procedure DrawSetka(TheCanvas:TCanvas; Width,Height:Integer);
    procedure DrawMashtab(Num1:Byte; TheCanvas:TCanvas;Width,Height:Integer);
    procedure DrawLine(Num1:Byte;k:Integer;TheCanvas:TCanvas);
    procedure Redraw(Num1:Byte;TheCanvas:TCanvas);
    procedure FormClear;
    function  GetDataStrList(var Data1:TStringList):Boolean;
  public
    { Public declarations }
    procedure DrawLines(Line1:Integer);
  end;

var
  ControlUGIForm: TControlUGIForm;

implementation
{$R *.DFM}
uses UnAMKini,ApllDlgs,UnData,UnMeteo,GraphLib;
```

```
const
  LineIIIconval= '    N    X1    X2    X3    X4    X5    X6  X7/X8  X9/X8   X10   err';
  TitleContro_R = '*       Данные устройства       *';
  LineII_R1  = 'СРЕДНИЕ ЗНАЧЕНИЯ:';

{*********************************}
procedure TControlUGIForm.UpdateEditFont;
var
  Col1,Col2:TColor;
begin
  Col1:=GetColorFromIndex(EditColor);
  Col2:=GetColorFromIndex(DigitColor);
  if IsDeviceP then begin
  EditX5.Color:=Col1;
  EditX5.Font.Color:=Col2;
  end else begin
  EditX5.Color:=clSilver;
  EditX5.Font.Color:=clBlack;
  end;
  if IsDeviceHu then begin
  EditX6.Color:=Col1;
  EditX6.Font.Color:=Col2;
  EditX10.Color:=Col1;
  EditX10.Font.Color:=Col2;
  end else begin
  EditX6.Color:=clSilver;
  EditX6.Font.Color:=clBlack;
  EditX10.Color:=clSilver;
  EditX10.Font.Color:=clBlack;
  end;
  EditX7.Color:=Col1;
  EditX7.Font.Color:=Col2;
  EditX8.Color:=Col1;
  EditX8.Font.Color:=Col2;
  EditX9.Color:=Col1;
  EditX9.Font.Color:=Col2;
  EditGx.Color:=Col1;
  EditGx.Font.Color:=Col2;
  EditGy.Color:=Col1;
  EditGy.Font.Color:=Col2;
  EditS1.Color:=Col1;
  EditS1.Font.Color:=Col2;
  EditS2.Color:=Col1;
  EditS2.Font.Color:=Col2;
  EditS3.Color:=Col1;
  EditS3.Font.Color:=Col2;
  EditS4.Color:=Col1;
  EditS4.Font.Color:=Col2;
end;

{*********************************}
procedure TControlUGIForm.ClearDigits;
begin
  EditS1.Text:='';
  EditS2.Text:='';
  EditS3.Text:='';
  EditS4.Text:='';
  EditX5.Text:='';
  EditX6.Text:='';
  EditX7.Text:='';
```

```
  EditX8.Text:='';
  EditX9.Text:='';
  EditX10.Text:='';
  EditGx.Text:='';
  EditGy.Text:='';
  SaveAsBtn.Enabled:=false;
end;

{*********************************}
procedure TControlUGIForm.OutDigits;
begin
  if Con_Data=nil then ClearDigits else Begin
  SaveAsBtn.Enabled:=true;
  if MainForm.BreakReadDataUGI then begin
  EditS1.Text:=IntToStr(round(Con_Sred[0]));
  EditS2.Text:=IntToStr(round(Con_Sred[1]));
  EditS3.Text:=IntToStr(round(Con_Sred[2]));
  EditS4.Text:=IntToStr(round(Con_Sred[3]));
  if IsDeviceP then
    EditX5.Text:=Format('%6.1f',[Con_Sred[4]])
    else EditX5.Text:='';
  if IsDeviceHu then begin
    EditX6.Text:=Format('%6.1f',[Con_Sred[5]]);
    EditX10.Text:=Format('%6.1f',[Con_Sred[8]]);
    end else begin
    EditX6.Text:='';
    EditX10.Text:='';
    end;
    EditX7.Text:=IntToStr(Data_UGI[7]);
    EditX8.Text:=IntToStr(Data_UGI[8]);
    EditX9.Text:=IntToStr(Data_UGI[9]);
    if Data_Ugi[8]<>0 then begin
    EditGx.Text:=Format('%6.4f',[Con_Sred[6]]);
    EditGy.Text:=Format('%6.4f',[Con_Sred[7]]);
    end else begin
    EditGx.Text:='';
    EditGy.Text:='';
    end;
  end else begin
  EditS1.Text:=IntToStr(Data_UGI[1]);
  EditS2.Text:=IntToStr(Data_UGI[2]);
  EditS3.Text:=IntToStr(Data_UGI[3]);
  EditS4.Text:=IntToStr(Data_UGI[4]);
  if IsDeviceP then
    EditX5.Text:=IntToStr(Data_UGI[5])
    else EditX5.Text:='';
  if IsDeviceHu then begin
    EditX6.Text:=IntToStr(Data_UGI[6]);
    EditX10.Text:=IntToStr(Data_UGI[10]);
    end else begin
    EditX6.Text:='';
    EditX10.Text:='';
    end;
    EditX7.Text:=IntToStr(Data_UGI[7]);
    EditX8.Text:=IntToStr(Data_UGI[8]);
    EditX9.Text:=IntToStr(Data_UGI[9]);
    if Data_Ugi[8]<>0 then begin
    EditGx.Text:=Format('%6.4f',[Data_Ugi[7]/Data_Ugi[8]]);
    EditGy.Text:=Format('%6.4f',[Data_Ugi[9]/Data_Ugi[8]]);
    end else begin
```

```
      EditGx.Text:='';
      EditGy.Text:='';
      end;
    end;
    End;
  end;

  {*********************************}
  procedure TControlUGIForm.UpdateGraphics;
  var
    Col1:TColor;
    i:Byte;
  begin
    Col1:=GetColorFromIndex(WindowColor);
    PanelT.Color:=Col1;
    PaintS1.Color:=Col1;
    PanelVh.Color:=Col1;
    PaintS2.Color:=Col1;
    PanelF.Color:=Col1;
    PaintS3.Color:=Col1;
    PanelVz.Color:=Col1;
    PaintS4.Color:=Col1;
    UserLineColor:=GetColorFromIndex(LineColor);
    for i:=0 to 3 do
      RayM[i]:=Rayti;
    UpdateEditFont;
  end;

  {*****************************************************************************}
  procedure TControlUGIForm.DrawSetka(TheCanvas:TCanvas; Width,Height:Integer);
  var
    i:Byte;
    xx:Real;
    x:Integer;
  begin
    With TheCanvas do Begin
    Pen.Color:=GetColorFromIndex(GridColor);
    Brush.Color:=GetColorFromIndex(WindowColor);
    Pen.Width:=1;
    Pen.Style:=psDot;
    Ystart:=Height div 2;
    MoveTo(Xstart,0);
    LineTo(Width,0);
    MoveTo(Xstart,Height div 4);
    LineTo(Width,Height div 4);
    MoveTo(Xstart,Ystart);
    LineTo(Width,Ystart);
    MoveTo(Xstart,round(3*Height/4));
    LineTo(Width,round(3*Height/4));
    MoveTo(Xstart,Height-1);
    LineTo(Width,Height-1);
    xx:=(Width-Xstart)/10;
    for i:=0{1} to 10 do Begin
    if i=10 then x:=Width-1
          else x:=Xstart+round(i*xx);
    MoveTo(x,0);
    LineTo(x,Height);
    End; {i}
    Pen.Color:=GetColorFromIndex(LabelColor);
    Pen.Style:=psSolid;
```

```
  MoveTo(Xstart,0);
  LineTo(Xstart,Height);
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
   End;
end;

{*****************************************************************************}
procedure TControlUGIForm.DrawMashtab(Num1:Byte; TheCanvas:TCanvas;
                          Width,Height:Integer);
var
  st5:string[5];
  per,MaxColPoi:Integer;
begin
  try
  MaxColPoi:=round(15/TimeCikle)+2;
  if Con_Data<>nil then begin
  if (Length(Con_Data)>=3) and MainForm.BreakReadDataUGI then
    MaxColPoi:=Length(Con_Data);
    end;
  MashX:=(Width-Xstart)/(MaxColPoi-1);
  MashY[Num1]:=Height/RayM[Num1];
  AmStart[Num1]:=Data_UGI[Num1+1];
  With TheCanvas do Begin
  Pen.Color:=GetColorFromIndex(LabelColor);
  Brush.Color:=GetColorFromIndex(WindowColor);
  Pen.Width:=1;
  Pen.Style:=psSolid;
  Font.Size:=8;
  Font.Color:=GetColorFromIndex(LabelColor);
  MoveTo(Xstart,0);
  LineTo(Xstart,Height);
  MoveTo(Xstart,Height-1);
  LineTo(Width,Height-1);
  MoveTo(Xstart,0);
  LineTo(Xstart-4,0);
  MoveTo(Xstart,Height div 4);
  LineTo(Xstart-4,Height div 4);
  MoveTo(Xstart,Ystart);
  LineTo(Xstart-4,Ystart);
  MoveTo(Xstart,round(3*Height/4));
  LineTo(Xstart-4,round(3*Height/4));
  MoveTo(Xstart,Height-1);
  LineTo(Xstart-4,Height-1);
  str(AmStart[Num1]:5,st5);
  TextOut(1,Ystart - 6,st5);
  per:=AmStart[Num1]-RayM[Num1] div 2;
  str(per:5,st5);
  TextOut(1,Height-13,st5);
  per:=AmStart[Num1]+RayM[Num1] div 2;
  str(per:5,st5);
  TextOut(1,0,st5);
  End;
  except
  raise;
  end;
end;

{*********************************}
procedure TControlUGIForm.DrawLine(Num1:Byte;k:Integer;TheCanvas:TCanvas);
```

```
var
  a1,a2,y1,y2:Real;
  er1:Byte;
  MyColor:TColor;
begin
  if Con_Data=nil then Exit;
  if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
  if MainForm.BreakReadDataUGI then begin
    er1:=Con_Data[k].Sboj;
    a2:=Con_Data[k].Mas[Num1];
    if k>0 then a1:=Con_Data[k-1].Mas[Num1] else a1:=0;
    end else begin
    er1:=Graf_Data[k].Sboj;
    a2:=Graf_Data[k].Mas[Num1];
    if k>0 then a1:=Graf_Data[k-1].Mas[Num1] else a1:=0;
    end;
  case er1 of
   0:MyColor:=UserLineColor;
   1:MyColor:=UserLineColor;
   2:MyColor:=ClBlue;
   else MyColor:=UserLineColor+1;
  end;
 with TheCanvas do Begin
  y2:=a2-AmStart[Num1];
  if (LineStyle>4) or (k=0) then
  Pixels[Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1])]:=MyColor
  else begin
  Pen.Style:=GetStyleLineFromIndex(LineStyle);
  Pen.Color:=MyColor;
  Pen.Width:=LineTick;
  y1:=a1-AmStart[Num1];
  MoveTo(Xstart+round((k-1)*MashX),Ystart-round(y1*MashY[Num1]));
  LineTo(Xstart+round(k*MashX),Ystart-round(y2*MashY[Num1]));
  end; {if (k<2)}
 End; {with TheCanvas}
end;

{*********************************}
procedure TControlUGIForm.DrawLines(Line1:Integer);
begin
  try
  if Con_Data<>nil then begin
  OutDigits;
  if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
  DrawLine(0,Line1,PaintS1.Canvas);
  DrawLine(1,Line1,PaintS2.Canvas);
  DrawLine(2,Line1,PaintS3.Canvas);
  DrawLine(3,Line1,PaintS4.Canvas);
  end;
  except
  raise
  end;
end;

{*************************************}
procedure TControlUGIForm.Redraw(Num1:Byte;TheCanvas:TCanvas);
var
 k:Integer;
Begin
 try
```

```
 if Con_Data=nil then Exit;
 if (not MainForm.BreakReadDataUGI) and (Graf_Data=nil) then Exit;
 k:=0;
 With TheCanvas do Begin
   if not MainForm.BreakReadDataUGI then begin
    while k<=High(Graf_Data) do
     begin
       DrawLine(Num1,k,TheCanvas);
       Inc(k);
     end;
   end else begin
    while k<=High(Con_Data) do
     begin
       DrawLine(Num1,k,TheCanvas);
       Inc(k);
     end;
   end;
 End;
 except;
 raise;
 end;
End;

{*********************************}
procedure TControlUGIForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  with MainForm do
  if RegimVabor.ItemIndex=2 then ViewBtn.Enabled:=true;
  Action:=caHide;
end;

{*********************************}
procedure TControlUGIForm.FormCreate(Sender: TObject);
begin
  Top:=0;
  Height:=Screen.Height-32;
  Left:=round(0.33*Screen.Width);
  Width:=Screen.Width-Left;
  Xstart:=36;
  UpdateGraphics;
end;

{*********************************}
procedure TControlUGIForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then CloseBtnClick(Sender);
end;

{*********************************}
procedure TControlUGIForm.FormPaint(Sender: TObject);
begin
  try
  OutDigits;
  UpdateGraphics;
  PaintS1Paint(Sender);
  PaintS2Paint(Sender);
  PaintS3Paint(Sender);
  PaintS4Paint(Sender);
```

```
  except
  raise
  end;
end;

{*********************************}
procedure TControlUGIForm.FormResize(Sender: TObject);
begin
  SonicBox.Height:=Height-38;
  SonicBox.Width:=Width-146;
  PanelT.Width:=SonicBox.Width-64;
  PanelVh.Width:=PanelT.Width;
  PanelF.Width:=PanelT.Width;
  PanelVz.Width:=PanelT.Width;
  PanelT.Height:=(SonicBox.Height-38) div 4;
  PanelVh.Height:=PanelT.Height;
  PanelF.Height:=PanelT.Height;
  PanelVz.Height:=PanelT.Height;
  PanelVh.Top:=PanelT.Top+PanelT.Height+5;
  PanelF.Top:=PanelVh.Top+PanelVh.Height+5;
  PanelVz.Top:=PanelF.Top+PanelF.Height+5;
  EditS1.Top:=PanelT.Top+37;
  EditS2.Top:=PanelVh.Top+37;
  EditS3.Top:=PanelF.Top+37;
  EditS4.Top:=PanelVz.Top+37;
  LabT.Top:=EditS1.Top-19;
  LabVh.Top:=EditS2.Top-19;
  LabF.Top:=EditS3.Top-19;
  LabVz.Top:=EditS4.Top-19;
end;

{*********************************}
procedure TControlUGIForm.PaintS1Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  if IsOutGrid then DrawSetka(PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  DrawMashtab(0,PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  Redraw(0,PaintS1.Canvas);
end;

{*********************************}
procedure TControlUGIForm.PaintS2Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  if IsOutGrid then DrawSetka(PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  DrawMashtab(1,PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  Redraw(1,PaintS2.Canvas);
end;

{*********************************}
procedure TControlUGIForm.PaintS3Paint(Sender: TObject);
begin
  ClearGraphWindow(PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  if IsOutGrid then DrawSetka(PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  DrawMashtab(2,PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  Redraw(2,PaintS3.Canvas);
end;

{*********************************}
procedure TControlUGIForm.PaintS4Paint(Sender: TObject);
```

```
begin
  ClearGraphWindow(PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
  if IsOutGrid then DrawSetka(PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
  DrawMashtab(3,PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
  Redraw(3,PaintS4.Canvas);
end;

{*********************************}
procedure TControlUGIForm.FormClear;
begin
  ClearGraphWindow(PaintS1.Canvas,PaintS1.Width,PaintS1.Height);
  ClearGraphWindow(PaintS2.Canvas,PaintS2.Width,PaintS2.Height);
  ClearGraphWindow(PaintS3.Canvas,PaintS3.Width,PaintS3.Height);
  ClearGraphWindow(PaintS4.Canvas,PaintS4.Width,PaintS4.Height);
end;

{****************************************}
procedure TControlUGIForm.UpdateVidGraph(Sender: TObject);
begin
  UpdateGraphics;
  FormClear;
  FormPaint(Sender);
end;

{*********************************}
procedure TControlUGIForm.CloseBtnClick(Sender: TObject);
begin
  Close;
end;

{*****************************************************************************}
function TControlUGIForm.GetDataStrList(var Data1:TStringList):Boolean;
var
  st,st2:String;
  Hour1,Min1,Sec1,MSec1:Word;
  per:Real;
  i,k0:Integer;
begin
  try
  Result:=false;
  if Con_Data=nil then Exit;
  if Length(Con_Data)=0 then Exit;
  if GetSredGlobData(Con_Data,Con_Sred) then Begin
  with Data1 do Begin
  Add(LineTitle_R);
  Add(LineLider_R);
  Add(LineCaption_R);
  Add(TitleContro_R);
  Add(LineTitle_R);
  Add(LineIch_R);
  Add(LineI_R);
  Add(Format('%s%s',[DateTo_MyStr(Con_Data[NumPosGlobData].TDr),Title_Data]));
  Add(Format('%s%s',[TimeTo_MyStr(Con_Data[NumPosGlobData].TDr),Title_Time]));
  if NumPosGlobData=High(Con_Data) then k0:=0
                          else k0:=NumPosGlobData+1;
  DecodeTime(Con_Data[NumPosGlobData].TDr-Con_Data[k0].TDr,Hour1,Min1,Sec1,MSec1);
  per:=(3600*Hour1+60*Min1+Sec1+MSec1/1000)/60;
  str(per:8:5,st);
  Add(Format('%s%s',[st,Title_TimeUsred]));
  str(1/TimeCikle:8:3,st);
```

```
Add(Format('%s%s',[st,Title_TimeCikle]));
st:=Format('%8d',[Length(Con_Data)]);
Add(Format('%s%s',[st,Title_CountData]));
Add(LineIIch_R);
Add(LineII_R1);
for i:=0 to 5 do begin
str(Con_Sred[i]:8:2,st);
Add(Format('%s%s%s%s',[st,' - <X',IntToStr(i+1),'>']));
end;
str(Con_Sred[6]:8:4,st);
Add(Format('%s%s',[st,' - <X7/X8>']));
str(Con_Sred[7]:8:4,st);
Add(Format('%s%s',[st,' - <X7/X8>']));
str(Con_Sred[8]:8:2,st);
Add(Format('%s%s',[st,' - <X10>']));
Add(LineIIch_R);
Add(LineIII_R);
Add(LineIIIconval);
for i:=k0 to High(Con_Data) do Begin
st:=Format('%6d',[i+1]);
str(Con_Data[i].Mas[0]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[1]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[2]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[3]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[4]:6:1,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[5]:6:1,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[6]:6:4,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[7]:6:4,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[8]:6:1,st2);
st:=Format('%s%s%s',[st,' ',st2]);
st2:=Format('%3d',[Con_Data[i].Sboj]);
st:=Format('%s%s%s',[st,' ',st2]);
Add(st);
end;
if k0<>0 then begin
for i:=0 to NumPosGlobData do Begin
st:=Format('%6d',[i+1]);
str(Con_Data[i].Mas[0]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[1]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[2]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[3]:6:0,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[4]:6:1,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[5]:6:1,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[6]:6:4,st2);
st:=Format('%s%s%s',[st,' ',st2]);
str(Con_Data[i].Mas[7]:6:4,st2);
```

```
  st:=Format('%s%s%s',[st,' ',st2]);
  str(Con_Data[i].Mas[8]:6:1,st2);
  st:=Format('%s%s%s',[st,' ',st2]);
  st2:=Format('%3d',[Con_Data[i].Sboj]);
  st:=Format('%s%s%s',[st,' ',st2]);
  Add(st);
  end;
  end;
  End;
  Result:=true;
  End;
  except
  raise
  end;
end;

{*********************************************}
procedure TControlUGIForm.SaveAsBtnClick(Sender: TObject);
label 1;
var
  NewFile:TStringList;
  Msg:String;
  W:Integer;
begin
  try
  if DirectoryExists(DataDir) then SaveDialog1.InitialDir:= DataDir
                        else SaveDialog1.InitialDir:='a:\';
  SaveDialog1.FileName:='';
1:if SaveDialog1.Execute then Begin
    if FileExists(SaveDialog1.FileName) then Begin
    Msg:=Format('%s%s%s',[Msg_BukSave0,SaveDialog1.FileName,Msg_BukSave1]);
    W:=GetMyConfirm3Dlg(Msg);
    End else W:=IdYes;
    case W of
      IdYes:
       begin
         try
    Screen.Cursor:=crHourGlass;
    NewFile:=TStringList.Create;
    if GetDataStrList(NewFile) then
          NewFile.SaveToFile(SaveDialog1.FileName);
      DataDir:=Copy(SaveDialog1.FileName,1,
            Length(ExtractFilePath(SaveDialog1.FileName))-1);
    finally
    NewFile.Free;
    Screen.Cursor:=crDefault;
    end;
       end;
      IdNo:
        begin
          SaveDialog1.InitialDir:=Copy(SaveDialog1.FileName,1,
            Length(ExtractFilePath(SaveDialog1.FileName))-1);
          GOTO 1;
        end;
      IdCanCel: Exit;
     end;
    End; {if SaveDialog1.Execute}
  except
  raise;
  end;
```

```
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit Report;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls,Printers,Main;

type
  TReportForm = class(TForm)
    Panel1: TPanel;
    Memo1: TMemo;
    PrintBtn: TBitBtn;
    HelpBtn: TBitBtn;
    SaveAsBtn: TBitBtn;
    SaveDialog1: TSaveDialog;
    PrinterSetupDialog1: TPrinterSetupDialog;
    ChStatistic: TCheckBox;
    ChTurbulent: TCheckBox;
    CloseBtn: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure PrintBtnClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure CloseBtnClick(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ChStatisticClick(Sender: TObject);
    procedure ChTurbulentClick(Sender: TObject);
    procedure ChStatisticKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ChTurbulentKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure SaveAsBtnClick(Sender: TObject);
  private
    { Private declarations }
    procedure ChangeSizeWin;
    procedure UpdateData;
    procedure PrintAllText(Copy1:Integer; Font1:TFont);
    procedure PrintPagesText(FromPage,ToPage,Copy1:Integer; Font1:TFont);
    procedure PrintSelectText(Copy1:Integer; Font1:TFont);
  public
    { Public declarations }
    procedure ViewReportForm(const Title:String);
  end;

var
  ReportForm: TReportForm;

implementation
{$R *.DFM}
uses UnAMKini, ApllDlgs, UnData, UnMeteo, PrnText;

{*************************************************}
procedure TReportForm.ViewReportForm(const Title:String);
```

```
begin
  try
  Caption:=Title;
  ChStatistic.Checked:=not IsStatistica;
  ChTurbulent.Checked:=not IsTurbulent;
  MainForm.Report.Enabled:=false;
  UpdateData;
  Show;
  BringToFront;
  except
  end;
end;

{*************************************************}
procedure TReportForm.ChangeSizeWin;
begin
  Panel1.Width:=Width-24;
  Panel1.Height:=Height-130;
  ChStatistic.Top:=Height-114;
  ChTurbulent.Top:=Height-94;
  CloseBtn.Top:=Height-70;
  PrintBtn.Top:=Height-70;
  SaveAsBtn.Top:=Height-70;
  HelpBtn.Top:=Height-70;
end;

{*************************************************}
procedure TReportForm.FormCreate(Sender: TObject);
begin
  Top:=0;
  Height:=Screen.Height-32;
  Left:=round(0.33*Screen.Width);
  Width:=Screen.Width-Left;
  ChangeSizeWin;
end;

{*************************************************}
procedure TReportForm.FormResize(Sender: TObject);
begin
  ChangeSizeWin;
end;

{*************************************************}
procedure TReportForm.UpdateData;
var
  Report:TStringList;
begin
  try
  Memo1.Clear;
  if Met_Data=nil then Exit;
  if Length(Met_Data)<3 then Exit;
  Screen.Cursor:=crHourGlass;
  Report:=TStringList.Create;
  if MeteoData_CalcStatistic then
  if MeteoData_GetReportStrList(Report) then
  Memo1.Lines.Assign(Report);
  finally
  Report.Free;
  Screen.Cursor:=crDefault;
  end;
```

```
end;

{**************************************************}
procedure TReportForm.ChStatisticClick(Sender: TObject);
begin
  try
  IsStatistica:=not ChStatistic.Checked;
  UpdateData;
  except
  end;
end;

{**************************************************}
procedure TReportForm.ChTurbulentClick(Sender: TObject);
begin
  try
  IsTurbulent:=not ChTurbulent.Checked;
  UpdateData;
  except
  end;
end;

{*****************************************************}
procedure TReportForm.PrintAllText(Copy1:Integer; Font1:TFont);
const
  Sleft='         ';
var
  PrintText:System.Text;
  k,Line:Integer;
begin
  try
  AssignPrn(PrintText);
  Rewrite(PrintText);
  Printer.Canvas.Font:=Font1;
  for k:=1 to Copy1 do begin
    for Line:=0 to Memo1.Lines.Count-1 do begin
      Write(PrintText,Sleft);
      Writeln(PrintText,Memo1.Lines[Line]);
      end;
    if k < Copy1 then Printer.NewPage;
    end;
  finally
  System.Close(PrintText);
  end;
end;

{*****************************************************}
procedure TReportForm.PrintPagesText(FromPage,ToPage,Copy1:Integer; Font1:TFont);
const
  Sleft='         ';
var
  i,k,FromLine,ToLine,Line,LinesInPage:Integer;
  PrintText:System.Text;
  S:ShortString;
  Extent:TSize;
begin
  try
  AssignPrn(PrintText);
  Rewrite(PrintText);
  Printer.Canvas.Font:=Font1;
```

```
    with Printer do begin
      Canvas.Font:=Font1;
      S[0]:=Chr(224);
      for i:=32 to 255 do
        S[i-31]:=Chr(i);
      GetTextExtentPoint(Handle,@S[1],Length(S),Extent);
      LinesInPage:=PageHeight div (Extent.Cy+2);
      if PageHeight mod Extent.Cy <>0 then Dec(LinesInPage);
      FromLine:=(FromPage-1)*LinesInPage;
      if FromLine < 0 then FromLine:=0;
      if FromLine > Memo1.Lines.Count-1 then FromLine:=Memo1.Lines.Count-1;
      ToLine:=ToPage*LinesInPage;
      if ToLine < FromLine then ToLine:=FromLine;
      if ToLine > Memo1.Lines.Count-1 then ToLine:=Memo1.Lines.Count-1;
      for k:=1 to Copy1 do begin
        for Line:=FromLine to ToLine do begin
          Write(PrintText,Sleft);
          Writeln(PrintText,Memo1.Lines[Line]);
          end;
         if k < Copy1 then Printer.NewPage;
         end;
       end;
      finally
      System.Close(PrintText);
      end;
  end;

  {*****************************************************}
  procedure TReportForm.PrintSelectText(Copy1:Integer; Font1:TFont);
  const
    Sleft='          ';
  var
    PrintText:System.Text;
    k,Line:Integer;
    D1:TStringList;
  begin
    if (Memo1.Lines.Count=0) or (Memo1.SelLength=0) then begin
      ShowMessage(Msg_NoSelectText);
      Exit;
      end;
    try
    AssignPrn(PrintText);
    Rewrite(PrintText);
    Printer.Canvas.Font:=Font1;
    D1:=TStringList.Create;
    D1.Assign(Memo1);
    Memo1.CopyToClipboard;
    Memo1.Clear;
    Memo1.PasteFromClipboard;
    for k:=1 to Copy1 do begin
      for Line:=0 to Memo1.Lines.Count-1 do begin
        Write(PrintText,Sleft);
        Writeln(PrintText,Memo1.Lines[Line]);
        end;
       if k < Copy1 then Printer.NewPage;
       end;
    Memo1.Lines.Assign(D1);
    D1.Free;
    System.Close(PrintText);
    except
```

```
  raise
  end;
end;

{*****************************************************}
procedure TReportForm.PrintBtnClick(Sender: TObject);
const
  Sleft='          ';
var
  PrnPages,FromPage,ToPage,Copy:Integer;
  Font1:TFont;
begin
  if Memo1.Lines.Count<=0 then begin
    ShowMessage(Msg_NoDataPrint);
    Exit;
    end;
  Font1:=Memo1.Font;
  if GetPrintTextDlg(PrnPages,FromPage,ToPage,Copy,Font1) Then Begin
   try
   case PrnPages of
    0:PrintAllText(Copy,Font1);
    1:PrintPagesText(FromPage,ToPage,Copy,Font1);
    2:PrintSelectText(Copy,Font1);
   end;
  except
  raise
  end
  end;
end;

{***********************************************}
procedure TReportForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action:=caHide;
  if Met_Data<>nil then
  if Length(Met_Data)>=3 then MainForm.Report.Enabled:=true;
end;

{***********************************************}
procedure TReportForm.CloseBtnClick(Sender: TObject);
begin
  Close;
end;

{***********************************************}
procedure TReportForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then CloseBtnClick(Sender);
end;

{***********************************************}
procedure TReportForm.ChStatisticKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  ChStatisticClick(Sender);
end;

{***********************************************}
procedure TReportForm.ChTurbulentKeyDown(Sender: TObject; var Key: Word;
```

```
  Shift: TShiftState);
begin
  ChTurbulentClick(Sender);
end;

{*************************************************}
procedure TReportForm.SaveAsBtnClick(Sender: TObject);
label 1;
var
  Report:TStringList;
  Msg,ext1:String;
  W:Integer;
begin
if Met_Data=nil then Exit;
if Length(Met_Data)<3 then Exit;
  With SaveDialog1 do Begin
  Title:=Capt_SaveAsReport;
  InitialDir:=DataDir;
  FileName:=GetNameFileDef(Met_Data[NumPosGlobData].TDr);
  FileName:=Format('%s%s%s%s',[DataDir,'\',FileName,'R']);
  FilterIndex:=0;
  Ext1:=ExtractFileExt(FileName);
1:if Execute then Begin
  FileName:=ChangeFileExt(FileName,Ext1);
  if FileExists(FileName) then Begin
  Msg:=Format('%s%s%s',[Msg_BukSave0,FileName,Msg_BukSave1]);
  W:=GetMyConfirm3Dlg(Msg);
  End else W:=IdYes;
  case W of
     IdYes:
      begin
        try
        Screen.Cursor:=crHourGlass;
        Report:=TStringList.Create;
        if MeteoData_CalcStatistic then
        if MeteoData_GetReportStrList(Report) then begin
        Report.SaveToFile(FileName);
        DataDir:=Copy(FileName,1,Length(ExtractFilePath(FileName))-1);
        ReportForm.ViewReportForm(Capt_Viewer+': '+FileName);
        end;
        finally
        Report.Free;
        Screen.Cursor:=crDefault;
        end;
      end;
     IdNo:    GOTO 1;
     IdCancel: Exit;
    end;
  End;
End;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit UnMinFrm;
interface
uses Windows, SysUtils, Classes, Forms, StdCtrls, Controls, ExtCtrls, Main;
```

```
type
  TMinForm = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Label1b: TLabel;
    Edit1: TEdit;
    Panel3: TPanel;
    Label2b: TLabel;
    Label2c: TLabel;
    Edit2: TEdit;
    Panel4: TPanel;
    Label3b: TLabel;
    Label3c: TLabel;
    Edit3: TEdit;
    Panel5: TPanel;
    Label5b: TLabel;
    Edit5: TEdit;
    Panel6: TPanel;
    Label6b: TLabel;
    Edit6: TEdit;
    EditTime: TEdit;
    EditDevice: TEdit;
    LabelTime: TLabel;
    Lab_UGI1: TLabel;
    Lab_UGI2: TLabel;
    Panel7: TPanel;
    Label4b: TLabel;
    Edit4: TEdit;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    procedure UpdateColorEditFont;
    procedure UpdatePanel;
  public
    { Public declarations }
    procedure UpdateEdit;
    procedure UpdateStatusWork;
  end;

var
  MinForm: TMinForm;

implementation
{$R *.dfm}

{****************************************}
procedure TMinForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  try
  if not MainForm.Visible then MainForm.Show;
  MainForm.BringToFront;
  Action:=caHide;
  except
  raise
  end;
end;
```

```
{*********************************************}
procedure TMinForm.FormCreate(Sender: TObject);
begin
  Top:=0;
  Left:=Screen.Width-Width-1;
  Caption:=MainForm.MinFormCaption;
end;

{***************************************}
procedure TMinForm.FormShow(Sender: TObject);
begin
  UpdateColorEditFont;
  UpdatePanel;
  UpdateEdit;
  UpdateStatusWork;
end;

{*********************************}
procedure TMinForm.UpdateColorEditFont;
begin
  Edit1.Color:=MainForm.Edit1.Color;
  Edit2.Color:=MainForm.Edit2.Color;
  Edit3.Color:=MainForm.Edit3.Color;
  Edit4.Color:=MainForm.Edit4.Color;
  Edit5.Color:=MainForm.Edit5.Color;
  Edit6.Color:=MainForm.Edit6.Color;
  Edit1.Font.Color:=MainForm.Edit1.Font.Color;
  Edit2.Font.Color:=MainForm.Edit2.Font.Color;
  Edit3.Font.Color:=MainForm.Edit3.Font.Color;
  Edit4.Font.Color:=MainForm.Edit4.Font.Color;
  Edit5.Font.Color:=MainForm.Edit5.Font.Color;
  Edit6.Font.Color:=MainForm.Edit6.Font.Color;
end;

{*********************************}
procedure TMinForm.UpdatePanel;
begin
  label1b.Caption:=MainForm.label1b.Caption;
  label2b.Caption:=MainForm.label2b.Caption;
  label2c.Visible:=MainForm.label2c.Visible;
  label3b.Caption:=MainForm.label3b.Caption;
  label3c.Visible:=MainForm.label3c.Visible;
  label4b.Caption:=MainForm.label4b.Caption;
  if MainForm.RegimVabor.ItemIndex<>2 then label5b.Left:=2
                              else label5b.Left:=26;
  label5b.Caption:=MainForm.label5b.Caption;
  label6b.Caption:=MainForm.label6b.Caption;
  LabelTime.Caption:=MainForm.LabelTime.Caption
end;

{****************************}
procedure TMinForm.UpdateEdit;
begin
  Edit1.Text:=MainForm.Edit1.Text;
  Edit2.Text:=MainForm.Edit2.Text;
  Edit3.Text:=MainForm.Edit3.Text;
  Edit4.Text:=MainForm.Edit4.Text;
  Edit5.Text:=MainForm.Edit5.Text;
  Edit6.Text:=MainForm.Edit6.Text;
end;
```

```
{***********************************}
procedure TMinForm.UpdateStatusWork;
begin
  Lab_UGI1.Caption:=MainForm.Lab_UGI1.Caption;
  Lab_UGI2.Caption:=MainForm.Lab_UGI2.Caption;
  EditTime.Text:=MainForm.EditTime.Text;
  EditDevice.Color:=MainForm.EditDevice.Color;
  EditDevice.Text:=MainForm.EditDevice.Text;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit PPU25;
interface
uses Windows, SysUtils, Graphics, Forms, Classes, Controls, StdCtrls,
  Buttons, ExtCtrls, Grids, Main, Spin;

type
  TPPU25Dlg = class(TForm)
    Bevel2: TBevel;
    Panel1: TPanel;
    DataGrid: TStringGrid;
    HelpBtn: TBitBtn;
    CloseBtn: TBitBtn;
    WriteBtn: TBitBtn;
    ReadBtn: TBitBtn;
    NewBtn: TBitBtn;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox2: TGroupBox;
    Label7: TLabel;
    Panel2: TPanel;
    Label1b: TLabel;
    EditV: TEdit;
    Panel3: TPanel;
    Label2b: TLabel;
    Label2c: TLabel;
    EditD: TEdit;
    Panel4: TPanel;
    Label3b: TLabel;
    Label3c: TLabel;
    EditT: TEdit;
    Panel5: TPanel;
    Label5b: TLabel;
    EditP: TEdit;
    Panel6: TPanel;
    Label6b: TLabel;
    EditHu: TEdit;
    NewDataBtn: TBitBtn;
    EditTime: TSpinEdit;
    procedure FormCreate(Sender: TObject);
```

```
    procedure NewBtnClick(Sender: TObject);
    procedure ReadBtnClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure WriteBtnClick(Sender: TObject);
    procedure DataGridKeyPress(Sender: TObject; var Key: Char);
    procedure NewDataBtnClick(Sender: TObject);
    procedure EditTimeKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
    function  GetValueConstProgram(Num1:Byte):Single;
    function  GetStrConst(Num1:Byte; par:Single):String;
    procedure SetGridConstantProgram(Row1:Byte);
    function  CheckOneCell4(Num1:Byte):Boolean;
    function  CheckAllCell4:Boolean;
  public
    { Public declarations }
  end;

var
  PPU25Dlg: TPPU25Dlg;

procedure GetPPU25Dlg;

implementation
{$R *.dfm}
uses UnAMKsys,UnAMKini, UnAMKport, ApllDlgs;

const
 CaptPPU_Cells10='  Константа';
 CaptPPU_Cells20='  В пульте';
 CaptPPU_Cells30=' В программе';
 CaptPPU_Cells40='   Новые';
 MsgPPU_New='Вы хотите установить в разделе "Новые" значения градуировочных параметров устройства как в программе?';
 MsgPPU_Read='Чтение значений градуировочных констант из пульта выполнено ';
 MsgPPU_Write='Запись значений градуировочных констант в пульт выполнена ';
 MsgPPU_Uspeh='успешно.';
 MsgPPU_Sboj='с ошибками. Не хотите повторить эту операцию ?';
 MsgPPU_NoNewData='Отсутствуют значения параметров в разделе "Новые" (или введены с ошибками). ';
 MsgPPU_NoWrite='Запись ни одного из градуировочных параметров невозможна.';

{*************************************************}
procedure GetPPU25Dlg;
begin
  try
  Screen.Cursor:=crHourGlass;
  Application.CreateForm(TPPU25Dlg, PPU25Dlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  PPU25Dlg.ShowModal;
  finally
  PPU25Dlg.Free;
  end;
end;

{*****************************************}
function TPPU25Dlg.GetValueConstProgram(Num1:Byte):Single;
var
```

```
  par:Single;
begin
  try
  case Num1 of
    {ConsBeg}
    1:par:=0;
    2:par:=UGI_Fkg;
    3:par:=RastMM[0]/1000;
    4:par:=RastMM[1]/1000;
    5:par:=RastMM[2]/1000;
    6:par:=RastMM[3]/1000;
    7:par:=Tcalibr;
    8:par:=UGI_Dela[0];
    9:par:=UGI_Dela[1];
   10:par:=UGI_Dela[2];
   11:par:=UGI_Dela[3];
   12:par:=UGI_Tcoef*1.e-5;
   13:par:=UGI_Amin;
   14:par:=UGI_Amax;
   15:par:=r_r0;
   16:par:=r_X0;
   17:par:=r_ln;
   18:par:=r_kv*1.e-7;
   19:par:=r_delta;
   20:par:=P_P0;
   21:par:=P_x0;
   22:par:=P_ln;
   23:par:=P_delta;
   24:par:=G_X0;
   25:par:=G_Y0;
   26:par:=G_Xd;
   27:par:=G_Yd;
   28:par:=G_hor;
   29:par:=G_Xdelta;
   30:par:=G_Ydelta;
   31:par:=UGI_Coefz;
   32:par:=UGI_Fiz;
   33:par:=UGI_Az0;
   34:par:=UGI_Az1;
   35:par:=UGI_Az2;
   36:par:=UGI_Az3;
   37:par:=UGI_Az4;
   38:par:=SizeDirWind;
   39:par:=SizeDavlen;
   40:par:=P_T0;
   41:par:=PT_ln;
   42:par:=PT_kv*1.e-3;
   43:par:=PT_kb*1.e-5;
   44:par:=0;//r_kb*1.e-7; //не используется
   45:par:=IsDCB_15;
   46:par:=Tr_T0;
   47:par:=Tr_x0;
   48:par:=Tr_ln;
   49:par:=Tr_kv*1.e-7;
   50:par:=Tr_delta;
   51:par:=NumTimePPU25;
   52:par:=SetMinVolt;
   {ConsEnd}
   else par:=0;
  end;
```

```
  Result:=par;
  except
  raise
  end;
end;

{*****************************************}
function TPPU25Dlg.GetStrConst(Num1:Byte; par:Single):String;
var
  st:String;
begin
  try
  case Num1 of
    {ConsBeg}
    1:st:=Format('%9.0f',[par]);
    2:st:=Format('%9.0f',[par]);
    3:st:=Format('%10.7f',[par]);
    4:st:=Format('%10.7f',[par]);
    5:st:=Format('%10.7f',[par]);
    6:st:=Format('%10.7f',[par]);
    7:st:=Format('%9.2f',[par]);
    8:st:=Format('%9.0f',[par]);
    9:st:=Format('%9.0f',[par]);
   10:st:=Format('%9.0f',[par]);
   11:st:=Format('%9.0f',[par]);
   12:st:=Format('%12.4e',[par]);
   13:st:=Format('%9.0f',[par]);
   14:st:=Format('%9.0f',[par]);
   15:st:=Format('%9.2f',[par]);
   16:st:=Format('%9.2f',[par]);
   17:st:=Format('%14.6e',[par]);
   18:st:=Format('%14.6e',[par]);
   19:st:=Format('%9.2f',[par]);
   20:st:=Format('%9.2f',[par]);
   21:st:=Format('%9.2f',[par]);
   22:st:=Format('%14.6e',[par]);
   23:st:=Format('%9.2f',[par]);
   24:st:=Format('%9.5f',[par]);
   25:st:=Format('%9.5f',[par]);
   26:st:=Format('%9.5f',[par]);
   27:st:=Format('%9.5f',[par]);
   28:st:=Format('%9.2f',[par]);
   29:st:=Format('%9.2f',[par]);
   30:st:=Format('%9.2f',[par]);
   31:st:=Format('%9.7f',[par]);
   32:st:=Format('%9.5f',[par]);
   33:st:=Format('%9.5f',[par]);
   34:st:=Format('%9.5f',[par]);
   35:st:=Format('%9.5f',[par]);
   36:st:=Format('%9.5f',[par]);
   37:st:=Format('%9.5f',[par]);
   38:st:=Format('%9.0f',[par]);
   39:st:=Format('%9.0f',[par]);
   40:st:=Format('%9.2f',[par]);
   41:st:=Format('%14.6e',[par]);
   42:st:=Format('%14.6e',[par]);
   43:st:=Format('%14.6e',[par]);
   44:st:=Format('%14.6e',[par]);
   45:st:=Format('%9.0f',[par]);
   46:st:=Format('%9.2f',[par]);
```

```
    47:st:=Format('%9.2f',[par]);
    48:st:=Format('%14.6e',[par]);
    49:st:=Format('%14.6e',[par]);
    50:st:=Format('%9.4f',[par]);
    51:st:=Format('%9.0f',[par]);
    52:st:=Format('%9.1f',[par]);
    {ConsEnd}
    53:st:=Format('%9.0f',[par]);
  end;
  Result:=st;
  except
  raise
  end;
end;

{****************************************}
procedure TPPU25Dlg.SetGridConstantProgram(Row1:Byte);
var
  Num1:Byte;
  par:Single;
  IS4:array [0..3] of Byte;
  SumKS1:Word;
begin
  SumKs1:=0;
  With DataGrid do Begin
  for Num1:=1 to CountConstant_PPU do begin
    par:=GetValueConstProgram(Num1);
    Cells[Row1,Num1]:=GetStrConst(Num1,par);
    par:=StrToFloat(Cells[Row1,Num1]);
    SumKS1:=SumKS1+IS4[0]+IS4[1]+IS4[2]+IS4[3];
    end;
    Cells[Row1,CountConstant_PPU+1]:=IntToStr(SumKS1);
  End;
end;

{*****************************************}
procedure TPPU25Dlg.FormCreate(Sender: TObject);
var
  i:Integer;
begin
  With DataGrid do Begin
  RowCount:=CountConstant_PPU+2;
  ColWidths[0]:=25;
  ColWidths[1]:=94;
  ColWidths[2]:=94;
  ColWidths[3]:=94;
  ColWidths[4]:=94;
  Cells[0,0]:=' № ';
  Cells[1,0]:=CaptPPU_Cells10;
  Cells[2,0]:=CaptPPU_Cells20;
  Cells[3,0]:=CaptPPU_Cells30;
  Cells[4,0]:=CaptPPU_Cells40;
  Cells[1,1]:=' {begin}';
  Cells[1,2]:='    Fkg';
  for i:=0 to 3 do
    Cells[1,3+i]:='     S'+IntToStr(i+1);
  Cells[1,7]:='  Tcalibr';
  for i:=0 to 3 do
    Cells[1,8+i]:='     D'+IntToStr(i+1);
  Cells[1,12]:='   Tcoef';
```

```
  Cells[1,13]:='   A_min';
  Cells[1,14]:='   A_max';
  Cells[1,15]:='    r_r0';
  Cells[1,16]:='    r_X0';
  Cells[1,17]:='    r_ln';
  Cells[1,18]:='    r_kv';
  Cells[1,19]:='  r_delta';
  Cells[1,20]:='    P_P0';
  Cells[1,21]:='    P_X0';
  Cells[1,22]:='    P_ln';
  Cells[1,23]:='  P_delta';
  Cells[1,24]:='    G_X0';
  Cells[1,25]:='    G_Y0';
  Cells[1,26]:='    G_Xd';
  Cells[1,27]:='    G_Yd';
  Cells[1,28]:='    G_hor';
  Cells[1,29]:='  G_Xdelta';
  Cells[1,30]:='  G_Ydelta';
  Cells[1,31]:='  Coef_Zat';
  Cells[1,32]:='   Fi_Zat';
  Cells[1,33]:='   A0_Zat';
  Cells[1,34]:='   A1_Zat';
  Cells[1,35]:='   A2_Zat';
  Cells[1,36]:='   A3_Zat';
  Cells[1,37]:='   A4_Zat';
  Cells[1,38]:='   SizeDW';
  Cells[1,39]:='   SiseP';
  Cells[1,40]:='   P_T0';
  Cells[1,41]:='   PT_ln';
  Cells[1,42]:='   PT_kv';
  Cells[1,43]:='   PT_kb';
  Cells[1,44]:='   r_kb';
  Cells[1,45]:='   DCB_15';
  Cells[1,46]:='   Tr_T0';
  Cells[1,47]:='   Tr_X0';
  Cells[1,48]:='   Tr_ln';
  Cells[1,49]:='   Tr_kv';
  Cells[1,50]:=' Tr_delta';
  Cells[1,51]:='  NumTime';
  Cells[1,52]:='  SetVolt';
  Cells[1,53]:='   {end}';
  Cells[1,CountConstant_PPU+1]:='   KS ?';
  for i:=1 to CountConstant_PPU do
    Cells[0,i]:=Format('%3d',[i]);
  Cells[0,CountConstant_PPU+1]:=' @';
  End;
  SetGridConstantProgram(3);
  StartAMKport_PPU(COM_BaudRate,true);
end;

{*********************************************************}
procedure TPPU25Dlg.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Close_AMKport(Handle_AMK,IsOpenCOMport);
end;

{*************************************}
procedure TPPU25Dlg.NewBtnClick(Sender: TObject);
var
  i:Integer;
```

```
begin
  With DataGrid do Begin
    for i:=1 to CountConstant_PPU+1 do
      Cells[4,i]:=Cells[3,i];
    End;
end;

{**********************************************}
procedure TPPU25Dlg.ReadBtnClick(Sender: TObject);
label 1;
var
  Num1:Byte;
  par:Single;
  k,ks:Integer;
  IS4:array [0..3] of Byte;
  SumKS1:Word;
begin
  try
  if not IsOpenCOMport then StartAMKport_PPU(COM_BaudRate,true);
  if not IsOpenCOMport then Exit;
1:ks:=0;
  try
  Screen.Cursor:=crHourGlass;
  SumKS1:=0;
  for Num1:=1 to CountConstant_PPU do begin
    k:=ReadPPU_OneConstant(Handle_AMK,Num1,par,OverLap_AMK);
    if (Num1=1) and (k=9) then begin
      GetMyErrorDlg(MsgPPUErr9_0+MsgErr9_1);
      Screen.Cursor:=crDefault;
      Exit;
      end;
    SumKS1:=SumKS1+IS4[0]+IS4[1]+IS4[2]+IS4[3];
    ks:=ks+k;
    DataGrid.Cells[2,Num1]:=GetStrConst(Num1,par)+' *'+IntToStr(k);
    end;
  k:=ReadPPU_KS(Handle_AMK,CountConstant_PPU+1,SumKS1,OverLap_AMK);
  ks:=ks+k;
  DataGrid.Cells[2,CountConstant_PPU+1]:=IntToStr(SumKS1)+' *'+IntToStr(k);
  finally
  Screen.Cursor:=crDefault;
  end;
  if ks=0 then GetMyInformDlg(MsgPPU_Read+MsgPPU_Uspeh)
  else k:=GetMyConfirm2Dlg(MsgPPU_Read+MsgPPU_Sboj);
  if k=IdYes then GOTO 1;
  except
  raise
  end;
end;

{*****************************************}
function TPPU25Dlg.CheckOneCell4(Num1:Byte):Boolean;
var
  i,k:Byte;
  S1:String[1];
begin
  try
  Result:=false;
  k:=0;
  for i:=1 to Length(DataGrid.Cells[4,Num1]) do begin
    S1:=Copy(DataGrid.Cells[4,Num1],i,1);
```

```
      if (S1<>' ') and (S1<>'.') and (S1<>',') and (S1<>'-') then Inc(k);
      end;
    if k>=1 then begin
      StrToFloat(DataGrid.Cells[4,Num1]);
      Result:=true;
      end;
    except
    Result:=false;
    end;
  end;

  {*****************************************}
  function TPPU25Dlg.CheckAllCell4:Boolean;
  var
    Num1:Byte;
  begin
    try
    Result:=true;
    for Num1:=1 to CountConstant_PPU do
      if not CheckOneCell4(Num1) then Result:=false;
    if not Result then GetMyErrorDlg(MsgPPU_NoNewData+MsgPPU_NoWrite);
    except
    Result:=false;
    GetMyErrorDlg(MsgPPU_NoNewData+MsgPPU_NoWrite);
    end;
  end;

  {*****************************************}
  procedure TPPU25Dlg.WriteBtnClick(Sender: TObject);
  label 1;
  var
    Num1:Byte;
    par:Single;
    k,ks:Integer;
    Flag1:Boolean;
    SumKS1:Word;
    IS4:array [0..3] of Byte;
  begin
    k:=GetMyConfirm2Dlg(Msg_Warning+Msg_ChangeGraduir);
    if k <> IdYes then Exit;
    if not CheckAllCell4 then Exit;
    Flag1:=(ParolCurrent=Par_System) or (ParolCurrent=Par_Standart);
    if not Flag1 then Flag1:=GetAdminParol(Par_System,Par_Standart,ParolCurrent);
    if not Flag1 then Exit;
    try
    if not IsOpenCOMport then StartAMKport_PPU(COM_BaudRate,true);
    if not IsOpenCOMport then Exit;
  1:ks:=0;
    try
    Screen.Cursor:=crHourGlass;
    SumKS1:=0;
    for Num1:=1 to CountConstant_PPU do begin
      par:=StrToFloat(DataGrid.Cells[4,Num1]);
      k:=WritePPU_OneConstant(Handle_AMK,Num1,par,OverLap_AMK);
      if (Num1=1) and (k=9) then begin
        GetMyErrorDlg(MsgPPUErr9_0+MsgErr9_1);
        Screen.Cursor:=crDefault;
        Exit;
        end;
      SumKS1:=SumKS1+IS4[0]+IS4[1]+IS4[2]+IS4[3];
```

```
    DataGrid.Cells[2,Num1]:=GetStrConst(Num1,par)+' *'+IntToStr(k);
    ks:=ks+k;
    end;
  finally
  DataGrid.Cells[4,CountConstant_PPU+1]:=IntToStr(SumKS1);
  k:=WritePPU_KS(Handle_AMK,CountConstant_PPU+1,SumKS1,OverLap_AMK);
  DataGrid.Cells[2,CountConstant_PPU+1]:=IntToStr(SumKS1)+' *'+IntToStr(k);
  ks:=ks+k;
  Screen.Cursor:=crDefault;
  end;
  if ks=0 then GetMyInformDlg(MsgPPU_Write+MsgPPU_Uspeh)
  else k:=GetMyConfirm2Dlg(MsgPPU_Write+MsgPPU_Sboj);
  if k=IdYes then GOTO 1;
  except
  raise
  end;
end;

//***********************************
procedure TPPU25Dlg.NewDataBtnClick(Sender: TObject);
var
  T1,V1,D1,P1,r1:Single;
  k,Tim1:Byte;
  eps1:Integer;
begin
  try
  if not IsOpenCOMport then StartAMKport_PPU(COM_BaudRate,true);
  if not IsOpenCOMport then Exit;
  Tim1:=EditTime.Value;
  k:=GetPPU_MeteoParam(Handle_AMK,1,Tim1,P1,OverLap_AMK);
  if k=9 then begin
    EditP.Text:='Err=9';
    EditT.Text:='Err=9';
    EditHu.Text:='Err=9';
    EditD.Text:='Err=9';
    EditV.Text:='Err=9';
    GetMyErrorDlg(MsgPPUErr9_0+MsgErr9_1);
    Exit;
    end;
  if k=0 then begin
    eps1:=round(P1);
    case eps1 of
    -997:GetMyErrorDlg(MsgPPU_997);
    -998:GetMyErrorDlg(MsgPPU_998);
    -999:GetMyErrorDlg(MsgPPU_999);
    end;
    EditP.Text:=Format('%6.1f',[P1]);
    end else EditP.Text:='Err='+IntToStr(k);
  k:=GetPPU_MeteoParam(Handle_AMK,2,Tim1,T1,OverLap_AMK);
  if k=0 then EditT.Text:=Format('%6.2f',[T1])
        else EditT.Text:='Err='+IntToStr(k);
  k:=GetPPU_MeteoParam(Handle_AMK,3,Tim1,r1,OverLap_AMK);
  if k=0 then EditHu.Text:=Format('%6.1f',[r1])
        else EditHu.Text:='Err='+IntToStr(k);
  k:=GetPPU_MeteoParam(Handle_AMK,4,Tim1,D1,OverLap_AMK);
  if k=0 then EditD.Text:=Format('%6.1f',[D1])
        else EditD.Text:='Err='+IntToStr(k);
  k:=GetPPU_MeteoParam(Handle_AMK,5,Tim1,V1,OverLap_AMK);
  if k=0 then EditV.Text:=Format('%6.2f',[V1])
        else EditV.Text:='Err='+IntToStr(k);
```

```
  except
  raise
  end;
end;

{*****************************************************}
procedure TPPU25Dlg.DataGridKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.','-'])=false then Key:=Chr(0);
end;

{*****************************************************}
procedure TPPU25Dlg.EditTimeKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit Options;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, ComCtrls, Spin, Dialogs, Main;

type
  TOptionsDlg = class(TForm)
    Pages1: TPageControl;
    TabOptions: TTabSheet;
    TabValues: TTabSheet;
    TabCOM: TTabSheet;
    TabUGI: TTabSheet;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Bevel3: TBevel;
    Bevel4: TBevel;
    ParamPortBox: TGroupBox;
    EditP: TEdit;
    LabelP: TLabel;
    EditHu: TEdit;
    LabelHu: TLabel;
    EditGx: TEdit;
    LabelGx: TLabel;
    EditGy: TEdit;
    LabelGy: TLabel;
    LabelZm: TLabel;
    SystemBox: TGroupBox;
    LabelFkg: TLabel;
    EditFkg: TEdit;
    SonicBox: TGroupBox;
    LabelRast1: TLabel;
    LabelRast2: TLabel;
    LabelRast3: TLabel;
    LabelRast4: TLabel;
    EditRast1: TEdit;
```

```
EditRast2: TEdit;
EditRast3: TEdit;
EditRast4: TEdit;
EditTCalibr: TEdit;
LabelTCalibr: TLabel;
TabFiltr: TTabSheet;
Bevel5: TBevel;
BoundBox: TGroupBox;
LabelAmax: TLabel;
LabelAmin: TLabel;
EditAmax: TEdit;
EditAmin: TEdit;
Filtr1Box: TGroupBox;
SpinValFiltr1: TSpinEdit;
EditIntWinFiltr1: TEdit;
CheckFiltr1: TCheckBox;
LabelValFiltr1: TLabel;
LabelIntWinFiltr1: TLabel;
Filtr2Box: TGroupBox;
CheckFiltr2: TCheckBox;
DelayBox: TGroupBox;
LabelD1: TLabel;
LabelD2: TLabel;
LabelD3: TLabel;
LabelD4: TLabel;
EditD1: TEdit;
EditD2: TEdit;
EditD3: TEdit;
EditD4: TEdit;
EditTcoef: TEdit;
LabelTcoef: TLabel;
DatchBox: TGroupBox;
CheckP: TCheckBox;
CheckHu: TCheckBox;
CenzuirBox: TGroupBox;
LabelCenzVolum: TLabel;
LabelBound: TLabel;
SpinVolumEdit: TSpinEdit;
CheckCenzuir: TCheckBox;
SpinBoundEdit: TSpinEdit;
GroupBox1: TGroupBox;
ChZatenen: TCheckBox;
EditBlockData: TSpinEdit;
LabBlockData: TLabel;
EditTimer1: TEdit;
LabTimer1: TLabel;
EditDataDir: TEdit;
LabelDataDir: TLabel;
CheckHu100: TCheckBox;
TabSensor: TTabSheet;
Bevel6: TBevel;
DavlBox: TGroupBox;
LabelP_P0: TLabel;
LabelP_x0: TLabel;
LabPT_ln: TLabel;
LabPT_kv: TLabel;
LabP_ln: TLabel;
LabPT_kb: TLabel;
EditP_P0: TEdit;
EditP_x0: TEdit;
```

```
EditPT_ln: TEdit;
EditPT_kv: TEdit;
EditP_ln: TEdit;
EditPT_kb: TEdit;
VlagBox: TGroupBox;
Labr_r0: TLabel;
Labr_X0: TLabel;
Labr_ln: TLabel;
Labr_kv: TLabel;
Editr_r0: TEdit;
Editr_X0: TEdit;
Editr_ln: TEdit;
Editr_kv: TEdit;
TiltBox: TGroupBox;
LabelG_x0: TLabel;
LabelG_y0: TLabel;
LabelG_xd: TLabel;
LabelG_yd: TLabel;
LabelG_hor: TLabel;
EditG_x0: TEdit;
EditG_y0: TEdit;
EditG_xd: TEdit;
EditG_yd: TEdit;
EditG_hor: TEdit;
KeyBtn: TSpeedButton;
EditP_T0: TEdit;
LabP_T0: TLabel;
EditTr_T0: TEdit;
LabTr_T0: TLabel;
EditTr_X0: TEdit;
LabTr_X0: TLabel;
EditTr_ln: TEdit;
LabTr_ln: TLabel;
CheckRonT: TCheckBox;
SizeDavlenRG: TRadioGroup;
SizeDirWindRG: TRadioGroup;
BaudRateBox: TComboBox;
SizeRxBufBox: TComboBox;
SizeTxBufBox: TComboBox;
LabBaudRate: TLabel;
LabSizeTx1: TLabel;
LabSizeTx2: TLabel;
LabSizeRx1: TLabel;
LabSizeRx2: TLabel;
EditZm: TEdit;
Label2: TLabel;
EditSetVolt: TEdit;
LabelSetVolt: TLabel;
EditTr_kv: TEdit;
LabTr_kv: TLabel;
EditUGI_hor: TEdit;
LabelUGI_hor: TLabel;
LabUgihor2: TLabel;
Label9: TLabel;
PortBox: TGroupBox;
PortCom: TComboBox;
GroupBox2: TGroupBox;
CoefFreqIzmBox: TComboBox;
LabCoefFreqIzm: TLabel;
Label3: TLabel;
```

```
    DelayDataBox: TComboBox;
    Label1: TLabel;
    EditDeviceOrient: TEdit;
    LabelFi: TLabel;
    Label10: TLabel;
    LabelFi1: TLabel;
    EditKoefZat: TEdit;
    LabKoefZat: TLabel;
    ConnectRG: TRadioGroup;
    TelBox: TGroupBox;
    EditNumberTeleFon: TEdit;
    SaveFileBox: TGroupBox;
    ChReportFile: TCheckBox;
    ChDataFile: TCheckBox;
    ChBinFile: TCheckBox;
    ChInfoFile: TCheckBox;
    ChArrStrFile: TCheckBox;
    ChTehnolDDE: TCheckBox;
    PodstBox: TComboBox;
    LabelPodst: TLabel;
    EditNumServer: TEdit;
    LabelNumServer: TLabel;
    procedure SetData(Sender: TObject);
    procedure EditPKeyPress(Sender: TObject; var Key: Char);
    procedure EditHuKeyPress(Sender: TObject; var Key: Char);
    procedure EditDeviceOrientKeyPress(Sender: TObject; var Key: Char);
    procedure EditGxKeyPress(Sender: TObject; var Key: Char);
    procedure EditGyKeyPress(Sender: TObject; var Key: Char);
    procedure EditZmKeyPress(Sender: TObject; var Key: Char);
    procedure EditFkgKeyPress(Sender: TObject; var Key: Char);
    procedure EditAminKeyPress(Sender: TObject; var Key: Char);
    procedure EditAmaxKeyPress(Sender: TObject; var Key: Char);
    procedure EditTCoefKeyPress(Sender: TObject; var Key: Char);
    procedure EditZatenCoefKeyPress(Sender: TObject; var Key: Char);
    procedure EditPKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditHuKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditHuClick(Sender: TObject);
    procedure EditGxKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditGxClick(Sender: TObject);
    procedure EditGyKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditGyClick(Sender: TObject);
    procedure EditZmKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditZmClick(Sender: TObject);
    procedure EditFkgKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditFkgClick(Sender: TObject);
    procedure EditAminKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditAminClick(Sender: TObject);
    procedure EditAmaxKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditAmaxClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SpinValFiltr1Click(Sender: TObject);
    procedure SpinValFiltr1KeyDown(Sender: TObject; var Key: Word;
```

```
  Shift: TShiftState);
procedure SpinValFiltr1KeyPress(Sender: TObject; var Key: Char);
procedure EditIntWinFiltr1Click(Sender: TObject);
procedure EditIntWinFiltr1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditIntWinFiltr1KeyPress(Sender: TObject; var Key: Char);
procedure EditRast1KeyPress(Sender: TObject; var Key: Char);
procedure EditRast2KeyPress(Sender: TObject; var Key: Char);
procedure EditRast3KeyPress(Sender: TObject; var Key: Char);
procedure EditRast4KeyPress(Sender: TObject; var Key: Char);
procedure EditTCalibrKeyPress(Sender: TObject; var Key: Char);
procedure EditSpeedCalibrKeyPress(Sender: TObject; var Key: Char);
procedure EditD1KeyPress(Sender: TObject; var Key: Char);
procedure EditD2KeyPress(Sender: TObject; var Key: Char);
procedure EditD3KeyPress(Sender: TObject; var Key: Char);
procedure EditD4KeyPress(Sender: TObject; var Key: Char);
procedure EditRast1Click(Sender: TObject);
procedure EditRast1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditRast2Click(Sender: TObject);
procedure EditRast2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditRast3Click(Sender: TObject);
procedure EditRast3KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditRast4Click(Sender: TObject);
procedure EditRast4KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditTCalibrClick(Sender: TObject);
procedure EditTCalibrKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditD1Click(Sender: TObject);
procedure EditD1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditD2Click(Sender: TObject);
procedure EditD2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditD3Click(Sender: TObject);
procedure EditD3KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditD4Click(Sender: TObject);
procedure EditD4KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure SpinVolumEditClick(Sender: TObject);
procedure SpinVolumEditKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure SpinVolumEditKeyPress(Sender: TObject; var Key: Char);
procedure SpinBoundEditClick(Sender: TObject);
procedure SpinBoundEditKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure SpinBoundEditKeyPress(Sender: TObject; var Key: Char);
procedure CheckPClick(Sender: TObject);
procedure CheckPKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure CheckHuClick(Sender: TObject);
procedure CheckHuKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditTimer1KeyPress(Sender: TObject; var Key: Char);
procedure EditTimer1Click(Sender: TObject);
procedure EditTimer1KeyDown(Sender: TObject; var Key: Word;
```

```
  Shift: TShiftState);
procedure EditPClick(Sender: TObject);
procedure EditDataDirClick(Sender: TObject);
procedure EditDataDirKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditP_P0Click(Sender: TObject);
procedure EditP_P0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditP_P0KeyPress(Sender: TObject; var Key: Char);
procedure EditP_x0Click(Sender: TObject);
procedure EditP_x0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditP_x0KeyPress(Sender: TObject; var Key: Char);
procedure EditP_lnClick(Sender: TObject);
procedure EditP_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditP_lnKeyPress(Sender: TObject; var Key: Char);
procedure EditPT_lnClick(Sender: TObject);
procedure EditPT_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditPT_lnKeyPress(Sender: TObject; var Key: Char);
procedure EditPT_kvClick(Sender: TObject);
procedure EditPT_kvKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditPT_kvKeyPress(Sender: TObject; var Key: Char);
procedure EditPT_kbClick(Sender: TObject);
procedure EditPT_kbKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditPT_kbKeyPress(Sender: TObject; var Key: Char);
procedure Editr_r0Click(Sender: TObject);
procedure Editr_r0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Editr_r0KeyPress(Sender: TObject; var Key: Char);
procedure Editr_X0Click(Sender: TObject);
procedure Editr_X0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Editr_X0KeyPress(Sender: TObject; var Key: Char);
procedure Editr_lnClick(Sender: TObject);
procedure Editr_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Editr_lnKeyPress(Sender: TObject; var Key: Char);
procedure Editr_kvClick(Sender: TObject);
procedure Editr_kvKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Editr_kvKeyPress(Sender: TObject; var Key: Char);
procedure EditG_x0Click(Sender: TObject);
procedure EditG_x0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditG_x0KeyPress(Sender: TObject; var Key: Char);
procedure EditG_y0Click(Sender: TObject);
procedure EditG_y0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditG_y0KeyPress(Sender: TObject; var Key: Char);
procedure EditG_horClick(Sender: TObject);
procedure EditG_horKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EditG_horKeyPress(Sender: TObject; var Key: Char);
procedure EditG_xdClick(Sender: TObject);
procedure EditG_xdKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
```

```
    procedure EditG_xdKeyPress(Sender: TObject; var Key: Char);
    procedure EditG_ydClick(Sender: TObject);
    procedure EditG_ydKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditG_ydKeyPress(Sender: TObject; var Key: Char);
    procedure Pages1Change(Sender: TObject);
    procedure KeyBtnClick(Sender: TObject);
    procedure EditP_T0KeyPress(Sender: TObject; var Key: Char);
    procedure EditP_T0Click(Sender: TObject);
    procedure EditP_T0KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditTr_T0KeyPress(Sender: TObject; var Key: Char);
    procedure EditTr_X0KeyPress(Sender: TObject; var Key: Char);
    procedure EditTr_lnKeyPress(Sender: TObject; var Key: Char);
    procedure EditTr_kvKeyPress(Sender: TObject; var Key: Char);
    procedure EditTr_T0Click(Sender: TObject);
    procedure EditTr_T0KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditTr_X0Click(Sender: TObject);
    procedure EditTr_X0KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditTr_lnClick(Sender: TObject);
    procedure EditTr_lnKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditTr_kvClick(Sender: TObject);
    procedure EditTr_kvKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditTcoefClick(Sender: TObject);
    procedure EditTcoefKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditSetVoltKeyPress(Sender: TObject; var Key: Char);
    procedure EditSetVoltClick(Sender: TObject);
    procedure EditSetVoltKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditUGI_horKeyPress(Sender: TObject; var Key: Char);
    procedure EditUGI_horClick(Sender: TObject);
    procedure EditUGI_horKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditKoefZatKeyPress(Sender: TObject; var Key: Char);
    procedure EditKoefZatClick(Sender: TObject);
    procedure EditKoefZatKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditDeviceOrientClick(Sender: TObject);
    procedure EditDeviceOrientKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ConnectRGClick(Sender: TObject);
    procedure EditNumberTeleFonKeyPress(Sender: TObject; var Key: Char);
    procedure BaudRateBoxKeyPress(Sender: TObject; var Key: Char);
    procedure BaudRateBoxClick(Sender: TObject);
    procedure BaudRateBoxKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditNumServerKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
    procedure SetupDlg(F1:Boolean);
    procedure SetupParolAdmin;
    procedure SetupParolStandart;
    procedure UpdateTypeConnect;
    procedure GetCoefFreqIzm;
    procedure SetCoefFreqIzm;
```

```
    procedure GetData;
    Function  ValidData:Boolean;
    Function  Execute:Boolean;
  public
    { Public declarations }
  end;

var
  OptionsDlg: TOptionsDlg;

function GetOptionsDlg:WordBool;

implementation
{$R *.DFM}
Uses UnAMKini, UnAMKsys, ApllDlgs, Phone;

//****************************************
function GetOptionsDlg:WordBool;
begin
  try
  Screen.Cursor:=crHourGlass;
  Application.CreateForm(TOptionsDlg, OptionsDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  Result:=OptionsDlg.Execute;
  finally
  OptionsDlg.Free;
  end;
end;

//****************************************
Function TOptionsDlg.Execute:Boolean;
Label 10;
Begin
  Result:=false;
10:if ShowModal=idOk Then
  Begin
     if ValidData=true then Result:=true
                  else goto 10;
     GetData;
  End;
End;

//*****************************************************
procedure TOptionsDlg.Pages1Change(Sender: TObject);
begin
  if Pages1.ActivePageIndex>=4 then begin
    KeyBtn.Visible:=true;
    if (ParolCurrent=Par_System) or (ParolCurrent=Par_Standart) then KeyBtn.Enabled:=false;
    end else KeyBtn.Visible:=false;
end;

//*************************************
procedure TOptionsDlg.SetupDlg(F1:Boolean);
begin
  EditFkg.ReadOnly:=f1;
  EditRast1.ReadOnly:=f1;
  EditRast2.ReadOnly:=f1;
```

```
  EditRast3.ReadOnly:=f1;
  EditRast4.ReadOnly:=f1;
  EditTCalibr.ReadOnly:=f1;
  EditD1.ReadOnly:=f1;
  EditD2.ReadOnly:=f1;
  EditD3.ReadOnly:=f1;
  EditD4.ReadOnly:=f1;
  EditTcoef.ReadOnly:=f1;
  EditKoefZat.ReadOnly:=f1;
  EditUGI_hor.ReadOnly:=f1;
  EditSetVolt.ReadOnly:=f1;
  EditP_P0.ReadOnly:=f1;
  EditP_x0.ReadOnly:=f1;
  EditP_ln.ReadOnly:=f1;
  EditP_T0.ReadOnly:=f1;
  EditPT_ln.ReadOnly:=f1;
  EditPT_kv.ReadOnly:=f1;
  EditPT_kb.ReadOnly:=f1;
  Editr_r0.ReadOnly:=f1;
  Editr_X0.ReadOnly:=f1;
  Editr_ln.ReadOnly:=f1;
  Editr_kv.ReadOnly:=f1;
  EditTr_T0.ReadOnly:=f1;
  EditTr_x0.ReadOnly:=f1;
  EditTr_ln.ReadOnly:=f1;
  EditTr_kv.ReadOnly:=f1;
  CheckRonT.Enabled:=not f1;
  EditGx.ReadOnly:=f1;
  EditGy.ReadOnly:=f1;
  EditG_x0.ReadOnly:=f1;
  EditG_y0.ReadOnly:=f1;
  EditG_xd.ReadOnly:=f1;
  EditG_yd.ReadOnly:=f1;
  EditG_hor.ReadOnly:=f1;
end;

{*********************************}
procedure TOptionsDlg.SetupParolAdmin;
begin
  SetupDlg(false);
end;

{*********************************}
procedure TOptionsDlg.SetupParolStandart;
begin
  SetupDlg(false);
  EditFkg.ReadOnly:=true;
  EditD1.ReadOnly:=true;
  EditD2.ReadOnly:=true;
  EditD3.ReadOnly:=true;
  EditD4.ReadOnly:=true;
  EditGx.ReadOnly:=true;
  EditGy.ReadOnly:=true;
  EditG_x0.ReadOnly:=true;
  EditG_y0.ReadOnly:=true;
  EditG_xd.ReadOnly:=true;
  EditG_yd.ReadOnly:=true;
  EditG_hor.ReadOnly:=true;
end;
```

```
{*********************************}
procedure TOptionsDlg.FormCreate(Sender: TObject);
begin
  Pages1Change(Sender);
  SetData(Sender);
  SetupDlg(true);
  if ParolCurrent=Par_System then SetupParolAdmin;
  if ParolCurrent=Par_Standart then SetupParolStandart;
end;

{****************************************}
procedure TOptionsDlg.KeyBtnClick(Sender: TObject);
  var
  W:Integer;
  Flag1:Boolean;
begin
  W:=GetMyConfirm2Dlg(Msg_Warning+Msg_ChangeGraduir);
  if W <> IdYes then Exit;
  Flag1:=(ParolCurrent=Par_System) or (ParolCurrent=Par_Standart);
  if not Flag1 then Flag1:=GetAdminParol(Par_System,Par_Standart,ParolCurrent);
  if Flag1 then Begin
    if ParolCurrent=Par_System then SetupParolAdmin;
    if ParolCurrent=Par_Standart then SetupParolStandart;
    KeyBtn.Enabled:=false;
    End;
end;

{***********************************}
procedure TOptionsDlg.UpdateTypeConnect;
begin
  if ConnectRG.ItemIndex = 0 then begin
    EditNumberTeleFon.Enabled:=false;
    TelBox.Enabled:=false;
    end else begin
    EditNumberTeleFon.Enabled:=true;
    TelBox.Enabled:=true;
    end;
end;

//**********************************************
procedure TOptionsDlg.ConnectRGClick(Sender: TObject);
begin
  UpdateTypeConnect;
end;

{*******************************}
procedure TOptionsDlg.CheckPClick(Sender: TObject);
begin
  if CheckP.Checked then EditP.Text:=Format('%6.1f',[P_delta])
                 else EditP.Text:=Format('%6.1f',[MMdavlKl]);
end;

{********************************}
procedure TOptionsDlg.CheckPKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  CheckPClick(Sender);
end;

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

```
procedure TOptionsDlg.CheckHuClick(Sender: TObject);
begin
  if CheckHu.Checked then EditHu.Text:=Format('%6.1f',[r_delta])
                 else EditHu.Text:=Format('%6.1f',[OtvlagKl]);
end;

{*******************************}
procedure TOptionsDlg.CheckHuKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  CheckHuClick(Sender);
end;

{*******************************}
procedure TOptionsDlg.GetCoefFreqIzm;
var
  per:Real;
begin
  per:=TimeCikle/CoefFreqIzm;
  case CoefFreqIzmBox.ItemIndex of
  0:CoefFreqIzm:=1;
  1:CoefFreqIzm:=2;
  2:CoefFreqIzm:=4;
  3:CoefFreqIzm:=8;
  4:CoefFreqIzm:=16;
  5:CoefFreqIzm:=32;
  6:CoefFreqIzm:=64;
  end;
  TimeCikle:=per*CoefFreqIzm;
end;

{*******************************}
procedure TOptionsDlg.SetCoefFreqIzm;
begin
  if CoefFreqIzm = 64 then CoefFreqIzmBox.ItemIndex:=6
    else if CoefFreqIzm = 32 then CoefFreqIzmBox.ItemIndex:=5
      else if CoefFreqIzm = 16 then CoefFreqIzmBox.ItemIndex:=4
        else if CoefFreqIzm = 8 then CoefFreqIzmBox.ItemIndex:=3
          else if CoefFreqIzm = 4 then CoefFreqIzmBox.ItemIndex:=2
            else if CoefFreqIzm = 2 then CoefFreqIzmBox.ItemIndex:=1
              else CoefFreqIzmBox.ItemIndex:=0;
end;

{*******************************}
procedure TOptionsDlg.SetData(Sender: TObject);
var
  k,i:byte;
Begin
  try
  {Communication}
  if IsModem then ConnectRG.ItemIndex:=1
          else ConnectRG.ItemIndex:=0;
  EditNumberTeleFon.Text:=NumberTeleFon;
  UpdateTypeConnect;
  PortCOM.ItemIndex:=COM_Number;
  BaudRateBox.Text:=IntToStr(COM_BaudRate);
  SizeRxBufBox.Text:=IntToStr(COM_SizeRxBuf);
  SizeTxBufBox.Text:=IntToStr(COM_SizeTxBuf);
  SetCoefFreqIzm;
  DelayDataBox.ItemIndex:=DelayDataCable-1;
```

```
{Application}
CheckP.Checked:=IsDeviceP;
CheckHu.Checked:=IsDeviceHu;
ChInfoFile.Checked:=IsSaveInfoFile;
ChReportFile.Checked:=IsSaveReportFile;
ChDataFile.Checked:=IsSaveDataFile;
ChBinFile.Checked:=IsSaveBinFile;
ChArrStrFile.Checked:=IsSaveArrStrFile;
ChTehnolDDE.Checked:=IsTehnologyDDE;
CheckHu100.Checked:=IsCutHu_100;
ChZatenen.Checked:=IsCorrectZ;
EditBlockData.Value:=BlockDataCable;
EditTimer1.Text:=Format('%3.1f',[1000/TimeTimerSpeed]);
{Values}
CheckPClick(Sender);
CheckHuClick(Sender);
EditGx.Text:=Format('%6.1f',[G_xdelta]);
EditGy.Text:=Format('%6.1f',[G_ydelta]);
EditDeviceOrient.Text:=Format('%6.1f',[DeviceOrient]);
SizeDirWindRG.ItemIndex:=SizeDirWind;
SizeDavlenRG.ItemIndex:=SizeDavlen;
PodstBox.ItemIndex:=TypeSurface;
EditDataDir.Text:=DataDir;
k:=Length(EditNumServer.Text);
if k>6 then EditNumServer.Text:=Copy(EditNumServer.Text,1,6)
    else if k<6 then begin
    for i:=k+1 to 6 do  EditNumServer.Text:=' '+EditNumServer.Text;
    end;
{Filtres}
EditAmin.Text:=IntToStr(UGI_Amin);
EditAmax.Text:=IntToStr(UGI_Amax);
CheckCenzuir.Checked:=IsCenzuir;
SpinVolumEdit.Value:=VolumeCenzuir;
SpinBoundEdit.Value:=BoundsCenzuir;
SpinValFiltr1.Value:=MaxMasFiltrInput;
EditIntWinFiltr1.Text:=IntToStr(IntWinFiltrInput);
CheckFiltr1.Checked:=IsFiltrInput;
CheckFiltr2.Checked:=IsFiltrOutput;
{Thermoanemometer}
EditFkg.Text:=Format('%6.3f',[UGI_Fkg/1.e6]);
EditRast1.Text:=Format('%8.4f',[RastMM[0]]);
EditRast2.Text:=Format('%8.4f',[RastMM[1]]);
EditRast3.Text:=Format('%8.4f',[RastMM[2]]);
EditRast4.Text:=Format('%8.4f',[RastMM[3]]);
EditTCalibr.Text:=Format('%8.4f',[Tcalibr]);
EditD1.Text:=IntToStr(UGI_Dela[0]);
EditD2.Text:=IntToStr(UGI_Dela[1]);
EditD3.Text:=IntToStr(UGI_Dela[2]);
EditD4.Text:=IntToStr(UGI_Dela[3]);
EditTcoef.Text:=Format('%8.4f',[UGI_Tcoef]);
EditKoefZat.Text:=Format('%6.3f',[UGI_CoefZ]);
EditUGI_hor.Text:=Format('%5.1f',[UGI_hor]);
EditZm.Text:=Format('%6.2f',[UGI_Z]);
EditSetVolt.Text:=Format('%5.1f',[SetMinVolt]);
{Sensores}
EditP_P0.Text:=Format('%7.2f',[P_P0]);
EditP_x0.Text:=Format('%7.1f',[P_x0]);
EditP_ln.Text:=Format('%9.7f',[P_ln]);
EditP_T0.Text:=Format('%6.2f',[P_T0]);
EditPT_ln.Text:=Format('%9.6f',[PT_ln]);
```

```
    EditPT_kv.Text:=Format('%9.6f',[PT_kv]);
    EditPT_kb.Text:=Format('%9.6f',[PT_kb]);
    Editr_r0.Text:=Format('%6.2f',[r_r0]);
    Editr_X0.Text:=Format('%7.1f',[r_X0]);
    Editr_ln.Text:=Format('%9.7f',[r_ln]);
    Editr_kv.Text:=Format('%9.6f',[r_kv]);
    EditTr_T0.Text:=Format('%6.2f',[Tr_T0]);
    EditTr_x0.Text:=Format('%7.1f',[Tr_x0]);
    EditTr_ln.Text:=Format('%9.7f',[Tr_ln]);
    EditTr_kv.Text:=Format('%9.6f',[Tr_kv]);
    CheckRonT.Checked:=isDCB_15=1;
    EditG_x0.Text:=Format('%6.4f',[G_x0]);
    EditG_y0.Text:=Format('%6.4f',[G_y0]);
    EditG_xd.Text:=Format('%6.4f',[G_xd]);
    EditG_yd.Text:=Format('%6.4f',[G_yd]);
    EditG_hor.Text:=Format('%6.2f',[G_hor]);
    except
    raise;
    End;
  End;

  {***********************************}
  procedure TOptionsDlg.GetData;
  Begin
    try
    {Communication}
    if IsPhoneForm then PhoneForm.Close;
    if ConnectRG.ItemIndex=0 then begin
    Mainform.SetupConnect.Enabled:=false;
    Mainform.OpenDevice;
    end else begin
    Mainform.SetupConnect.Enabled:=true;
    Mainform.CloseDevice;
    end;
    Mainform.ViewPhoneItem.Enabled:=Mainform.SetupConnect.Enabled;
    if ConnectRG.ItemIndex<>1 then IsModem:=false
                          else IsModem:=true;
    NumberTeleFon:=EditNumberTeleFon.Text;
    COM_Number:=PortCOM.ItemIndex;
    COM_BaudRate:=StrToInt(BaudRateBox.Text);
    COM_SizeRxBuf:=StrToInt(SizeRxBufBox.Text);
    COM_SizeTxBuf:=StrToInt(SizeTxBufBox.Text);
    GetCoefFreqIzm;
    DelayDataCable:=DelayDataBox.ItemIndex+1;
    {Application}
    IsDeviceP:=CheckP.Checked;
    IsDeviceHu:=CheckHu.Checked;
    IsSaveInfoFile:=ChInfoFile.Checked;
    IsSaveReportFile:=ChReportFile.Checked;
    IsSaveDataFile:=ChDataFile.Checked;
    IsSaveBinFile:=ChBinFile.Checked;
    IsSaveArrStrFile:=ChArrStrFile.Checked;
    IsTehnologyDDE:=ChTehnolDDE.Checked;
    IsCutHu_100:=CheckHu100.Checked;
    IsCorrectZ:=ChZatenen.Checked;
    BlockDataCable:=EditBlockData.Value;
    TimeTimerSpeed:=round(1000/StrToFloat(EditTimer1.Text));
    {Values}
    if CheckP.Checked then P_delta:=StrToFloat(EditP.Text)
                  else MMdavlKl:=StrToFloat(EditP.Text);
```

```
if CheckHu.Checked then begin
                r_delta:=StrToFloat(EditHu.Text);
            end else OtVlagKl:=StrToFloat(EditHu.Text);
G_xdelta:=StrToFloat(EditGx.Text);
G_ydelta:=StrToFloat(EditGy.Text);
DeviceOrient:=StrToFloat(EditDeviceOrient.Text);
TypeSurface:=PodstBox.ItemIndex;
DataDir:=EditDataDir.Text;
SizeDirWind:=SizeDirWindRG.ItemIndex;
SizeDavlen:=SizeDavlenRG.ItemIndex;
{Filtres}
UGI_Amin:=StrToInt(EditAmin.Text);
UGI_Amax:=StrToInt(EditAmax.Text);
IsCenzuir:=CheckCenzuir.Checked;
VolumeCenzuir:=SpinVolumEdit.Value;
BoundsCenzuir:=SpinBoundEdit.Value;
MaxMasFiltrInput:=SpinValFiltr1.Value;
IntWinFiltrInput:=StrToInt(EditIntWinFiltr1.Text);
IsFiltrInput:=CheckFiltr1.Checked;
IsFiltrOutput:=CheckFiltr2.Checked;
{Thermoanemometer}
UGI_Fkg:=round(1.e6*StrToFloat(EditFkg.Text));
RastMM[0]:=StrToFloat(EditRast1.Text);
RastMM[1]:=StrToFloat(EditRast2.Text);
RastMM[2]:=StrToFloat(EditRast3.Text);
RastMM[3]:=StrToFloat(EditRast4.Text);
Tcalibr:=StrToFloat(EditTCalibr.Text);
UGI_Dela[0]:=StrToInt(EditD1.Text);
UGI_Dela[1]:=StrToInt(EditD2.Text);
UGI_Dela[2]:=StrToInt(EditD3.Text);
UGI_Dela[3]:=StrToInt(EditD4.Text);
UGI_TCoef:=StrToFloat(EditTCoef.Text);
UGI_CoefZ:=StrToFloat(EditKoefZat.Text);
UGI_hor:=StrToFloat(EditUGI_hor.Text);
UGI_Z:=StrToFloat(EditZm.Text);
SetMinVolt:=StrToFloat(EditSetVolt.Text);
{Sensores}
P_P0:=StrToFloat(EditP_P0.Text);
P_x0:=StrToFloat(EditP_x0.Text);
P_ln:=StrToFloat(EditP_ln.Text);
P_T0:=StrToFloat(EditP_T0.Text);
PT_ln:=StrToFloat(EditPT_ln.Text);
PT_kv:=StrToFloat(EditPT_kv.Text);
PT_kb:=StrToFloat(EditPT_kb.Text);
r_r0:=StrToFloat(Editr_r0.Text);
r_X0:=StrToFloat(Editr_X0.Text);
r_ln:=StrToFloat(Editr_ln.Text);
r_kv:=StrToFloat(Editr_kv.Text);
Tr_T0:=StrToFloat(EditTr_T0.Text);
Tr_x0:=StrToFloat(EditTr_x0.Text);
Tr_ln:=StrToFloat(EditTr_ln.Text);
Tr_kv:=StrToFloat(EditTr_kv.Text);
if CheckRonT.Checked then isDCB_15:=1
            else isDCB_15:=0;
G_x0:=StrToFloat(EditG_x0.Text);
G_y0:=StrToFloat(EditG_y0.Text);
G_xd:=StrToFloat(EditG_xd.Text);
G_yd:=StrToFloat(EditG_yd.Text);
G_hor:=StrToFloat(EditG_hor.Text);
except
```

```
  raise;
  End;
End;

{*************************************}
Function TOptionsDlg.ValidData:Boolean;
var
  Msg,se1,Svar:String;
  perE:Real;
  perL,perL2:Integer;
Begin
   try
   Result:=false;
   {Communication}
   {BaudRate}
   perL:=StrToInt(BaudRateBox.Text);
   if (perL<110) or (perL>256000) then
    begin
      Msg:=Format('%s%s%s%s%s%s',[MsgErr_sf,"",LabBaudRate.Caption,"": (<110',MsgErr_or,'>256000).']);
      GetMyInvalidDlg(Msg);
      BaudRateBox.Color:=ClRed;
      Exit;
    end;
    {SizeRxBuf}
   perL:=StrToInt(SizeRxBufBox.Text);
   if (perL<110) or (perL>65000) then
    begin
      Msg:=Format('%s%s%s%s%s%s%s%s',[MsgErr_sf,"",LabSizeTx1.Caption,' ',
          LabSizeTx2.Caption,"": (<110',MsgErr_or,'>65000).']);
      GetMyInvalidDlg(Msg);
      SizeRxBufBox.Color:=ClRed;
      Exit;
    end;
   {SizeTxBuf}
   perL:=StrToInt(SizeTxBufBox.Text);
   if (perL<110) or (perL>65000) then
    begin
      Msg:=Format('%s%s%s%s%s%s%s%s',[MsgErr_sf,"",LabSizeRx1.Caption,' ',
          LabSizeRx2.Caption,"": (<110',MsgErr_or,'>65000).']);
      GetMyInvalidDlg(Msg);
      SizeTxBufBox.Color:=ClRed;
      Exit;
    end;

   {application}
   {}
    se1:=Format('%s%s%s',["",LabTimer1.Caption," "]);
    if not ControlEdit(EditTimer1) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabOptions;
      EditTimer1.Color:=ClRed;
      Exit;
   End;
   perE:=StrToFloat(EditTimer1.Text);
   if (perE<0.2) or (perE>10) then
    begin
     Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0.2',MsgErr_or,'>10).']);
     GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabOptions;
      EditTimer1.Color:=ClRed;
```

```
      Exit;
    end;
    {Values}
    {MMdavlKl}
    se1:=Format('%s%s%s',["",LabelP.Caption,"' "]);
    if not ControlEdit(EditP) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabValues;
      EditP.Color:=ClRed;
      Exit;
    End;
    perE:=StrToFloat(EditP.Text);
   if CheckP.Checked then begin
    if (perE<-50) or (perE>50) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-50',MsgErr_or,'>50).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabValues;
      EditP.Color:=ClRed;
      Exit;
     end;
    end else begin
    if (perE<500) or (perE>850) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<500',MsgErr_or,'>850).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabValues;
      EditP.Color:=ClRed;
      Exit;
     end;
    end;
    {OtVlagKl}
    se1:=Format('%s%s%s',["",LabelHu.Caption,"' "]);
    if not ControlEdit(EditHu) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabValues;
      EditHu.Color:=ClRed;
      Exit;
    End;
    perE:=StrToFloat(EditHu.Text);
   if CheckHu.Checked then begin
    if (perE<-50) or (perE>50) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-50',MsgErr_or,'>50).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabValues;
      EditHu.Color:=ClRed;
      Exit;
     end;
    end else begin
    if (perE<0.1) or (perE>100) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0.1',MsgErr_or,'>100).']);
      GetMyInvalidDlg(msg);
      Pages1.ActivePage:=TabValues;
      EditHu.Color:=ClRed;
      Exit;
     end;
    end;
     {DeviceOrient}
```

```
 se1:=Format('%s%s%s',["",LabelFi.Caption,"' "]);
 if not ControlEdit(EditDeviceOrient) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabValues;
   EditDeviceOrient.Color:=ClRed;
   Exit;
End;
perE:=StrToFloat(EditDeviceOrient.Text);
if (perE<0) or (perE>=360) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>=360).']);
  GetMyInvalidDlg(Msg);
   Pages1.ActivePage:=TabValues;
   EditDeviceOrient.Color:=ClRed;
   Exit;
 end;
{GamX}
se1:=Format('%s%s%s',["",LabelGx.Caption,"' "]);
if not ControlEdit(EditGx) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabValues;
   EditGx.Color:=ClRed;
   Exit;
End;
perE:=StrToFloat(EditGx.Text);
if (perE<-75) or (perE>75) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-75',MsgErr_or,'>75).']);
  GetMyInvalidDlg(Msg);
   Pages1.ActivePage:=TabValues;
   EditGx.Color:=ClRed;
   Exit;
 end;
 {GamY}
se1:=Format('%s%s%s',["",LabelGy.Caption,"' "]);
if not ControlEdit(EditGy) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabValues;
   EditGy.Color:=ClRed;
   Exit;
End;
perE:=StrToFloat(EditGy.Text);
if (perE<-75) or (perE>75) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-75',MsgErr_or,'>75).']);
  GetMyInvalidDlg(Msg);
   Pages1.ActivePage:=TabValues;
   EditGy.Color:=ClRed;
   Exit;
 end;
{DataDir}
if not DirectoryExists(EditDataDir.Text) then
if not CreateDir(EditDataDir.Text) then begin
  Msg:=Format('%s%s%s',[Msg_NoDir,EditDataDir.Text,'.']);
  GetMyInvalidDlg(Msg);
  Pages1.ActivePage:=TabValues;
  EditDataDir.Color:=ClRed;
  Exit;
  end;
{Filteres}
```

```
      {Amax}
    se1:=Format('%s%s%s',["",LabelAmax.Caption,"' "]);
    if not ControlEdit(EditAmax) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabFiltr;
      EditAmax.Color:=ClRed;
      Exit;
    End;
    perL:=StrToInt(EditAmax.Text);
    if (perL<3000) or (perL>65535) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<3000',MsgErr_or,'>65535).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabFiltr;
      EditAmax.Color:=ClRed;
      Exit;
     end;
    {Amin}
    se1:=Format('%s%s%s',["",LabelAmin.Caption,"' "]);
    if not ControlEdit(EditAmin) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabFiltr;
      EditAmin.Color:=ClRed;
      Exit;
    End;
    perL2:=perL;
    Svar:=IntToStr(perL2);
    perL:=StrToInt(EditAmin.Text);
    if perL>=perL2 then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(>=',Svar,').']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabFiltr;
      EditAmin.Color:=ClRed;
      Exit;
     end;
    {IntWinFiltr1}
    se1:=Format('%s%s%s',["",LabelIntWinFiltr1.Caption,"' "]);
    if not ControlEdit(EditIntWinFiltr1) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabFiltr;
      EditIntWinFiltr1.Color:=ClRed;
      Exit;
    End;
    perL:=StrToInt(EditIntWinFiltr1.Text);
    if (perL<10) or (perL>2000) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<10',MsgErr_or,'>2000).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabFiltr;
      EditIntWinFiltr1.Color:=ClRed;
      Exit;
     end;
  if (ParolCurrent=Par_System) or (ParolCurrent=Par_Standart) then Begin
     {Thermoanemometer}
    {Fkg}
    se1:=Format('%s%s%s',["",LabelFkg.Caption,"' "]);
    if not ControlEdit(EditFkg) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabUGI;
```

```
    EditFkg.Color:=ClRed;
    Exit;
  End;
  perE:=StrToFloat(EditFkg.Text);
  if (perE<10) or (perE>80) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<10',MsgErr_or,'>80).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditFkg.Color:=ClRed;
     Exit;
   end;
   {Rast1}
  se1:=Format('%s%s%s',["",LabelRast1.Caption,"' "]);
  if not ControlEdit(EditRast1) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabUGI;
     EditRast1.Color:=ClRed;
     Exit;
  End;
  perE:=StrToFloat(EditRast1.Text);
  if (perE<100) or (perE>300) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<100',MsgErr_or,'>300).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditRast1.Color:=ClRed;
     Exit;
   end;
   {Rast2}
   se1:=Format('%s%s%s',["",LabelRast2.Caption,"' "]);
  if not ControlEdit(EditRast2) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabUGI;
     EditRast2.Color:=ClRed;
     Exit;
  End;
  perE:=StrToFloat(EditRast2.Text);
  if (perE<100) or (perE>300) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<100',MsgErr_or,'>300).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditRast2.Color:=ClRed;
     Exit;
   end;
   {Rast3}
  se1:=Format('%s%s%s',["",LabelRast3.Caption,"' "]);
  if not ControlEdit(EditRast3) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabUGI;
     EditRast3.Color:=ClRed;
     Exit;
  End;
  perE:=StrToFloat(EditRast3.Text);
  if (perE<100) or (perE>300) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<100',MsgErr_or,'>300).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
```

```
    EditRast3.Color:=ClRed;
    Exit;
  end;
  {Rast4}
se1:=Format('%s%s%s',["",LabelRast4.Caption,"' "]);
if not ControlEdit(EditRast4) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditRast4.Color:=ClRed;
    Exit;
End;
perE:=StrToFloat(EditRast4.Text);
if (perE<100) or (perE>300) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<100',MsgErr_or,'>300).']);
  GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabUGI;
    EditRast4.Color:=ClRed;
    Exit;
 end;
  {Tcalibr}
se1:=Format('%s%s%s',["",LabelTCalibr.Caption,"' "]);
if not ControlEdit(EditTcalibr) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditTcalibr.Color:=ClRed;
    Exit;
End;
perE:=StrToFloat(EditTcalibr.Text);
if (perE<-55) or (perE>55) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-55',MsgErr_or,'>55).']);
  GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabUGI;
    EditTcalibr.Color:=ClRed;
    Exit;
 end;
{D1}
se1:=Format('%s%s%s',["",LabelD1.Caption,"' "]);
if not ControlEdit(EditD1) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditD1.Color:=ClRed;
    Exit;
End;
perL:=StrToInt(EditD1.Text);
if (perL<-32000) or (perL>32000) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-32000',MsgErr_or,'>32000).']);
  GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabUGI;
    EditD1.Color:=ClRed;
    Exit;
 end;
  {D2}
se1:=Format('%s%s%s',["",LabelD2.Caption,"' "]);
if not ControlEdit(EditD2) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditD2.Color:=ClRed;
```

```
    Exit;
  End;
  perL:=StrToInt(EditD2.Text);
  if (perL<-32000) or (perL>32000) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-32000',MsgErr_or,'>32000).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditD2.Color:=ClRed;
     Exit;
   end;
   {D3}
  se1:=Format('%s%s%s',["",LabelD3.Caption,"' "]);
  if not ControlEdit(EditD3) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabUGI;
     EditD3.Color:=ClRed;
     Exit;
  End;
  perL:=StrToInt(EditD3.Text);
  if (perL<-32000) or (perL>32000) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-32000',MsgErr_or,'>32000).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditD3.Color:=ClRed;
     Exit;
   end;
   {D4}
  se1:=Format('%s%s%s',["",LabelD4.Caption,"' "]);
  if not ControlEdit(EditD4) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabUGI;
     EditD4.Color:=ClRed;
     Exit;
  End;
  perL:=StrToInt(EditD4.Text);
  if (perL<-32000) or (perL>32000) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-32000',MsgErr_or,'>32000).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditD4.Color:=ClRed;
     Exit;
   end;
   {Tcoef}
  se1:=Format('%s%s%s',["",LabelTcoef.Caption,"' "]);
  if not ControlEdit(EditTcoef) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabUGI;
     EditTcoef.Color:=ClRed;
     Exit;
  End;
  perE:=StrToFloat(EditTcoef.Text);
  if (perE<0.01) or (perE>100) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0.01',MsgErr_or,'>100).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabUGI;
     EditTcoef.Color:=ClRed;
```

```
    Exit;
  end;
  {UGI_CoefZat}
 se1:=Format('%s%s%s',["",LabKoefZat.Caption,"' "]);
 if not ControlEdit(EditKoefZat) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditKoefZat.Color:=ClRed;
    Exit;
 End;
 perE:=StrToFloat(EditKoefZat.Text);
 if (perE<0.01) or (perE>=1) then
  begin
   Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0.01',MsgErr_or,'>=1).']);
   GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabUGI;
    EditKoefZat.Color:=ClRed;
    Exit;
  end;
  {UGI_hor}
  se1:=Format('%s%s%s',["",LabelUGI_hor.Caption,"' "]);
  if not ControlEdit(EditUGI_hor) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditUGI_hor.Color:=ClRed;
    Exit;
 End;
 perE:=StrToFloat(EditUGI_hor.Text);
 if (perE<0) or (perE>=360) then
  begin
   Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>=360).']);
   GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabUGI;
    EditUGI_hor.Color:=ClRed;
    Exit;
  end;
  {Zm}
 se1:=Format('%s%s%s',["",LabelZm.Caption,"' "]);
 if not ControlEdit(EditZm) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditZm.Color:=ClRed;
    Exit;
 End;
 perE:=StrToFloat(EditZm.Text);
 if (perE<=0) or (perE>500) then
  begin
   Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<=0',MsgErr_or,'>500).']);
   GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabUGI;
    EditZm.Color:=ClRed;
    Exit;
  end;
  {SetMinVolt}
  se1:=Format('%s%s%s',["",LabelSetVolt.Caption,"' "]);
  if not ControlEdit(EditSetVolt) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabUGI;
    EditSetVolt.Color:=ClRed;
    Exit;
```

```
End;
perE:=StrToFloat(EditSetVolt.Text);
if (perE<8) or (perE>30) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<8',MsgErr_or,'>30).']);
  GetMyInvalidDlg(Msg);
   Pages1.ActivePage:=TabUGI;
   EditSetVolt.Color:=ClRed;
   Exit;
 end;
{Sensores}
{P_P0}
 se1:=Format('%s%s%s',["",LabelP_P0.Caption," "]);
 if not ControlEdit(EditP_P0) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabSensor;
   EditP_P0.Color:=ClRed;
   Exit;
End;
perE:=StrToFloat(EditP_P0.Text);
if (perE<500) or (perE>800) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<500',MsgErr_or,'>800).']);
  GetMyInvalidDlg(Msg);
   Pages1.ActivePage:=TabSensor;
   EditP_P0.Color:=ClRed;
   Exit;
 end;
{P_x0}
 se1:=Format('%s%s%s',["",LabelP_x0.Caption," "]);
 if not ControlEdit(EditP_x0) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabSensor;
   EditP_x0.Color:=ClRed;
   Exit;
End;
perE:=StrToFloat(EditP_x0.Text);
if (perE<=0) or (perE>=65535) then
 begin
  Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<=0',MsgErr_or,'>=65535).']);
  GetMyInvalidDlg(Msg);
   Pages1.ActivePage:=TabSensor;
   EditP_x0.Color:=ClRed;
   Exit;
 end;
{P_ln}
 se1:=Format('%s%s%s',["",LabP_ln.Caption," "]);
 if not ControlEdit(EditP_ln) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabSensor;
   EditP_ln.Color:=ClRed;
   Exit;
End;
StrToFloat(EditP_ln.Text);
{P_T0}
 se1:=Format('%s%s%s',["",LabP_T0.Caption," "]);
 if not ControlEdit(EditP_T0) then Begin
   GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
   Pages1.ActivePage:=TabSensor;
   EditP_T0.Color:=ClRed;
```

```
    Exit;
  End;
  perE:=StrToFloat(EditP_T0.Text);
  if (perE<-50) or (perE>50) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-50',MsgErr_or,'>50).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabSensor;
     EditP_T0.Color:=ClRed;
     Exit;
   end;
   {PT_ln}
   se1:=Format('%s%s%s',["",LabPT_ln.Caption,"' "]);
   if not ControlEdit(EditPT_ln) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabSensor;
     EditPT_ln.Color:=ClRed;
     Exit;
  End;
  StrToFloat(EditPT_ln.Text);
  {PT_kv}
   se1:=Format('%s%s%s',["",LabPT_kv.Caption,"' "]);
   if not ControlEdit(EditPT_kv) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabSensor;
     EditPT_kv.Color:=ClRed;
     Exit;
  End;
  StrToFloat(EditPT_kv.Text);
   {PT_kb}
   se1:=Format('%s%s%s',["",LabPT_kb.Caption,"' "]);
   if not ControlEdit(EditPT_kb) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabSensor;
     EditPT_kb.Color:=ClRed;
     Exit;
  End;
  StrToFloat(EditPT_kb.Text);
  {r_r0}
   se1:=Format('%s%s%s',["",Labr_r0.Caption,"' "]);
   if not ControlEdit(Editr_r0) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabSensor;
     Editr_r0.Color:=ClRed;
     Exit;
  End;
  perE:=StrToFloat(Editr_r0.Text);
  if (perE<0) or (perE>100) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>100).']);
    GetMyInvalidDlg(Msg);
     Pages1.ActivePage:=TabSensor;
     Editr_r0.Color:=ClRed;
     Exit;
   end;
  {r_X0}
   se1:=Format('%s%s%s',["",Labr_X0.Caption,"' "]);
   if not ControlEdit(Editr_X0) then Begin
     GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
     Pages1.ActivePage:=TabSensor;
```

```
    Editr_r0.Color:=ClRed;
    Exit;
  End;
  perE:=StrToFloat(Editr_X0.Text);
  if (perE<0) or (perE>=65535) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>=65535).']);
    GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabSensor;
    Editr_X0.Color:=ClRed;
    Exit;
   end;
  {r_ln}
   se1:=Format('%s%s%s',["",Labr_ln.Caption,'" ']);
   if not ControlEdit(Editr_ln) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    Editr_ln.Color:=ClRed;
    Exit;
  End;
  StrToFloat(Editr_ln.Text);
  {r_kv}
   se1:=Format('%s%s%s',["",Labr_kv.Caption,'" ']);
   if not ControlEdit(Editr_kv) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    Editr_kv.Color:=ClRed;
    Exit;
  End;
  StrToFloat(Editr_kv.Text);
   {Tr_T0}
   se1:=Format('%s%s%s',["",LabTr_T0.Caption,'" ']);
   if not ControlEdit(EditTr_T0) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    EditTr_T0.Color:=ClRed;
    Exit;
  End;
  perE:=StrToFloat(EditTr_T0.Text);
  if (perE<-50) or (perE>50) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-50',MsgErr_or,'>50).']);
    GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabSensor;
    EditTr_T0.Color:=ClRed;
    Exit;
   end;
  {Tr_x0}
   se1:=Format('%s%s%s',["",LabTr_x0.Caption,'" ']);
   if not ControlEdit(EditTr_x0) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    EditTr_x0.Color:=ClRed;
    Exit;
  End;
  perE:=StrToFloat(EditTr_x0.Text);
  if (perE<=0) or (perE>=65535) then
   begin
    Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<=0',MsgErr_or,'>=65535).']);
    GetMyInvalidDlg(Msg);
```

```
    Pages1.ActivePage:=TabSensor;
    EditTr_x0.Color:=ClRed;
    Exit;
  end;
 {Tr_ln}
  se1:=Format('%s%s%s',["",LabTr_ln.Caption,"' "]);
  if not ControlEdit(EditTr_ln) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    EditTr_ln.Color:=ClRed;
    Exit;
 End;
 strToFloat(EditTr_ln.Text);
  {Tr_kv}
  se1:=Format('%s%s%s',["",LabTr_kv.Caption,"' "]);
  if not ControlEdit(EditTr_kv) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    EditTr_kv.Color:=ClRed;
    Exit;
 End;
 StrToFloat(EditTr_kv.Text);
 {G_x0}
  se1:=Format('%s%s%s',["",LabelG_x0.Caption,"' "]);
  if not ControlEdit(EditG_x0) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    EditG_x0.Color:=ClRed;
    Exit;
 End;
 perE:=StrToFloat(EditG_x0.Text);
 if (perE<0) or (perE>65000) then
  begin
   Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>65000).']);
   GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabSensor;
    EditG_x0.Color:=ClRed;
    Exit;
  end;
 {G_y0}
  se1:=Format('%s%s%s',["",LabelG_y0.Caption,"' "]);
  if not ControlEdit(EditG_y0) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
    Pages1.ActivePage:=TabSensor;
    EditG_y0.Color:=ClRed;
    Exit;
 End;
 perE:=StrToFloat(EditG_y0.Text);
 if (perE<0) or (perE>65000) then
  begin
   Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>65000).']);
   GetMyInvalidDlg(Msg);
    Pages1.ActivePage:=TabSensor;
    EditG_y0.Color:=ClRed;
    Exit;
  end;
 {G_xd}
  se1:=Format('%s%s%s',["",LabelG_xd.Caption,"' "]);
  if not ControlEdit(EditG_xd) then Begin
    GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
```

```
      Pages1.ActivePage:=TabSensor;
      EditG_xd.Color:=ClRed;
      Exit;
    End;
    perE:=StrToFloat(EditG_xd.Text);
    if (perE<0) or (perE>65000) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>65000).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabSensor;
      EditG_xd.Color:=ClRed;
      Exit;
     end;
    {G_y0}
     se1:=Format('%s%s%s',["",LabelG_yd.Caption,"' "]);
     if not ControlEdit(EditG_yd) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabSensor;
      EditG_yd.Color:=ClRed;
      Exit;
    End;
    perE:=StrToFloat(EditG_yd.Text);
    if (perE<0) or (perE>65000) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>65000).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabSensor;
      EditG_yd.Color:=ClRed;
      Exit;
     end;
    {G_hor}
     se1:=Format('%s%s%s',["",LabelG_hor.Caption,"' "]);
     if not ControlEdit(EditG_hor) then Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      Pages1.ActivePage:=TabSensor;
      EditG_hor.Color:=ClRed;
      Exit;
    End;
    perE:=StrToFloat(EditG_hor.Text);
    if (perE<0) or (perE>360) then
     begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<0',MsgErr_or,'>360).']);
      GetMyInvalidDlg(Msg);
      Pages1.ActivePage:=TabSensor;
      EditG_hor.Color:=ClRed;
      Exit;
     end;
    End; {if Prioritet=PrioritetDef}
    Result:=true;
    except
    raise
    End;
  End;

  //**************************************************
  procedure TOptionsDlg.EditPKeyPress(Sender: TObject; var Key: Char);
  begin
    if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
  end;
```

```
procedure TOptionsDlg.EditHuKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditDeviceOrientKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditGxKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditGyKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditZmKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditFkgKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditAminKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditAmaxKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTCoefKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditZatenCoefKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditRast1KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditRast2KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
```

```
end;

procedure TOptionsDlg.EditRast3KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditRast4KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTCalibrKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditSpeedCalibrKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditD1KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditD2KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditD3KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditD4KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.SpinValFiltr1KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditIntWinFiltr1KeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

{*******************************}
procedure TOptionsDlg.EditPKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditP.Color:=ClWindow;
end;
```

```
procedure TOptionsDlg.EditHuKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditHu.Color:=ClWindow;
end;

procedure TOptionsDlg.EditHuClick(Sender: TObject);
begin
  EditHu.Color:=ClWindow;
end;

procedure TOptionsDlg.EditGxKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditGx.Color:=ClWindow;
end;

procedure TOptionsDlg.EditGxClick(Sender: TObject);
begin
  EditGx.Color:=ClWindow;
end;

procedure TOptionsDlg.EditGyKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditGy.Color:=ClWindow;
end;

procedure TOptionsDlg.EditGyClick(Sender: TObject);
begin
  EditGy.Color:=ClWindow;
end;

procedure TOptionsDlg.EditZmKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditZm.Color:=ClWindow;
end;

procedure TOptionsDlg.EditZmClick(Sender: TObject);
begin
  EditZm.Color:=ClWindow;
end;

procedure TOptionsDlg.EditFkgKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditFkg.Color:=ClWindow;
end;

procedure TOptionsDlg.EditFkgClick(Sender: TObject);
begin
  EditFkg.Color:=ClWindow;
end;

procedure TOptionsDlg.EditAminKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditAmin.Color:=ClWindow;
```

```
end;

procedure TOptionsDlg.EditAminClick(Sender: TObject);
begin
  EditAmin.Color:=ClWindow;
end;

procedure TOptionsDlg.EditAmaxKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditAmax.Color:=ClWindow;
end;

procedure TOptionsDlg.EditAmaxClick(Sender: TObject);
begin
  EditAmax.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinValFiltr1Click(Sender: TObject);
begin
  SpinValFiltr1.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinValFiltr1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  SpinValFiltr1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditIntWinFiltr1Click(Sender: TObject);
begin
  EditIntWinFiltr1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditIntWinFiltr1KeyDown(Sender: TObject;
  var Key: Word; Shift: TShiftState);
begin
  EditIntWinFiltr1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast1Click(Sender: TObject);
begin
  EditRast1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditRast1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast2Click(Sender: TObject);
begin
  EditRast2.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditRast2.Color:=ClWindow;
```

```
end;

procedure TOptionsDlg.EditRast3Click(Sender: TObject);
begin
  EditRast3.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast3KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditRast3.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast4Click(Sender: TObject);
begin
  EditRast4.Color:=ClWindow;
end;

procedure TOptionsDlg.EditRast4KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditRast4.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTCalibrClick(Sender: TObject);
begin
  EditTCalibr.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTCalibrKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTCalibr.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD1Click(Sender: TObject);
begin
  EditD1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditD1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD2Click(Sender: TObject);
begin
  EditD2.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditD2.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD3Click(Sender: TObject);
begin
  EditD3.Color:=ClWindow;
```

```
end;

procedure TOptionsDlg.EditD3KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditD3.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD4Click(Sender: TObject);
begin
  EditD4.Color:=ClWindow;
end;

procedure TOptionsDlg.EditD4KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditD4.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinVolumEditClick(Sender: TObject);
begin
  SpinVolumEdit.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinVolumEditKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  SpinVolumEdit.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinVolumEditKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.SpinBoundEditClick(Sender: TObject);
begin
  SpinBoundEdit.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinBoundEditKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  SpinBoundEdit.Color:=ClWindow;
end;

procedure TOptionsDlg.SpinBoundEditKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTimer1KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditSetVoltKeyPress(Sender: TObject; var Key: Char);
begin
```

```
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTimer1Click(Sender: TObject);
begin
  EditTimer1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTimer1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTimer1.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPClick(Sender: TObject);
begin
  EditP.Color:=ClWindow;
end;

procedure TOptionsDlg.EditDataDirClick(Sender: TObject);
begin
  EditDataDir.Color:=ClWindow;
end;

procedure TOptionsDlg.EditDataDirKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditDataDir.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_P0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditP_x0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditP_lnKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditPT_lnKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditPT_kvKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditPT_kbKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;
```

```
procedure TOptionsDlg.Editr_r0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.Editr_X0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.Editr_lnKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.Editr_kvKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditG_x0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditG_y0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditG_xdKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditG_ydKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditG_horKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditP_P0Click(Sender: TObject);
begin
  EditP_P0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_P0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditP_P0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_x0Click(Sender: TObject);
begin
  EditP_x0.Color:=ClWindow;
end;
```

```
procedure TOptionsDlg.EditP_x0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditP_x0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_lnClick(Sender: TObject);
begin
  EditP_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditP_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPT_lnClick(Sender: TObject);
begin
  EditPT_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPT_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditPT_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPT_kvClick(Sender: TObject);
begin
  EditPT_kv.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPT_kvKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditPT_kv.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPT_kbClick(Sender: TObject);
begin
  EditPT_kb.Color:=ClWindow;
end;

procedure TOptionsDlg.EditPT_kbKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditPT_kb.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_r0Click(Sender: TObject);
begin
  Editr_r0.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_r0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Editr_r0.Color:=ClWindow;
```

```
end;

procedure TOptionsDlg.Editr_X0Click(Sender: TObject);
begin
  Editr_X0.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_X0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Editr_X0.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_lnClick(Sender: TObject);
begin
  Editr_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Editr_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_kvClick(Sender: TObject);
begin
  Editr_kv.Color:=ClWindow;
end;

procedure TOptionsDlg.Editr_kvKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Editr_kv.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_x0Click(Sender: TObject);
begin
  EditG_x0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_x0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditG_x0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_y0Click(Sender: TObject);
begin
  EditG_y0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_y0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditG_y0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_xdClick(Sender: TObject);
begin
  EditG_xd.Color:=ClWindow;
```

```
end;

procedure TOptionsDlg.EditG_xdKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditG_xd.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_ydClick(Sender: TObject);
begin
  EditG_yd.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_ydKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditG_yd.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_horClick(Sender: TObject);
begin
  EditG_hor.Color:=ClWindow;
end;

procedure TOptionsDlg.EditG_horKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditG_hor.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_T0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditP_T0Click(Sender: TObject);
begin
  EditP_T0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditP_T0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditP_T0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_T0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTr_X0KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTr_lnKeyPress(Sender: TObject; var Key: Char);
begin
   if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;
```

```
procedure TOptionsDlg.EditTr_kvKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditTr_T0Click(Sender: TObject);
begin
  EditTr_T0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_T0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTr_T0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_X0Click(Sender: TObject);
begin
  EditTr_X0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_X0KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTr_X0.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_lnClick(Sender: TObject);
begin
  EditTr_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_lnKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTr_ln.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_kvClick(Sender: TObject);
begin
  EditTr_kv.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTr_kvKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTr_kv.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTcoefClick(Sender: TObject);
begin
  EditTcoef.Color:=ClWindow;
end;

procedure TOptionsDlg.EditTcoefKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
   EditTcoef.Color:=ClWindow;
end;
```

```
procedure TOptionsDlg.EditSetVoltClick(Sender: TObject);
begin
  EditSetVolt.Color:=ClWindow;
end;

procedure TOptionsDlg.EditSetVoltKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditSetVolt.Color:=ClWindow;
end;

procedure TOptionsDlg.EditUGI_horKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditUGI_horClick(Sender: TObject);
begin
  EditUGI_hor.Color:=ClWindow;
end;

procedure TOptionsDlg.EditUGI_horKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditUGI_hor.Color:=ClWindow;
end;

procedure TOptionsDlg.EditKoefZatKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.EditKoefZatClick(Sender: TObject);
begin
  EditKoefZat.Color:=ClWindow;
end;

procedure TOptionsDlg.EditKoefZatKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
   EditKoefZat.Color:=ClWindow;
end;

procedure TOptionsDlg.EditDeviceOrientClick(Sender: TObject);
begin
  EditDeviceOrient.Color:=ClWindow;
end;

procedure TOptionsDlg.EditDeviceOrientKeyDown(Sender: TObject;
  var Key: Word; Shift: TShiftState);
begin
  EditDeviceOrient.Color:=ClWindow;
end;

procedure TOptionsDlg.EditNumberTeleFonKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9','-'])=false then Key:=Chr(0);
end;
```

```
procedure TOptionsDlg.BaudRateBoxKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TOptionsDlg.BaudRateBoxClick(Sender: TObject);
begin
  BaudRateBox.Color:=ClWindow;
end;

procedure TOptionsDlg.BaudRateBoxKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  BaudRateBox.Color:=ClWindow;
end;

procedure TOptionsDlg.EditNumServerKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit RunIzmer;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
    Buttons, ExtCtrls, Spin;

type
  TRunIzmerenDlg = class(TForm)
    Panel1: TPanel;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    RegimIzmerenRG: TRadioGroup;
    GlobalBox: TGroupBox;
    EditDur: TSpinEdit;
    LabTot1: TLabel;
    AddBox: TGroupBox;
    CheckDB: TCheckBox;
    ChPhone: TCheckBox;
    CheckLoadProg: TCheckBox;
    procedure EditDurClick(Sender: TObject);
    procedure EditDurKeyPress(Sender: TObject; var Key: Char);
    procedure EditDurKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
    Procedure GetData;
    Function ValidData:Boolean;
    Function Execute:Boolean;
  public
    { Public declarations }
  end;
```

```
var
  RunIzmerenDlg: TRunIzmerenDlg;

function GetParamIzmeren:WordBool;

implementation
{$R *.DFM}
Uses UnAMKini, UnAMKsys, ApllDlgs;

{****************************}
function GetParamIzmeren:WordBool;
begin
  Screen.Cursor:=crHourGlass;
  try
  Application.CreateForm(TRunIzmerenDlg, RunIzmerenDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  Result:=RunIzmerenDlg.Execute;
  finally
  RunIzmerenDlg.Free;
  end;
end;

{****************************}
procedure TRunIzmerenDlg.FormCreate(Sender: TObject);
begin
  AvtoZapuskIzm:=false;
  EditDur.Value:=TimeUsredn div 60;
  RegimIzmerenRG.ItemIndex:=RegimIzmer;
  if FileExists(BasaFileName) then CheckDB.Checked:=IsSaveBazaData
  else begin
    CheckDB.Checked:=false;
    CheckDB.Enabled:=false;
    end;
  if IsModem and IsConnectPhone then begin
    ChPhone.Visible:=true;
    ChPhone.Enabled:=true;
    ChPhone.Checked:=NotBreakPhone;
    end else begin
    ChPhone.Enabled:=false;
    ChPhone.Visible:=false;
    end;
end;

{*****************************************************************************}
Function TRunIzmerenDlg.Execute:Boolean;
Label 10;
Begin
  Result:=false;
10:if ShowModal=idOk Then
  Begin
    Result:=ValidData;
    if not Result then goto 10;
    GetData;
  End;
End;
```

```
{*************************************************************************}
Procedure TRunIzmerenDlg.GetData;
Begin
  try
  RegimIzmer:=RegimIzmerenRG.ItemIndex;
  TimeUsredn:=60*EditDur.Value;
  IsSaveBazaData:=CheckDB.Checked;
  AvtoZapuskIzm:=CheckLoadProg.Checked;
  if IsModem and IsConnectPhone then begin
    if ChPhone.Checked then NotBreakPhone:=true
                    else NotBreakPhone:=false;
    end;
  except
  raise;
  end;
End;

{*************************************************************************}
Function TRunIzmerenDlg.ValidData:Boolean;
var
  Msg,se1:String;
Begin
   try
   Result:=false;
   if (EditDur.Value in [1..20])=false then
    begin
      se1:=Format('%s%s%s',["'",LabTot1.Caption,"' "]);
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<1',MsgErr_or,'>20).']);
      GetMyInvalidDlg(Msg);
      EditDur.Color:=ClRed;
      Exit;
    end;
  Result:=true;
   except
   raise;
   end;
End;

{*************************************************************************}
procedure TRunIzmerenDlg.EditDurKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0)
end;

{*************************************************************************}
procedure TRunIzmerenDlg.EditDurClick(Sender: TObject);
begin
  EditDur.Color:=ClWindow;
end;

procedure TRunIzmerenDlg.EditDurKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditDur.Color:=ClWindow;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
```

```
unit AutSave;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, Dialogs, StdCtrls,
  Buttons, ExtCtrls;

type
  TAutSaveDlg = class(TForm)
    Panel1: TPanel;
    AutoBox: TGroupBox;
    BeginBox: TGroupBox;
    EditBegHour: TEdit;
    EditBegMin: TEdit;
    InterBox: TGroupBox;
    EditIntHour: TEdit;
    EditIntMin: TEdit;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    Label2: TLabel;
    Label3: TLabel;
    LabHour1: TLabel;
    LabMin1: TLabel;
    LabHour2: TLabel;
    LabMin2: TLabel;
    Label1: TLabel;
    procedure EditBegHourKeyPress(Sender: TObject; var Key: Char);
    procedure EditBegMinKeyPress(Sender: TObject; var Key: Char);
    procedure EditIntHourKeyPress(Sender: TObject; var Key: Char);
    procedure EditIntMinKeyPress(Sender: TObject; var Key: Char);
    procedure EditBegHourClick(Sender: TObject);
    procedure EditBegHourKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditBegMinClick(Sender: TObject);
    procedure EditBegMinKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditIntHourClick(Sender: TObject);
    procedure EditIntHourKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditIntMinClick(Sender: TObject);
    procedure EditIntMinKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    { Private declarations }
    Procedure SetData;
    Procedure GetData;
    Function ValidData:Boolean;
    Function Execute:Boolean;
  public
    { Public declarations }
  end;

var
  AutSaveDlg: TAutSaveDlg;

  function GetParamAutSave:WordBool;

implementation
{$R *.DFM}
Uses UnAMKini,ApllDlgs;
```

```
{****************************************}
function GetParamAutSave:WordBool;
begin
  Screen.Cursor:=crHourGlass;
  try
  Application.CreateForm(TAutSaveDlg, AutSaveDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  Result:=AutSaveDlg.Execute;
  finally
  AutSaveDlg.Free;
  end;
end;

{*****************************************************************************}
Function TAutSaveDlg.Execute:Boolean;
Label 10;
Begin
  Result:=false;
  SetData;
10:if ShowModal=idOk Then
  Begin
    Result:=ValidData;
    if not Result then goto 10;
    GetData;
  End;
End;

{*****************************************************************************}
Procedure TAutSaveDlg.SetData;
Begin
  try
  EditBegHour.Text:=IntToStr(BegHourAutSave);
  EditBegMin.Text:=IntToStr(BegMinAutSave);
  EditIntHour.Text:=IntToStr(InterHourAutSave);
  EditIntMin.Text:=IntToStr(InterMinAutSave);
  except
  raise
  end;
end;

{*****************************************************************************}
Procedure TAutSaveDlg.GetData;
Begin
  try
  BegHourAutSave:=StrToInt(EditBegHour.Text);
  BegMinAutSave:=StrToInt(EditBegMin.Text);
  InterHourAutSave:=StrToInt(EditIntHour.Text);
  InterMinAutSave:=StrToInt(EditIntMin.Text);
  except
  raise;
  end;
End;

{*****************************************************************************}
Function TAutSaveDlg.ValidData:Boolean;
var
```

```
  se1:String;
  t1:tDateTime;
  par,par2:Integer;
Begin
  try
  Result:=false;
  par:=StrToInt(EditBegHour.Text);
  se1:=Format('%s%s%s%s%s',["",BeginBox.Caption,': ',LabHour1.Caption,'" ']);
  if par>23 then
    begin
      GetMyInvalidDlg(Format('%s%s%s',[MsgErr_sf,se1,'(>23).']));
      EditBegHour.Color:=ClRed;
      Exit;
    end;
  par:=StrToInt(EditBegMin.Text);
  se1:=Format('%s%s%s%s%s',["",BeginBox.Caption,': ',LabMin1.Caption,'" ']);
  if par>59 then
    begin
      GetMyInvalidDlg(Format('%s%s%s',[MsgErr_sf,se1,'(>59).']));
      EditBegMin.Color:=ClRed;
      Exit;
    end;
  par:=StrToInt(EditIntHour.Text);
  se1:=Format('%s%s%s%s%s',["",InterBox.Caption,': ',LabHour2.Caption,'" ']);
  if par>23 then
    begin
      GetMyInvalidDlg(Format('%s%s%s',[MsgErr_sf,se1,'(>23).']));
      EditIntHour.Color:=ClRed;
      Exit;
    end;
  par2:=StrToInt(EditIntMin.Text);
  se1:=Format('%s%s%s%s%s',["",InterBox.Caption,': ',LabMin2.Caption,'" ']);
  if par2>59 then
    begin
      GetMyInvalidDlg(Format('%s%s%s',[MsgErr_sf,se1,'(>59).']));
      EditIntMin.Color:=ClRed;
      Exit;
    end;
  t1:=EncodeTime(par,par2,0,0);
  if EncodeTime(0,TimeUsredn div 60,0,0) > t1 then
    begin
      se1:=Format('%s%s%s',["",InterBox.Caption,'": ']);
      GetMyInvalidDlg(Format('%s%s%s',[MsgErr_sf,se1,MsgErr_Period]));
      EditIntMin.Color:=ClRed;
      Exit;
    end;
  Result:=true;
  except
  raise;
  end;
End;

procedure TAutSaveDlg.EditBegHourKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TAutSaveDlg.EditBegMinKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
```

```
end;

procedure TAutSaveDlg.EditIntHourKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TAutSaveDlg.EditIntMinKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;

procedure TAutSaveDlg.EditBegHourClick(Sender: TObject);
begin
  EditBegHour.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditBegHourKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditBegHour.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditBegMinClick(Sender: TObject);
begin
  EditBegMin.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditBegMinKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditBegMin.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditIntHourClick(Sender: TObject);
begin
  EditIntHour.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditIntHourKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditIntHour.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditIntMinClick(Sender: TObject);
begin
  EditIntMin.Color:=ClWindow;
end;

procedure TAutSaveDlg.EditIntMinKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditIntMin.Color:=ClWindow;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
```

```
unit RunCalib;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, Spin;

type
  TRunCalibratingDlg = class(TForm)
    Panel1: TPanel;
    CalibBox1: TGroupBox;
    LabTC: TLabel;
    LabTot1: TLabel;
    EditTC: TEdit;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    EditDur: TSpinEdit;
    procedure EditTCKeyPress(Sender: TObject; var Key: Char);
    procedure EditTCClick(Sender: TObject);
    procedure EditTCKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditDurKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure EditDurClick(Sender: TObject);
  private
    { Private declarations }
    Procedure SetData(T1:Real; Dur1:Word);
    Procedure GetData(var T1:Real; var Dur1:Word);
    Function ValidData:Boolean;
    Function Execute(var T1:Real; var Dur1:Word):Boolean;
  public
    { Public declarations }
  end;

var
  RunCalibratingDlg: TRunCalibratingDlg;

function GetParamSonicCalib(var T1:Real; var Dur1:Word):WordBool;

IMPLEMENTATION
{$R *.DFM}
uses UnAMKini,ApllDlgs;

{*****************************************************************************}
function GetParamSonicCalib(var T1:Real; var Dur1:Word):WordBool;
begin
  Screen.Cursor:=crHourGlass;
  try
  Application.CreateForm(TRunCalibratingDlg, RunCalibratingDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  Result:=RunCalibratingDlg.Execute(T1,Dur1);
  finally
  RunCalibratingDlg.Free;
  end;
end;

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

```
Function TRunCalibratingDlg.Execute(var T1:Real; var Dur1:Word):Boolean;
Label 10;
Begin
  Result:=false;
  SetData(T1,Dur1);
10:if ShowModal=mrOk Then
  Begin
    if ValidData=true then Result:=true
                  else goto 10;
    GetData(T1,Dur1);
  End;
End;

{*************************************************************************}
Procedure TRunCalibratingDlg.SetData(T1:Real; Dur1:Word);
Begin
  try
  EditDur.Value:=Dur1 div 60;
  EditTC.Text:=Format('%6.2f',[T1]);
  except
  raise;
  end;
End;

{*************************************************************************}
Procedure TRunCalibratingDlg.GetData(var T1:Real; var Dur1:Word);
Begin
  try
  Dur1:=60*EditDur.value;
  T1:=StrToFloat(EditTC.Text);
  except
  raise;
  end;
End;

{*************************************************************************}
Function TRunCalibratingDlg.ValidData:Boolean;
var
  Msg,se1:String;
  perE:Extended;
Begin
   try
   Result:=false;
    {DuratCalibr}
   if (EditDur.Value in [1..20])=false then
   begin
      se1:=Format('%s%s%s',["",LabTot1.Caption," "]);
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<1',MsgErr_or,'>20).']);
      GetMyInvalidDlg(Msg);
      EditDur.Color:=ClRed;
      Exit;
    end;
    {Tkalibr}
   se1:=Format('%s%s%s',["",LabTC.Caption," "]);
   if ControlEdit(EditTC) then Begin
   perE:=StrToFloat(EditTC.Text);
   if (perE<-50) or (perE>50) then
    begin
      Msg:=Format('%s%s%s%s%s',[MsgErr_sf,se1,'(<-50',MsgErr_or,'>50).']);
      GetMyInvalidDlg(Msg);
```

```
      EditTC.Color:=ClRed;
      Exit;
    end;
    End else Begin
      GetMyInvalidDlg(Format('%s%s',[MsgErr_so,se1]));
      EditTC.Color:=ClRed;
      Exit;
    End;
   Result:=true;
   except
   raise;
   end;
End;

{*****************************************************************************}
procedure TRunCalibratingDlg.EditTCKeyPress(Sender: TObject;
  var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9',',','.','-'])=false then Key:=Chr(0);
end;

{*****************************************************************************}
procedure TRunCalibratingDlg.EditTCClick(Sender: TObject);
begin
  EditTC.Color:=ClWindow;
end;
{____________________}
procedure TRunCalibratingDlg.EditTCKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditTC.Color:=ClWindow;
end;
{____________________}
procedure TRunCalibratingDlg.EditDurClick(Sender: TObject);
begin
  EditDur.Color:=ClWindow;
end;
{____________________}
procedure TRunCalibratingDlg.EditDurKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  EditDur.Color:=ClWindow;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit TestPPU;
interface
uses Windows, SysUtils, Classes, Controls, Forms, StdCtrls,
    Spin, Buttons, ExtCtrls;

type
  TTestPPUdlg = class(TForm)
    Bevel2: TBevel;
    CloseBtn: TBitBtn;
    ParamBox: TGroupBox;
    EditTime: TSpinEdit;
```

```
    LabTime: TLabel;
    Label9: TLabel;
    CommandBox: TGroupBox;
    CommandBtn: TBitBtn;
    GroupBox2: TGroupBox;
    EditRxStr: TEdit;
    Label7: TLabel;
    Memo1: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    EditVal: TEdit;
    Label4: TLabel;
    EditTxStr: TEdit;
    ComList: TComboBox;
    procedure EditTimeKeyPress(Sender: TObject; var Key: Char);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ComListChange(Sender: TObject);
    procedure EditTimeChange(Sender: TObject);
    procedure CommandBtnClick(Sender: TObject);
  private
    { Private declarations }
    procedure AutoUpdateTxStr;
    procedure UpdateNoOtvet;
    procedure DiagnozOtvet(const Buffer:array of Byte);
    procedure RunTestPPU;
  public
    { Public declarations }
  end;

var
  TestPPUdlg: TTestPPUdlg;

procedure GetTestPPUdlg;

implementation
{$R *.dfm}
uses UnAMKini,ApllDlgs,UnAMKport,UnData;

{**************************************************}
procedure GetTestPPUdlg;
begin
  try
  Screen.Cursor:=crHourGlass;
  Application.CreateForm(TTestPPUdlg, TestPPUdlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  TestPPUdlg.ShowModal;
  finally
  TestPPUdlg.Free;
  end;
end;

//**************************************************
procedure TTestPPUdlg.EditTimeKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in [Chr(8),'0'..'9'])=false then Key:=Chr(0);
end;
```

```
//***********************************************
procedure TTestPPUdlg.FormCreate(Sender: TObject);
begin
  AutoUpdateTxStr;
  StartAMKport_PPU(COM_BaudRate,true);
  if not IsOpenCOMport then begin
    UpdateNoOtvet;
    Exit;
    end;
end;

//***********************************************
procedure TTestPPUdlg.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Close_AMKport(Handle_AMK,IsOpenCOMport);
end;

//**********************************************
procedure TTestPPUdlg.AutoUpdateTxStr;
var
  i:Byte;
  Buffer:array [0..10] of Byte;
  t1:Word;
begin
  Buffer[0]:=80;
  Buffer[1]:=175;
  for i:=2 to 10 do
    Buffer[i]:=0;
  if ComList.ItemIndex<=6 then begin
    if EditTime.Value=0 then t1:=1
    else t1:=EditTime.Value*60;
    Buffer[2]:=Lo(t1);
    Buffer[3]:=Hi(t1);
    end;
  Buffer[4]:=ComList.ItemIndex;
  for i:=2 to 9 do
    Buffer[10]:=Buffer[10] xor Buffer[i];
  EditTxStr.Text:=IntToHex(Buffer[0],2);;
  for i:=1 to 10 do
    EditTxStr.Text:=EditTxStr.Text+' '+IntToHex(Buffer[i],2);
end;

//*********************************************
procedure TTestPPUdlg.ComListChange(Sender: TObject);
begin
  AutoUpdateTxStr;
end;

//********************************************
procedure TTestPPUdlg.EditTimeChange(Sender: TObject);
begin
  AutoUpdateTxStr;
end;

//***********************************
procedure TTestPPUdlg.RunTestPPU;
var
  WriteBytes:DWord;
  Buffer:array of Byte;
```

```
  k:Byte;
  i:Integer;
begin
  try
  if Length(EditTxStr.Text)<2 then Exit;
  try
  i:=0;
  k:=1;
  while k+1<=Length(EditTxStr.Text) do begin
    if Buffer=nil then SetLength(Buffer,1)
                else SetLength(Buffer,Length(Buffer)+1);
    Buffer[i]:=StrToInt('$'+Copy(EditTxStr.Text,k,2));
    k:=k+3;
    inc(i);
    end;
  for i:=0 to High(Buffer) do
  WriteFile(Handle_AMK,Buffer[i],1,WriteBytes,@OverLap_AMK);
  finally
  Buffer:=nil;
  end;
  except
  raise
  end;
end;

//**************************************************
procedure TTestPPUdlg.UpdateNoOtvet;
begin
  EditRxStr.Text:='';
  EditVal.Text:='';
  Memo1.Clear;
  Memo1.Lines.Add(MsgPPUErr_Kod+'9.');
  Memo1.Lines.Add(MsgPPUErr9_0+MsgErr9_1);
end;

//**************************************************
procedure TTestPPUdlg.DiagnozOtvet(const Buffer:array of Byte);
var
  i:Integer;
  Flag,Num1,Con1,ks:Byte;
  Value1:Single;
  IS4:array [0..3] of Byte ABSOLUTE Value1;
begin
  Flag:=1;
  Num1:=255;
  EditVal.Text:='';
  Memo1.Clear;
  EditRxStr.Text:=IntToHex(Buffer[0],2);
  for i:=1 to High(Buffer) do
    EditRxStr.Text:=EditRxStr.Text+' '+IntToHex(Buffer[i],2);
  if (Length(Buffer)<>11) or (Buffer[0]<>$AA) or (Buffer[1]<>$55) then begin
  Memo1.Lines.Add(MsgPPUErr_Kod+'5.');
  Memo1.Lines.Add(MsgPPUErr5);
  Flag:=5;
  end else begin
  if (buffer[2]=$FF) and (buffer[3]=$FF) then begin
  Memo1.Lines.Add(MsgPPUErr_Kod+'4.');
  Memo1.Lines.Add(MsgPPUErr4);
  Flag:=4;
  end else begin
```

```
Num1:=StrToInt('$'+Copy(EditTxStr.Text,13,2));
if buffer[4]<>Num1 then begin
Memo1.Lines.Add(MsgPPUErr_Kod+'3.');
Memo1.Lines.Add(MsgPPUErr3);
Flag:=3;
end else begin
if (Num1=128) or (Num1=64) then begin
Con1:=StrToInt('$'+Copy(EditTxStr.Text,16,2));
if buffer[5]<>Con1 then begin
Memo1.Lines.Add(MsgPPUErr_Kod+'2.');
Memo1.Lines.Add(MsgPPUErr2);
Flag:=2;
end;//if buffer[5]<>Con1
end;//if (Num1=128)...
end;//if buffer[4]<>Num1
end;//if (buffer[2]=$FF)....
end;//if (Length(Buffer)<>11)....
if Flag=1 then begin
ks:=0;
for i:=2 to High(Buffer)-1 do
  ks:=ks xor Buffer[i];
  if (buffer[High(Buffer)]<>ks) then begin
  Memo1.Lines.Add(MsgPPUErr_Kod+'1.');
  Memo1.Lines.Add(MsgErr1);
  end else begin
  Memo1.Lines.Add(MsgPPUErr_Kod+'0.');
  Memo1.Lines.Add(MsgPPUErr0);
  end;
end;//if Result=1
if (Num1<=6) and (Flag<=1) then begin
IS4[3]:=buffer[9];
Is4[2]:=buffer[8];
Is4[1]:=buffer[7];
Is4[0]:=buffer[6];
if Value1>-996 then begin
case Num1 of
 2:EditVal.Text:=Format('%7.2f',[Value1]);
 5:EditVal.Text:=Format('%7.2f',[Value1]);
 6:EditVal.Text:=GetStrDirWindDYg(6*Value1);
 else EditVal.Text:=Format('%7.1f',[Value1]);
 end;
end else begin
i:=round(Value1);
EditVal.Text:=IntToStr(i);
case i of
-997:begin
    Memo1.Clear;
    Memo1.Lines.Add(MsgPPUErr_Kod+'-997.');
    Memo1.Lines.Add(MsgPPU_997);
    end;
-998:begin
    Memo1.Clear;
    Memo1.Lines.Add(MsgPPUErr_Kod+'-998.');
    Memo1.Lines.Add(MsgPPU_998);
    end;
-999:begin
    Memo1.Clear;
    Memo1.Lines.Add(MsgPPUErr_Kod+'-999.');
    Memo1.Lines.Add(MsgPPU_999);
    end;
```

```
  end;
  end;//if Value1>-996
  end;//if (Num1<=6)...
end;

//**************************************************
procedure TTestPPUdlg.CommandBtnClick(Sender: TObject);
var
  Start1,Stop1:Longint;
  b2,Errors:DWord;
  Buffer:array of Byte;
  i:Integer;
  COM_Stat:tCOMStat;
begin
  try
  if not IsOpenCOMport then StartAMKport_PPU(COM_BaudRate,true);
  if not IsOpenCOMport then begin
    UpdateNoOtvet;
    Exit;
    end;
  PurgeComm(Handle_AMK,PURGE_RXCLEAR);
  PurgeComm(Handle_AMK,PURGE_TXCLEAR);
  Start1:=GetTickCount;
  Stop1:=Start1;
  while Stop1<Start1+50 do Stop1:=GetTickCount;
  RunTestPPU;
  Start1:=GetTickCount;
  Stop1:=Start1;
  while Stop1<Start1+500 do Stop1:=GetTickCount;
  ClearCommError(Handle_AMK,Errors,@COM_Stat);
  if COM_Stat.cbInQue<2 then UpdateNoOtvet
  else begin
  try
  SetLength(Buffer,COM_Stat.cbInQue);
  for i:=0 to High(buffer) do
    ReadFile(Handle_AMK,buffer[i],1,b2,@OverLap_AMK);
  DiagnozOtvet(buffer);
  finally
  buffer:=nil;
  end;
  end;
  except
  raise
  end;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit StyleGr;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, Spin;

type
  TStyleGraphDlg = class(TForm)
    MainPanel: TPanel;
    WindowBox: TGroupBox;
```

```
    BGColorLab: TLabel;
    CheckGrids: TCheckBox;
    WindowColorBox: TComboBox;
    LineBox: TGroupBox;
    LineColorLab: TLabel;
    LineStyleLab: TLabel;
    LineWidthLab: TLabel;
    LineTickEdit: TSpinEdit;
    LineStyleBox: TComboBox;
    MashtabBox: TGroupBox;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    RayTClab: TLabel;
    RayTClabDop: TLabel;
    RayVgLab: TLabel;
    RayVgEdit: TSpinEdit;
    RayTCEdit: TSpinEdit;
    RayFiEdit: TSpinEdit;
    RayFiLab: TLabel;
    RayFiLabDop: TLabel;
    RayVzEdit: TSpinEdit;
    RayVzLab: TLabel;
    RaySiEdit: TSpinEdit;
    RaySiLab: TLabel;
    RayDiEdit: TSpinEdit;
    RayDiLab: TLabel;
    RaytiEdit: TSpinEdit;
    RaytiLab: TLabel;
    Label3: TLabel;
    DefaultBtn: TBitBtn;
    DiagrBox: TGroupBox;
    Label5: TLabel;
    DigitsBox: TGroupBox;
    Label7: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    GridColorBox: TComboBox;
    LabelColorBox: TComboBox;
    DiagColorBox: TComboBox;
    ArrowColorBox: TComboBox;
    LineColorBox: TComboBox;
    EditColorBox: TComboBox;
    DigitColorBox: TComboBox;
    procedure FormCreate(Sender: TObject);
    procedure DefaultBtnClick(Sender: TObject);
  private
    { Private declarations }
    procedure Execute;
    procedure GetData;
  public
    { Public declarations }
  end;

var
  StyleGraphDlg: TStyleGraphDlg;

procedure GetStyleGraphDlg;
```

```
implementation
{$R *.dfm}
uses UnAMKini;

{****************************}
procedure GetStyleGraphDlg;
begin
  Screen.Cursor:=crHourGlass;
  try
  Application.CreateForm(TStyleGraphDlg, StyleGraphDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  StyleGraphDlg.Execute;
  finally
  StyleGraphDlg.Free;
  end;
end;

{****************************************}
procedure TStyleGraphDlg.Execute;
Begin
  try
  if ShowModal=IDOk then GetData;
  except
  raise
  end;
End;

{****************************************}
procedure TStyleGraphDlg.FormCreate(Sender: TObject);
begin
  CheckGrids.Checked:=IsOutGrid;
  WindowColorBox.ItemIndex:=WindowColor;
  GridColorBox.ItemIndex:=GridColor;
  LabelColorBox.ItemIndex:=LabelColor;
  LineColorBox.ItemIndex:=LineColor;
  DiagColorBox.ItemIndex:=DiagColor;
  ArrowColorBox.ItemIndex:=ArrowColor;
  EditColorBox.ItemIndex:=EditColor;
  DigitColorBox.ItemIndex:=DigitColor;
  LineStyleBox.ItemIndex:=LineStyle;
  LineTickEdit.Value:=LineTick;
  RayTCEdit.Value:=RayTC;
  RayVgEdit.Value:=RayVg;
  RayFiEdit.Value:=RayFi;
  RayVzEdit.Value:=RayVz;
  RaySiEdit.Value:=RaySi;
  RayDiEdit.Value:=RayDi;
  RaytiEdit.Value:=Rayti;
end;

{****************************************}
procedure TStyleGraphDlg.GetData;
begin
  IsOutGrid:=CheckGrids.Checked;
  WindowColor:=WindowColorBox.ItemIndex;
  GridColor:=GridColorBox.ItemIndex;
```

```
  LabelColor:=LabelColorBox.ItemIndex;
  LineColor:=LineColorBox.ItemIndex;
  DiagColor:=DiagColorBox.ItemIndex;
  ArrowColor:=ArrowColorBox.ItemIndex;
  EditColor:=EditColorBox.ItemIndex;
  DigitColor:=DigitColorBox.ItemIndex;
  LineStyle:=LineStyleBox.ItemIndex;
  LineTick:=LineTickEdit.Value;
  RayTC:=RayTCEdit.Value;
  RayVg:=RayVgEdit.Value;
  RayFi:=RayFiEdit.Value;
  RayVz:=RayVzEdit.Value;
  RaySi:=RaySiEdit.Value;
  RayDi:=RayDiEdit.Value;
  Rayti:=RaytiEdit.Value;
end;

{*************************************************}
procedure TStyleGraphDlg.DefaultBtnClick(Sender: TObject);
begin
  CheckGrids.Checked:=true;
  WindowColorBox.ItemIndex:=17;
  GridColorBox.ItemIndex:=7;
  LabelColorBox.ItemIndex:=0;
  LineColorBox.ItemIndex:=9;
  DiagColorBox.ItemIndex:=8;
  ArrowColorBox.ItemIndex:=4;
  EditColorBox.ItemIndex:=4;
  DigitColorBox.ItemIndex:=14;
  LineStyleBox.ItemIndex:=0;
  LineTickEdit.Value:=1;
  RayTCEdit.Value:=4;
  RayVgEdit.Value:=10;
  RayFiEdit.Value:=360;
  RayVzEdit.Value:=2;
  RaySiEdit.Value:=4;
  RayDiEdit.Value:=500;
  RaytiEdit.Value:=500;
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit Phone;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Buttons, StdCtrls, ExtCtrls, ComCtrls, Main;

const
  MsgModems_11='  Установлена';
  MsgModems_12=' Не установлена';

type
  TPhoneForm = class(TForm)
    ModemBox: TGroupBox;
    PhoneBox: TGroupBox;
    EditCommandModem: TEdit;
```

```
    LabAnswer: TLabel;
    MemoModem: TMemo;
    EditNumberTeleFon: TEdit;
    Label3: TLabel;
    HelpBtn: TBitBtn;
    OkBtn: TBitBtn;
    Label4: TLabel;
    UpTrubBtn: TBitBtn;
    CloseBtn: TBitBtn;
    StatusBar1: TStatusBar;
    OptionBtn: TBitBtn;
    Label5: TLabel;
    EditConnect: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure CloseBtnClick(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure OkBtnClick(Sender: TObject);
    procedure UpTrubBtnClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure OptionBtnClick(Sender: TObject);
  private
    { Private declarations }
    StartTimeRing:TDateTime;
    FlagMsg_9:Boolean;
    procedure MyDelay(Tay:Word);
    procedure ReadAnswerModem(Handle1:THandle; var OverLap1:tOverLapped; var Answer:TStringList);
    procedure DiagnozAnswerModem(const Answer:TStringList);
    procedure MakeRing;
    procedure MakeConnect;
    procedure InputControl;
  public
    { Public declarations }
    procedure ViewAnswerModem;
  end;

var
  PhoneForm: TPhoneForm;

procedure OpenPhoneForm;

implementation
{$R *.DFM}
uses UnAMKini, ApllDlgs, UnAMKsys, UnAMKport, SetPhone;

const
 MsgModems_0='Последняя команда выполнена успешно'; {OK}
 MsgModems_1='Связь установлена'; {CONNECT}
 MsgModems_2='Получен сигнал "звонок"'; {RING}
 MsgModems_3='Нет несущей удаленного модема'; {NO CARRIER}
 MsgModems_4='Ошибка в командной строке'; {ERROR}
 MsgModems_5='Нет длинного гудка в линии'; {NO DIALTONE}
 MsgModems_6='Вызываемый абонент занят'; {BUSY}
 MsgModems_7='Абонент не отвечает'; {NO ANSWER}
 MsgModems_8='Текст команды модему';
 MsgModems_9='Неопознанное сообщение модема';
 MsgModems_10='Обнаружена связь с устройством';

 MsgErrModem0='Нет отклика модема';
```

```
 MsgErrModem01=' через порт ';
 MsgErrModem0d='Проверьте в Windows подключение модема к компьютеру и/или настройки программы.';
 MsgErrModem3='Команда для модема не задана.';
 MsgClosePhone1=' При закрытии данного окна выполняемые измерения будут прерваны.';
 MsgClosePhone2=' Вы действительно хотите этого ?';

procedure SendCommandModem(Handle1:THandle; const Command:ShortString;
                    var OverLap1:tOverLapped);far; external 'AMKport.DLL';

{**********************************}
procedure OpenPhoneForm;
begin
  StartAMKport_UGI;
  if not IsOpenCOMport then Exit;
  Close_AMKport(Handle_AMK,IsOpenCOMport);
  try
  try
  Screen.Cursor:=crHourGlass;
  Application.CreateForm(TPhoneForm, PhoneForm);
  MainForm.SetupConnect.Enabled:=false;
  finally
  Screen.Cursor:=crDefault;
  end;
  except
  PhoneForm.Free;
  if IsOpenCOMport then Close_AMKport(Handle_AMK,IsOpenCOMport);
  end;
end;

{**************************************************}
procedure TPhoneForm.FormCreate(Sender: TObject);
begin
  Height:=270;
  Width:=320;
  Left:=Screen.Width-Width;
  Top:=Screen.Height-Height-32;
  EditNumberTeleFon.Text:=NumberTeleFon;
  EditConnect.Color:=ClNavy;
  EditConnect.Text:=MsgModems_12;
  IsConnectPhone:=false;
  IsRingPhone:=false;
  isPhoneForm:=true;
  FlagMsg_9:=false;
  Show;
  BringToFront;
  StartAMKport_UGI;
  if IsSbojPhone then MainForm.StopBtn.Enabled:=true;
  InputControl;
end;

{*********************************************}
procedure TPhoneForm.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
var
  W:Integer;
begin
  CanClose:=true;
  if IsSbojPhone then Exit;
  if not MainForm.BreakReadDataUGI then Begin
  W:=GetMyConfirm2Dlg(MsgClosePhone1+MsgClosePhone2);
```

```
  if W = IdYes then begin
    MainForm.StopBtnClick(Sender);
    CanClose:=true;
    end else CanClose:=false;
  End;
end;

{*************************************************}
procedure TPhoneForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if IsSbojPhone then NumberTeleFon:=EditNumberTeleFon.Text;
  with MainForm do begin
  IsConnectPhone:=false;
  CheckTimeRestModems:=Now+EncodeTime(0,0,DelayAutRingModem,0);
  SetupConnect.Enabled:=true;
  CloseDevice;
  if IsRunAutomatic  then begin
    StopBtn.Enabled:=true;
    StartBtn.Enabled:=false;
    end;
  Close_AMKport(Handle_AMK,IsOpenCOMport);
  IsPhoneForm:=false;
  end;
  Action:=caFree;
end;

{*************************************************}
procedure TPhoneForm.CloseBtnClick(Sender: TObject);
begin
  Close;
end;

{*************************}
procedure TPhoneForm.MakeConnect;
begin
  IsRingPhone:=false;
  UpTrubBtn.Enabled:=false;
  OkBtn.Visible:=false;
  OptionBtn.Enabled:=false;
  EditConnect.Color:=ClGreen;
  EditConnect.Text:=MsgModems_11;
  MemoModem.Visible:=false;
  LabAnswer.Visible:=false;
  StatusBar1.SimpleText :=MsgModems_1;
  IsConnectPhone:=true;
  MainForm.CheckTimeInQue:=Now;
  Invalidate;
  update;
  if not IsSbojPhone then begin
  with MainForm do Begin
    OpenDevice;
    if IsRunAutomatic then begin
      StopBtn.Enabled:=true;
      StartBtn.Enabled:=false;
      end;
    End;
  end else MainForm.ZapretReadCOMbuf:=false;
  IsSbojPhone:=false;
end;
```

```
{*********************************}
procedure TPhoneForm.DiagnozAnswerModem(const Answer:TStringList);
var
  St,st2:String;
  i:Integer;
begin
  try
  if Answer.Count=0 then Exit;
  for i:=0 to Answer.Count-1 do begin
  St:=Answer.Strings[i];
  MemoModem.Lines.Add(St);
  if (St<>'') or (St<>' ') or (St<>'  ') or (St<>'   ') then Begin
  if Length(st)>=7 then st2:=Copy(St,1,7);
  if st2='CONNECT' then begin
    StatusBar1.SimpleText :=MsgModems_1;
    MakeConnect;
    Exit;
    end else begin
  if St='OK' then begin
    StatusBar1.SimpleText:=MsgModems_0;
    end else
  if St='RING' then begin
    StatusBar1.SimpleText:=MsgModems_2;
    if IsRingPhone then MakeRing;
    end else
  if St='NO CARRIER' then begin
    StatusBar1.SimpleText:=MsgModems_3;
    if IsRingPhone then MakeRing;
    end else
  if St='ERROR' then begin
    StatusBar1.SimpleText:=MsgModems_4;
    if IsRingPhone then MakeRing;
    end else
  if (St='NO DIAL TONE') or (St='NO DIALTONE') then Begin
    StatusBar1.SimpleText:=MsgModems_5;
    if IsRingPhone then MakeRing;
    end else
  if St='BUSY' then begin
    StatusBar1.SimpleText:=MsgModems_6;
    if IsRingPhone then MakeRing;
    end else
  if St='NO ANSWER' then Begin
    StatusBar1.SimpleText:=MsgModems_7;
    if IsRingPhone then MakeRing;
    end else
  if st=EditCommandModem.Text then begin
    StatusBar1.SimpleText:=MsgModems_8;
    end else begin
    StatusBar1.SimpleText:=MsgModems_9;
    FlagMsg_9:=true;
  end; end;
  End; {if (St<>'')}
  Invalidate;
  update;
  end; {i}
  except
  raise
  end;
end;
```

```
{*******************************}
procedure TPhoneForm.ReadAnswerModem(Handle1:THandle; var OverLap1:tOverLapped; var Answer:TStringList);
var
  ReadBytes,Errors,k:DWord;
  COM_Stat:tCOMStat;
  Buff:array [0..65000] of Byte;
  St:String;
  b1:Byte;
begin
  try
  ClearCommError(Handle1,Errors,@COM_Stat);
  if COM_Stat.cbInQue>0 then Begin
  ReadFile(Handle1,Buff,COM_Stat.cbInQue,ReadBytes,@OverLap1);
  GetOverlappedResult(Handle1,OverLap1,ReadBytes,true);
  if ReadBytes>0 then Begin
  St:='';
  for k:=0 to ReadBytes-1 do begin
  b1:=Buff[k];
  case b1 of
    13:; {<Enter>}
    43:; {<Esc>}
     8:; {<BS>}
    10:begin {<LF> }
        if St<>'' then Answer.Add(St);
        St:='';
      end;
    else St:=St+Chr(b1);
  end; //case
  end; // for k
  if St<>'' then Answer.Add(St);
  End;
  End;
  except
  raise
  end;
end;

{************************}
procedure TPhoneForm.ViewAnswerModem;
var
  Answer:TStringList;
  Errors:DWord;
  COM_Stat:tCOMStat;
  Buf1:array[0..CountByteData_UGI] of Byte;
begin
  try
  Answer:=TStringList.Create;
  if IsConnectPhone then begin
    StatusBar1.SimpleText:=MsgModems_10;
    Exit;
    end;
  if IsRingPhone and
   (Now<StartTimeRing+EncodeTime(0,0,DelayConnectModems,0)) then Exit;
  StartTimeRing:=Now;
  ClearCommError(Handle_AMK,Errors,@COM_Stat);
if COM_Stat.cbInQue>0 then Begin
  if not FlagMsg_9 then begin
    ReadAnswerModem(Handle_AMK,OverLap_AMK,Answer);
    if Answer.Count>0 then DiagnozAnswerModem(Answer);
    end;
```

```
    if IsRingPhone and FlagMsg_9 then Begin
      if COM_Stat.cbInQue div (CountByteData_UGI+3)>1 then
      if ReadUGI_Data(Handle_AMK,CountByteData_UGI+1,Buf1,OverLap_AMK) then
      MakeConnect;
      if IsSbojPhone then MakeRing;
      FlagMsg_9:=false;
      End; //if IsRingPhone and FlagMsg_9
  End;// if COM_Stat.cbInQue>0
    finally
    Answer.Free;
    end;
  end;

  {***********************************************}
  procedure TPhoneForm.InputControl;
  var
    Answer:TStringList;
    St,Msg:String;
  begin
    try
    Answer:=TStringList.Create;
    StartAMKport_UGI;
    if IsOpenCOMport then begin
    EditCommandModem.Text:='AT E1';// Q V1 X4 S6=3 S10=255';
    SendCommandModem(Handle_AMK,EditCommandModem.Text,OverLap_AMK);
    MyDelay(1);
    ReadAnswerModem(Handle_AMK,OverLap_AMK,Answer);
    if Answer.Count=0 then begin
    St:='COM'+IntToStr(COM_Number+1);
    Msg:=MsgErrModem0+MsgErrModem01+st+'. '+MsgErrModem0d;
    GetMyErrorDlg(Msg);
    StatusBar1.SimpleText :=MsgErrModem0;
    end else DiagnozAnswerModem(Answer);
    end;
    finally
    Answer.Free;
    end;
  end;

  {*****************************************}
  procedure TPhoneForm.MakeRing;
  var
    st:String[1];
  begin
    if not isOpenComPort then StartAMKport_UGI;
    if MethodRingModem=0 then St:='P' else st:='T';
    EditCommandModem.Text:='AT D'+St+EditNumberTeleFon.Text;
    SendCommandModem(Handle_AMK,EditCommandModem.Text,OverLap_AMK);
    StartTimeRing:=Now;
    IsRingPhone:=true;
  end;

  {*****************************************}
  procedure TPhoneForm.UpTrubBtnClick(Sender: TObject);
  begin
    MakeRing;
  end;

  {*****************************************}
  procedure TPhoneForm.OkBtnClick(Sender: TObject);
```

```
begin
  SendCommandModem(Handle_AMK,EditCommandModem.Text,OverLap_AMK);
end;

{*****************************************}
procedure TPhoneForm.OptionBtnClick(Sender: TObject);
begin
  try
  GetOptionPhone(DelayConnectModems,DelayDataModem,DelayAutRingModem,
              BlockDataModem,MethodRingModem);
  except
  end;
end;

{*********************************************}
procedure TPhoneForm.MyDelay(Tay:Word);
var
  Start1,Stop1:Cardinal;
begin
  Start1:=GetTickCount div 1000;
  Stop1:=Start1;
  while Stop1 < (Start1+Tay) do
  Stop1:=GetTickCount div 1000;
end;

{*********************************************}
procedure TPhoneForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then CloseBtnClick(Sender);
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit SetPhone;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, Spin;

type
  TOptionPhoneDlg = class(TForm)
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    MethodRG: TRadioGroup;
    EditDelayConnect: TSpinEdit;
    EditDelayData: TSpinEdit;
    EditAutRing: TSpinEdit;
    EditBlockData: TSpinEdit;
  private
    { Private declarations }
    procedure SetData(Connect1,Data1,AutRing1,Block1:Integer;Method1:Byte);
```

```
    procedure GetData(var Connect1,Data1,AutRing1,Block1:Integer;var Method1:Byte);
    procedure Execute(var Connect1,Data1,AutRing1,Block1:Integer;var Method1:Byte);
  public
    { Public declarations }
  end;

var
  OptionPhoneDlg: TOptionPhoneDlg;
procedure GetOptionPhone(var Connect1,Data1,AutRing1,Block1:Integer;
                    var Method1:Byte);

implementation
{$R *.dfm}
{**********************************************}
procedure GetOptionPhone(var Connect1,Data1,AutRing1,Block1:Integer; var Method1:Byte);
begin
  Screen.Cursor:=crHourGlass;
  try
  Application.CreateForm(TOptionPhoneDlg, OptionPhoneDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  OptionPhoneDlg.Execute(Connect1,Data1,AutRing1,Block1,Method1);
  finally
  OptionPhoneDlg.Free;
  end;
end;

{****************************}
procedure TOptionPhoneDlg.SetData(Connect1,Data1,AutRing1,Block1:Integer;Method1:Byte);
begin
  MethodRG.ItemIndex:=Method1;
  EditDelayConnect.Value:=Connect1;
  EditDelayData.Value:=Data1;
  EditAutRing.Value:=AutRing1;
  EditBlockData.Value:=Block1;
end;

{*****************************************}
procedure TOptionPhoneDlg.GetData(var Connect1,Data1,AutRing1,Block1:Integer;
                         var Method1:Byte);
begin
  Method1:=MethodRG.ItemIndex;
  Connect1:=EditDelayConnect.Value;
  Data1:=EditDelayData.Value;
  AutRing1:=EditAutRing.Value;
  Block1:=EditBlockData.Value;
end;

{*****************************************}
procedure TOptionPhoneDlg.Execute(var Connect1,Data1,AutRing1,Block1:Integer;
                         var Method1:Byte);
Begin
  SetData(Connect1,Data1,AutRing1,Block1,Method1);
  if ShowModal=idOk then
    GetData(Connect1,Data1,AutRing1,Block1,Method1);
End;
end.
```

```
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit PrnText;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, Spin, Dialogs, Printers;

type
  TPrintTextDlg = class(TForm)
    Panel1: TPanel;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    PagesGr: TRadioGroup;
    NumStartEdit: TSpinEdit;
    NumStopEdit: TSpinEdit;
    PrinterBox: TGroupBox;
    SetupBtn: TBitBtn;
    PrinterSetupDialog1: TPrinterSetupDialog;
    CopyBox: TGroupBox;
    CopyEdit: TSpinEdit;
    LabCopy: TLabel;
    FontDialog1: TFontDialog;
    FontBox: TGroupBox;
    FontBtn: TBitBtn;
    procedure SetupBtnClick(Sender: TObject);
    procedure FontBtnClick(Sender: TObject);
    procedure PagesGrClick(Sender: TObject);
  private
    { Private declarations }
    MyFont:TFont;
    function  Execute(var PrnPages,P1,P2,Copy:Integer; var F1:TFont):Boolean;
    procedure SetData(const F1:TFont);
    procedure GetData(var PrnPages,P1,P2,Copy:Integer; var F1:TFont);
    function ValidData:Boolean;
  public
    { Public declarations }
  end;

var
  PrintTextDlg: TPrintTextDlg;

function GetPrintTextDlg(var PrnPages,P1,P2,Copy:Integer; var F1:TFont):WordBool;

implementation
{$R *.dfm}

{*****************************************************************************}
function GetPrintTextDlg(var PrnPages,P1,P2,Copy:Integer; var F1:TFont):WordBool;
begin
  Screen.Cursor:=crHourGlass;
  try
  Application.CreateForm(TPrintTextDlg, PrintTextDlg);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  Result:=PrintTextDlg.Execute(PrnPages,P1,P2,Copy,F1);
  finally
  PrintTextDlg.Free;
```

```
  end;
end;

{*****************************************************************************}
function TPrintTextDlg.Execute(var PrnPages,P1,P2,Copy:Integer; var F1:TFont):Boolean;
Label 10;
Begin
  Result:=false;
  SetData(F1);
10:if ShowModal=mrOk Then
  Begin
    Result:=ValidData;
    if not Result then goto 10;
    GetData(PrnPages,P1,P2,Copy,F1);
  End;
end;

{*****************************************************************************}
procedure TPrintTextDlg.SetData(const F1:TFont);
begin
  PagesGr.ItemIndex:=0;
  NumStartEdit.Value:=1;
  NumStopEdit.Value:=1;
  CopyEdit.Value:=1;
  MyFont:=F1;
  NumStartEdit.Enabled:=false;
  NumStopEdit.Enabled:=false;
end;

{*****************************************************************************}
procedure TPrintTextDlg.PagesGrClick(Sender: TObject);
begin
  if PagesGr.ItemIndex=1 then begin
    NumStartEdit.Enabled:=true;
    NumStopEdit.Enabled:=true;
    end;
end;

{*****************************************************************************}
procedure TPrintTextDlg.GetData(var PrnPages,P1,P2,Copy:Integer; var F1:TFont);
begin
  PrnPages:=PagesGr.ItemIndex;
  if PrnPages=1 then begin
    P1:=NumStartEdit.Value;
    P2:=NumStopEdit.Value;
    end;
  Copy:=CopyEdit.Value;
  F1:=MyFont;
end;

{*****************************************************************************}
function TPrintTextDlg.ValidData:Boolean;
begin
  try
  Result:=true;
  if PagesGr.ItemIndex=1 then
  if NumStopEdit.Value < NumStartEdit.Value then Result:=false;
  except
  raise;
  end;
```

```
end;

{*****************************************************************************}
procedure TPrintTextDlg.SetupBtnClick(Sender: TObject);
begin
  PrinterSetupDialog1.Execute;
end;

{*****************************************************************************}
procedure TPrintTextDlg.FontBtnClick(Sender: TObject);
begin
  FontDialog1.Font:=MyFont;
  if FontDialog1.Execute then
    MyFont:=FontDialog1.Font;
end;

END.

{888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit Splash;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, jpeg;

type
  TSplashForm = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel4: TPanel;
    Image1: TImage;
    LabRegNumber: TLabel;
    LabCopyr: TLabel;
    Label3: TLabel;
    Label16: TLabel;
    Label2: TLabel;
    Label10: TLabel;
    Label6: TLabel;
    Label5: TLabel;
    Label4: TLabel;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  SplashForm: TSplashForm;

IMPLEMENTATION
{$R *.DFM}
uses UnAMKsys;

procedure TSplashForm.FormCreate(Sender: TObject);
begin
  LabCopyr.Caption:=st_Copyr;
  LabRegNumber.Caption:='Зав.№ '+RegNumber;
```

```
end;

END.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

UNIT About;
INTERFACE
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, jpeg;

type
  TAboutBox = class(TForm)
    GroupBox1: TGroupBox;
    Image2: TImage;
    LabVersion: TLabel;
    LabCopyr: TLabel;
    Label1: TLabel;
    Label6: TLabel;
    Label10: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label3: TLabel;
    LabBriefName: TLabel;
    OKBtn: TBitBtn;
    Label2: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure OKBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

procedure GetAboutDlg;

IMPLEMENTATION
{$R *.DFM}
uses UnAMKsys;

{*************************************************}
procedure GetAboutDlg;
begin
  try
  Screen.Cursor:=crHourGlass;
  Application.CreateForm(TAboutBox, AboutBox);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  AboutBox.ShowModal;
  finally
  AboutBox.Free;
  end;
end;
```

```
{*****************************************}
procedure TAboutBox.FormCreate(Sender: TObject);
begin
  LabBriefName.Caption:=st_BriefNamePr;
  LabCopyr.Caption:=st_Copyr;
end;

procedure TAboutBox.OKBtnClick(Sender: TObject);
begin
  Close;
end;

End.

{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}
{88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888}

unit AboutDevice;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;

type
  TDeviceBox = class(TForm)
    OKBtn: TBitBtn;
    GroupBox1: TGroupBox;
    Image2: TImage;
    Label24: TLabel;
    Label3: TLabel;
    Label12: TLabel;
    Label7: TLabel;
    Label5: TLabel;
    LabRegNumber: TLabel;
    Label9: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label8: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    Label10: TLabel;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  DeviceBox: TDeviceBox;

procedure GetDeviceDlg;

implementation
{$R *.dfm}
uses UnAMKsys;

{*************************************************}
procedure GetDeviceDlg;
begin
  try
```

```
  Screen.Cursor:=crHourGlass;
  Application.CreateForm(TDeviceBox, DeviceBox);
  finally
  Screen.Cursor:=crDefault;
  end;
  try
  DeviceBox.ShowModal;
  finally
  DeviceBox.Free;
  end;
end;

{**************************************************}
procedure TDeviceBox.FormCreate(Sender: TObject);
begin
  LabRegNumber.Caption:='Зав.№ '+RegNumber;
end;

End.
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводит. докум. и дата	Подпись	Дата
	Изменен-ных	Заменен-ных	Новых	Изъятых					

Подп. и дата	Инв. № дубл.	Взам. инв.№	Подп. и дата	Инв. № подл.