ВОРОНЕЖСКИЙ ИНСТИТУТ МВД РОССИИ

Кафедра физики и радиоэлектроники

С.В. Железный
С.С. Никулин
М.М. Жуков
В.П. Удалов
О.В. Четкин

АВТОМАТИЗАЦИЯ ПРОЦЕССА УЧЕТА СРЕДСТВ ИЗМЕРЕНИЙ, НАХОДЯЩИХСЯ НА БАЛАНСЕ ОВД

Методические рекомендации

Воронеж – 2022

ББК 30.10

Рассмотрены и одобрены на заседании кафедры физики и радиоэлектроники. Протокол № 9 от 11 мая 2022 г.

Рассмотрены и одобрены на заседании методического совета. Протокол № 10 от 16 мая 2022 г.

Обсуждены и рекомендованы на редакционно-издательском совете. Протокол № 5 от 24 мая 2022 г.

Авторский коллектив: Железный С.В., Никулин С.С., Жуков М.М., Удалов В.П., Четкин О.В.

Автоматизация процесса учета средств измерений, находящихся на балансе ОВД: методические рекомендации [Электронный ресурс] / С.В. Железный [и др.]. – Электр. дан. и прогр. – Воронеж : Воронежский институт МВД России, 2022. – 1 электр. опт. диск (CD-ROM) : 12 см. – Систем. требования: процессор Intel с частотой не менее 1,3 ГГц ; ОЗУ 512 Мб ; операц. система семейства Windows ; CD-ROM дисковод.

ISBN 978-5-88591-489-5

В методических рекомендациях рассматриваются вопросы, связанные с установкой, описанием работы и практическим применением компьютерной программы формирования планирующих и отчетных документов для поверки средств измерений, находящихся на балансе ОВД. Приведен список литературы, необходимой для организации работы главных специалистов-метрологов территориальных органов МВД России.

ББК 30.10

СОДЕРЖАНИЕ

Назначение программы

Разработанная на языке программирования DELPHI компьютерная программа предназначена для ведения автоматизированного учета состояния средств измерений, подготовки технической документации по метрологическому обеспечению на персональном компьютере и выполняет следующие функции:

1. Формирование базы данных по учету наличия, места нахождения и состояния приборов.

2. Контроль и планирование различных видов метрологического надзора и контроля.

3. Контроль над проведением метрологического обслуживания средств измерений.

4. Ведение справочников: наименование, тип, марка, категории и виды, заводские и эксплуатационные данные средств измерений.

5. Анализ состояния средств измерений, находящихся в эксплуатации.

6. Формирование и ведение эксплуатационных паспортов средств измерений.

7. Заполнение учетной карточки средств измерений.

8. Автоматическое отслеживание не поверенных средств измерений.

9. Ведение данных по стоимости поверок, контроля.

10. Редактирование данных о приборе.

11. Осуществление поиска необходимого оборудования, находящегося в базе программы учета и анализа по маркерным запросам. В качестве маркеров могут выступать:

- оставшееся время до очередной поверки;
- наличие статуса поверен или не поверен;
- наличие статуса калиброван или не калиброван.

Установка программы

Для работы с компьютерной программой необходимо с установочного диска произвести копирование папки «АРМ Метролога» на жесткий диск персонального компьютера. Запуск программы осуществляется активацией файла metr.exe из папки «distr», находящейся в папке «АРМ Метролога» и ввести пароль доступа.

Структура программы

Программа состоит из следующих разделов:

1. Средства измерений:

- наименование средства измерений,
- тип средства измерений,
- вид измерений,
- подразделение, в котором находится средство измерений,
- заводской номер,
- дата изготовления,
- инвентарный номер,
- добавление данных:
    - эксплуатационные данные,
    - заводские данные,
- изменить данные,
- удалить данные.

2. Поверка:

- наименование средства измерений,
- тип средства измерений,
- подразделение, в котором находится средство измерений,
- дата поверки,
- дата следующей поверки,

- внести данные о поверке.

3. Калибровка:

- наименование средства измерений,

- тип средства измерений,

- подразделение, в котором находится средство измерений,

- дата калибровки,

- дата следующей калибровки,

- внести данные о калибровке.

4. Ремонт:

- наименование средства измерений,

- тип средства измерений,

- подразделение, в котором находится средство измерений,

- дата ремонта,

- причина ремонта,

- внести данные о ремонте,

- изменить данные.

5. Печать план-графика.

6. Печать книги учета.

Интерфейс пользователя

Рабочее окно программы имеет вид – рис. 1 – и состоит из шести рабочих окон:

- Средства измерения,
- Поверка,
- Калибровка,
- Ремонт,
- Печать план-графика,
- Печать книги учета.

Работа с программой

Ввод данных о средстве измерений.

Для ввода данных о средстве измерений осуществляется путем заполнения необходимых полей таблиц при нажатии кнопки «Добавить» в рабочем окне «Средства измерения» (рисунок 1). В результате перед пользователем откроется окно (рисунок 2).

Редактирование данных о приборе.

Редактирование данных о приборе осуществляется нажатием кнопки «Изменить» в окне «Средства измерения» (рисунок 1).

Печать план-графика.

Печать план-графика осуществляется путем нажатия клавиши «Печать план-графика» (рисунок 1).

Печать книги учета.

Печать книги учета осуществляется путем нажатия клавиши «Печать книги учета» (рисунок 1).

Ввод данных о поверке прибора.

Ввод данных о поверке прибора осуществляется путем заполнения необходимых полей таблиц в рабочем окне «Поверка» (рисунок 3), при нажатии клавиши «Внести данные о поверке». Окно ввода информации о поверке устройства имеет вид – рисунок 4.

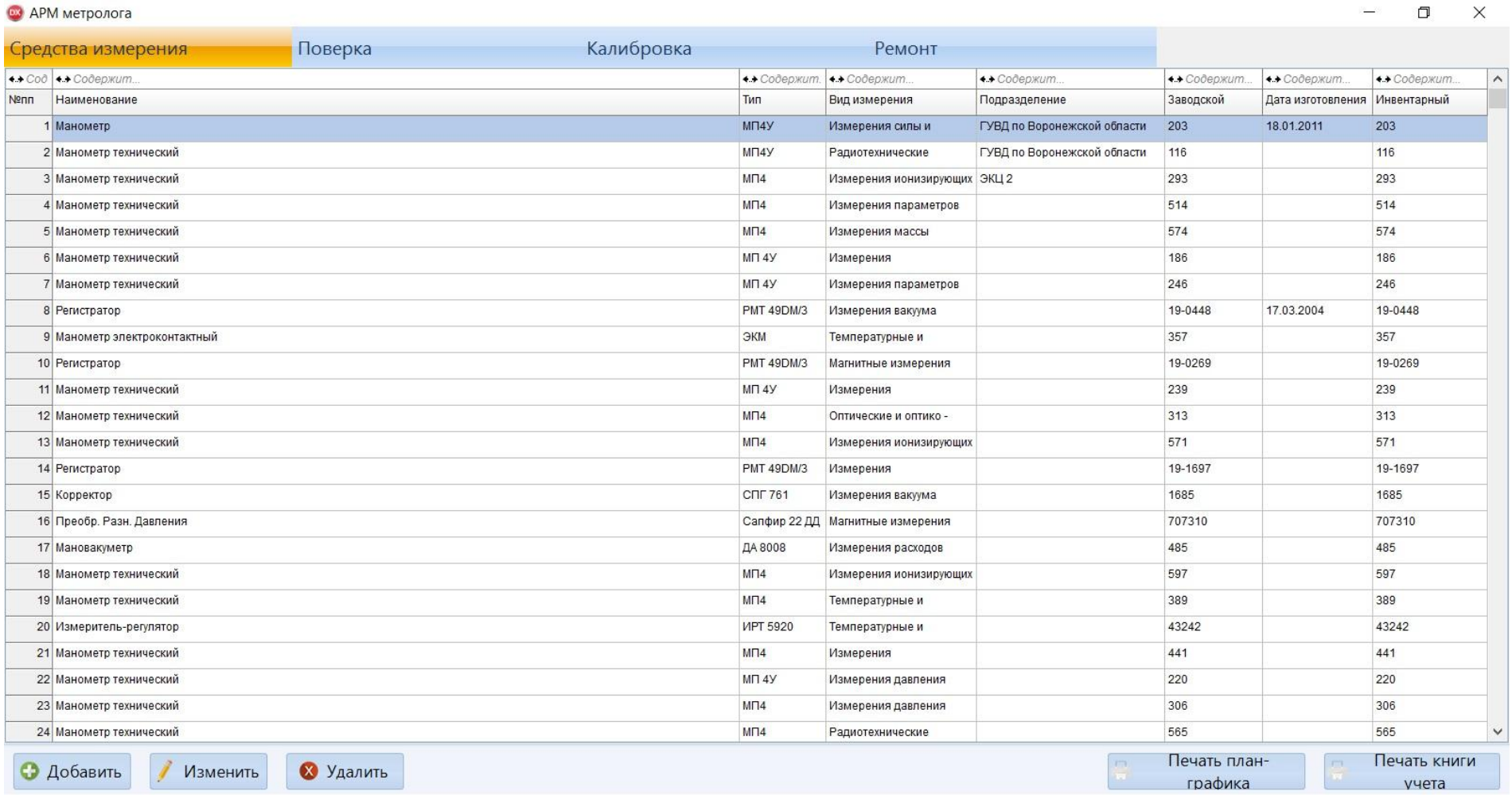

Рис. 1. Рабочее окно программы

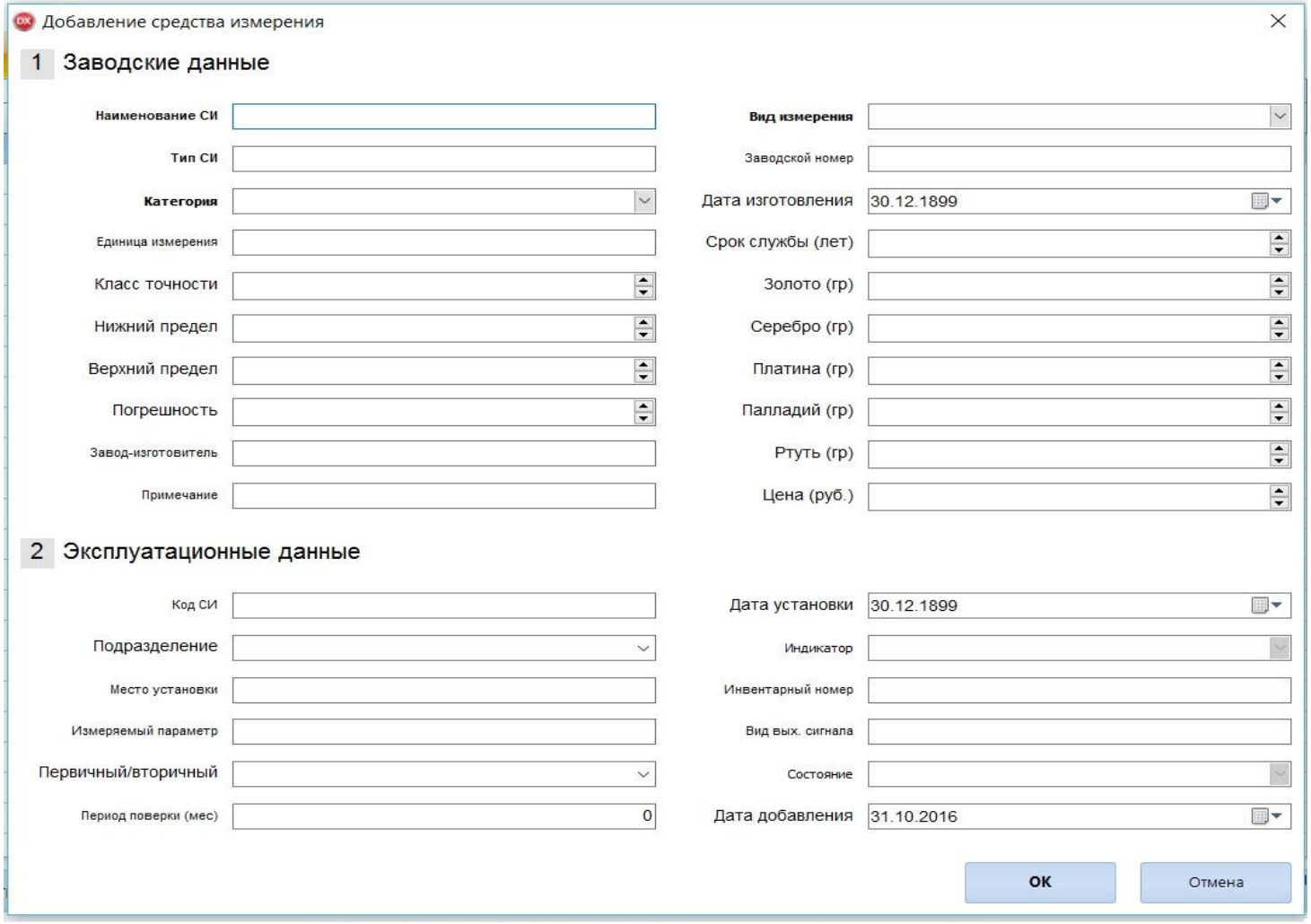

Рис. 2. Добавление средства измерений.

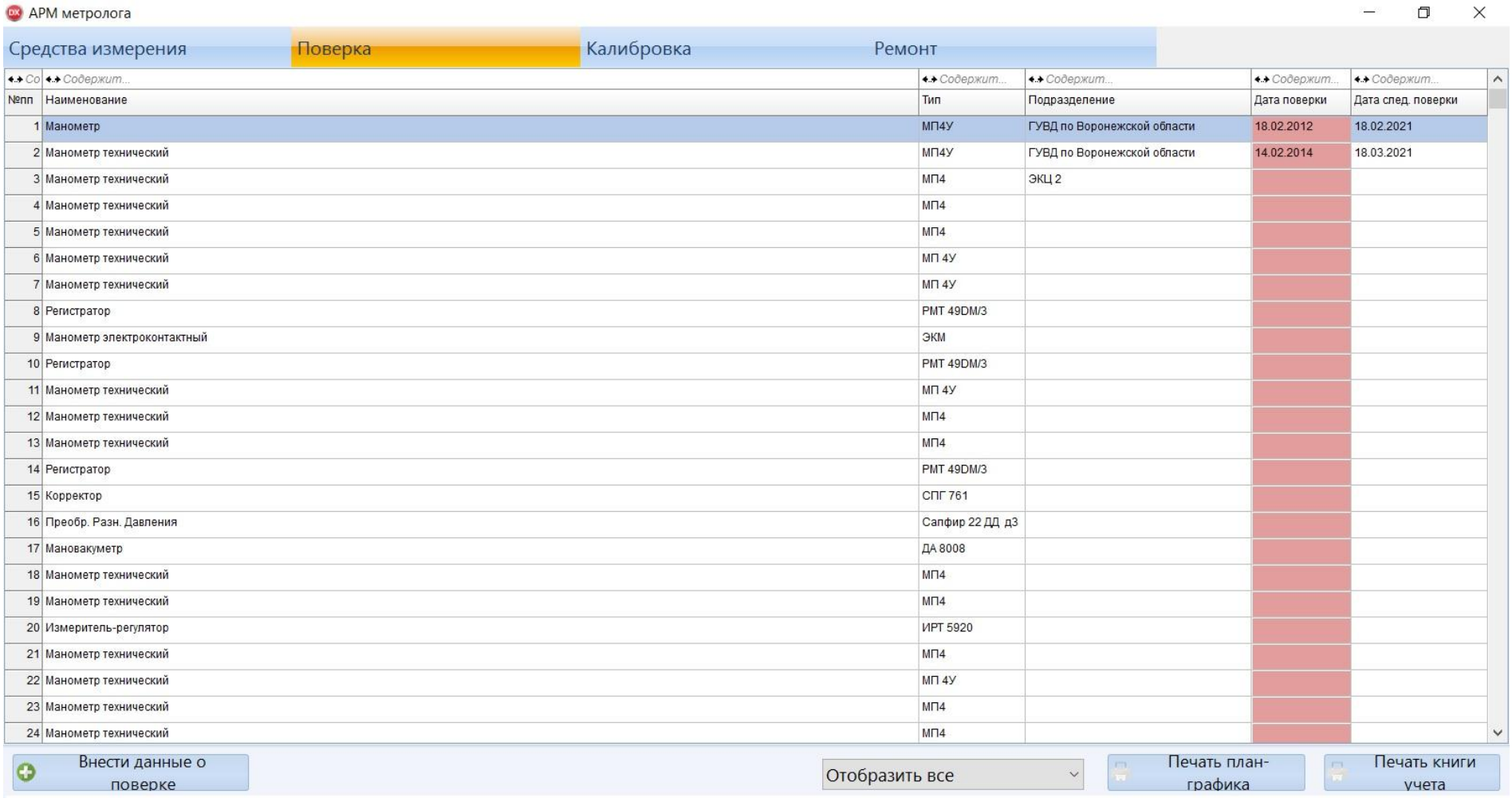

Рис. 3. Рабочее окно "Поверка".

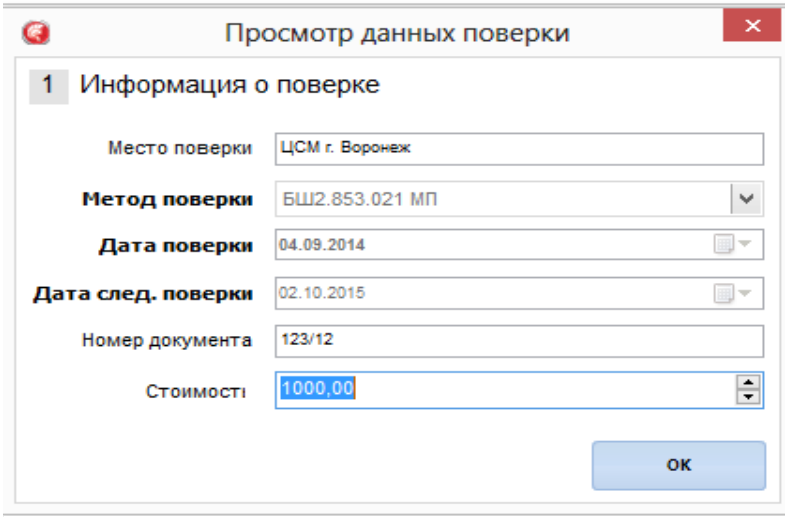

Рис. 4. Окно ввода информации о поверке

В рабочем окне «Поверка» возможно осуществить отображение приборов по следующим параметрам:

- отобразить все,

- отобразить за 3 месяца до окончания срока,

- только просроченные,

- не поверенные.

Для выбора соответствующего маркера необходимо в правом нижнем углу программы в режиме «Поверка» выбрать соответствующий раздел (рисунок 5).

Ввод данных о калибровке прибора.

Ввод данных о калибровке прибора осуществляется путем заполнения необходимых полей таблиц в рабочем окне «Калибровка» (рисунок 6).

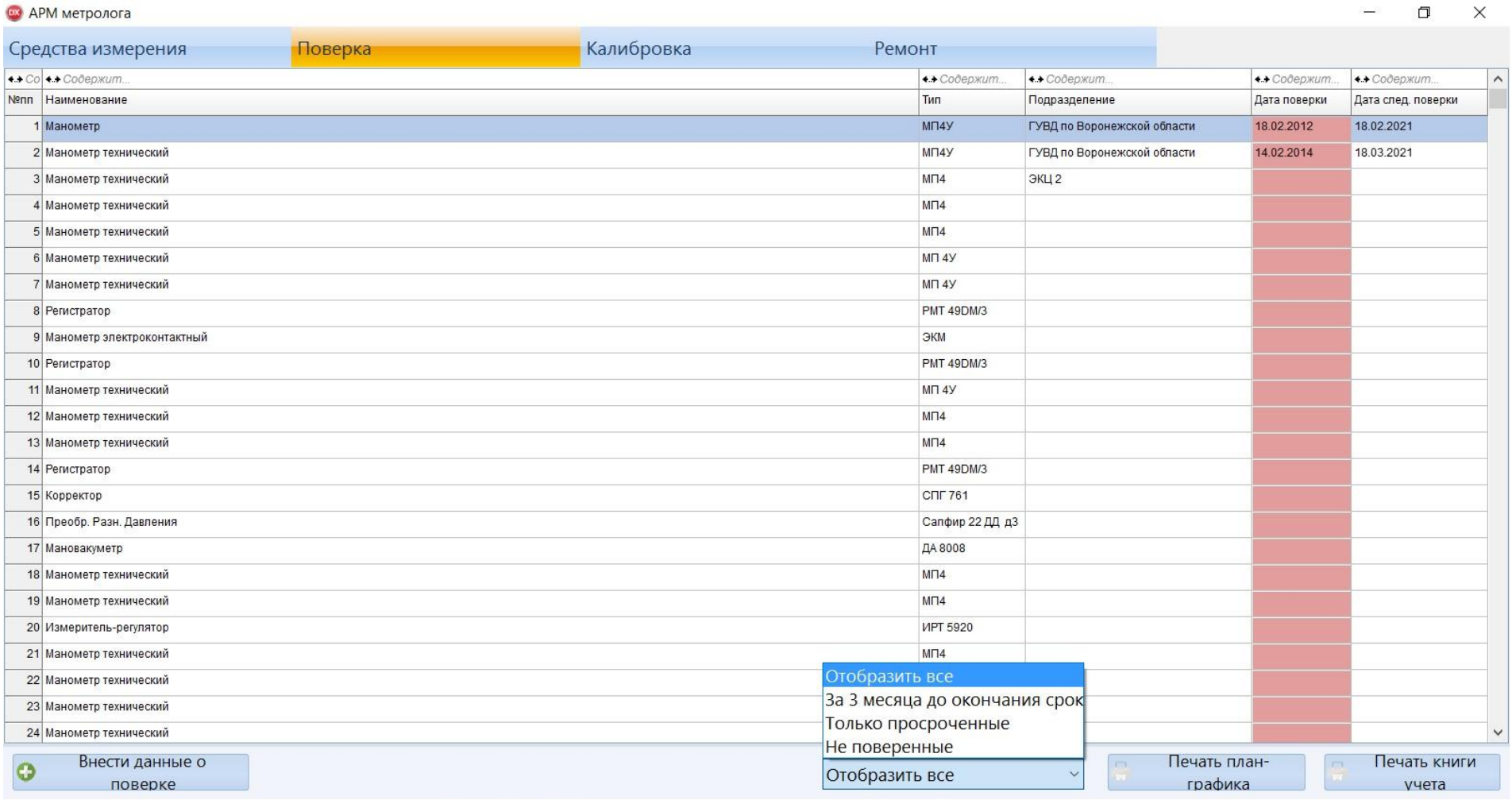

Рис. 5. Выбор маркера параметра отображения.

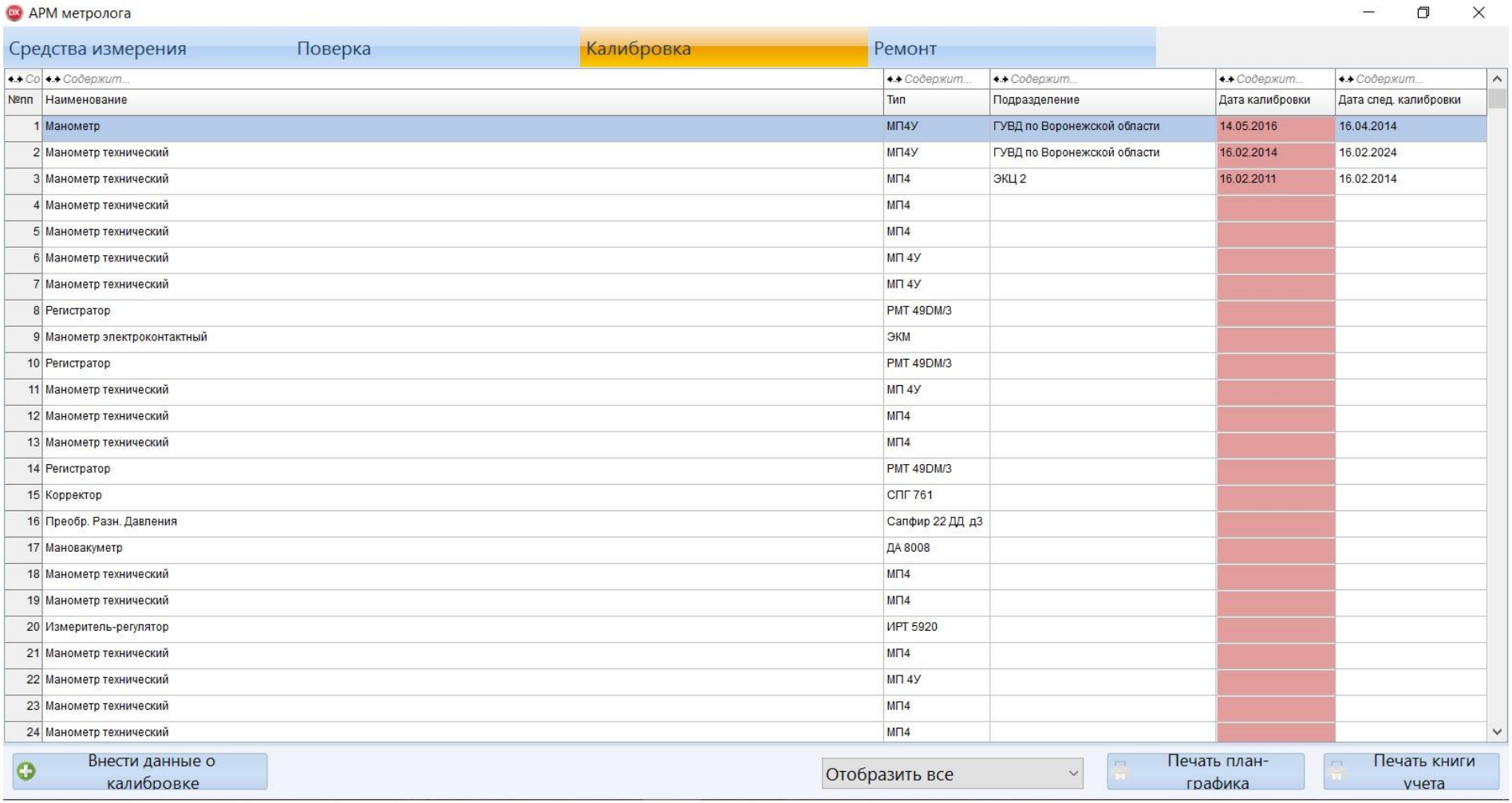

Рис. 6. Рабочее окно "Калибровка".

При нажатии клавиши «Внести данные о калибровке» открывается окно ввода информации о поверке устройства – рисунок 7.

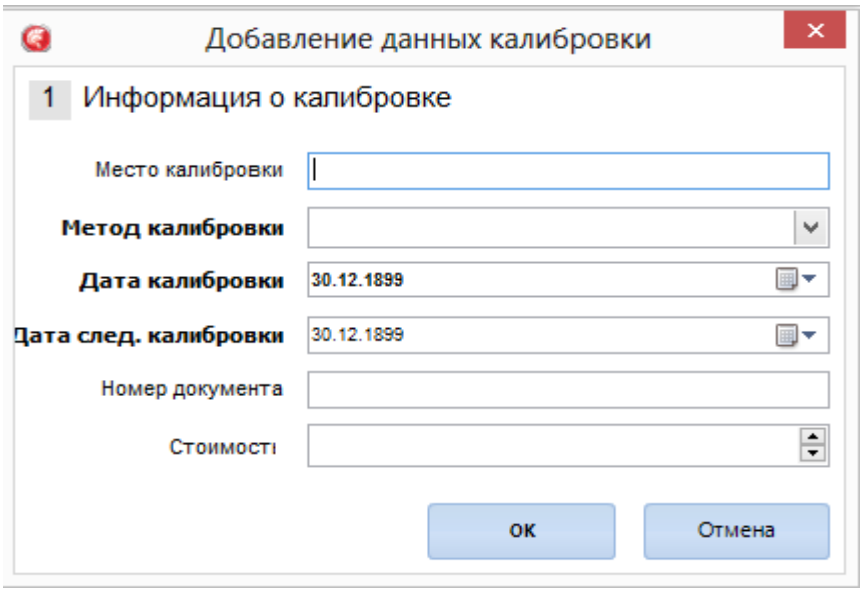

Рис. 7. Окно ввода информации о калибровке.

В рабочем окне «Калибровка» возможно осуществить отображение приборов по следующим параметрам:

- отобразить все,

- отобразить за 3 месяца до окончания срока,

- только просроченные,

- не поверенные.

Для выбора соответствующего маркера необходимо в правом нижнем углу программы в режиме «Калибровка» выбрать соответствующий раздел (рисунок 8).

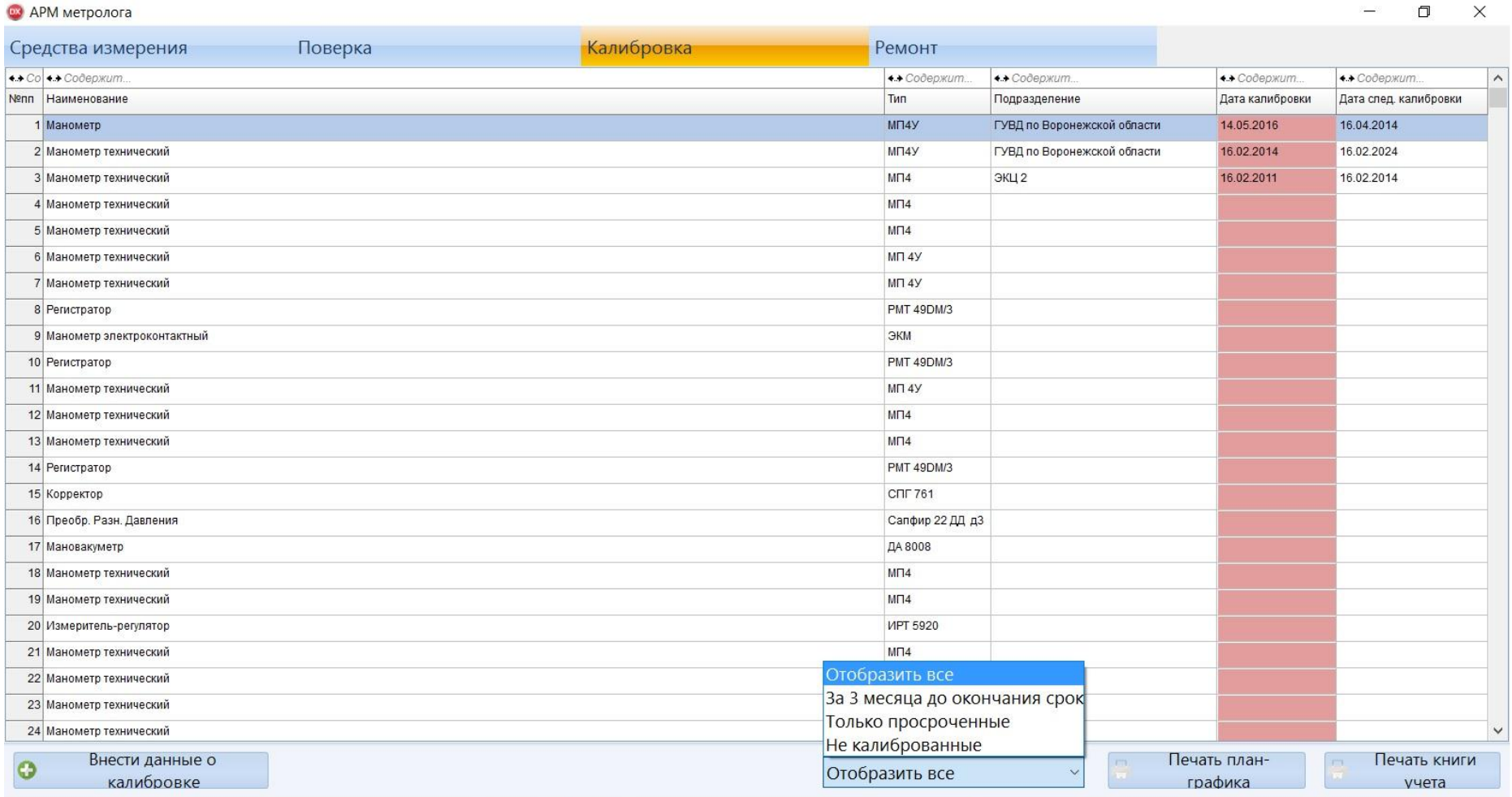

Рис. 8. Выбор маркера параметра отображения.

Редактирование данных о поверке и калибровке.

Редактирование данных о поверке или калибровке осуществляется в соответствующих окнах «Поверка» или «Калибровка» путем выделения прибора из общего перечня и нажатием клавиши «Внести данные о поверке» или «Внести данные о калибровке».

Для выделения прибора необходимо кликнуть на соответствующем приборе и нажать правую клавишу мыши.

Ввод данных о ремонте прибора.

Ввод данных о ремонте прибора осуществляется путем заполнения необходимых полей таблиц в рабочем окне «Ремонт» (рисунок 9), при нажатии клавиши «Внести данные о ремонте». Окно ввода информации о ремонте устройства имеет вид – рисунок 10.

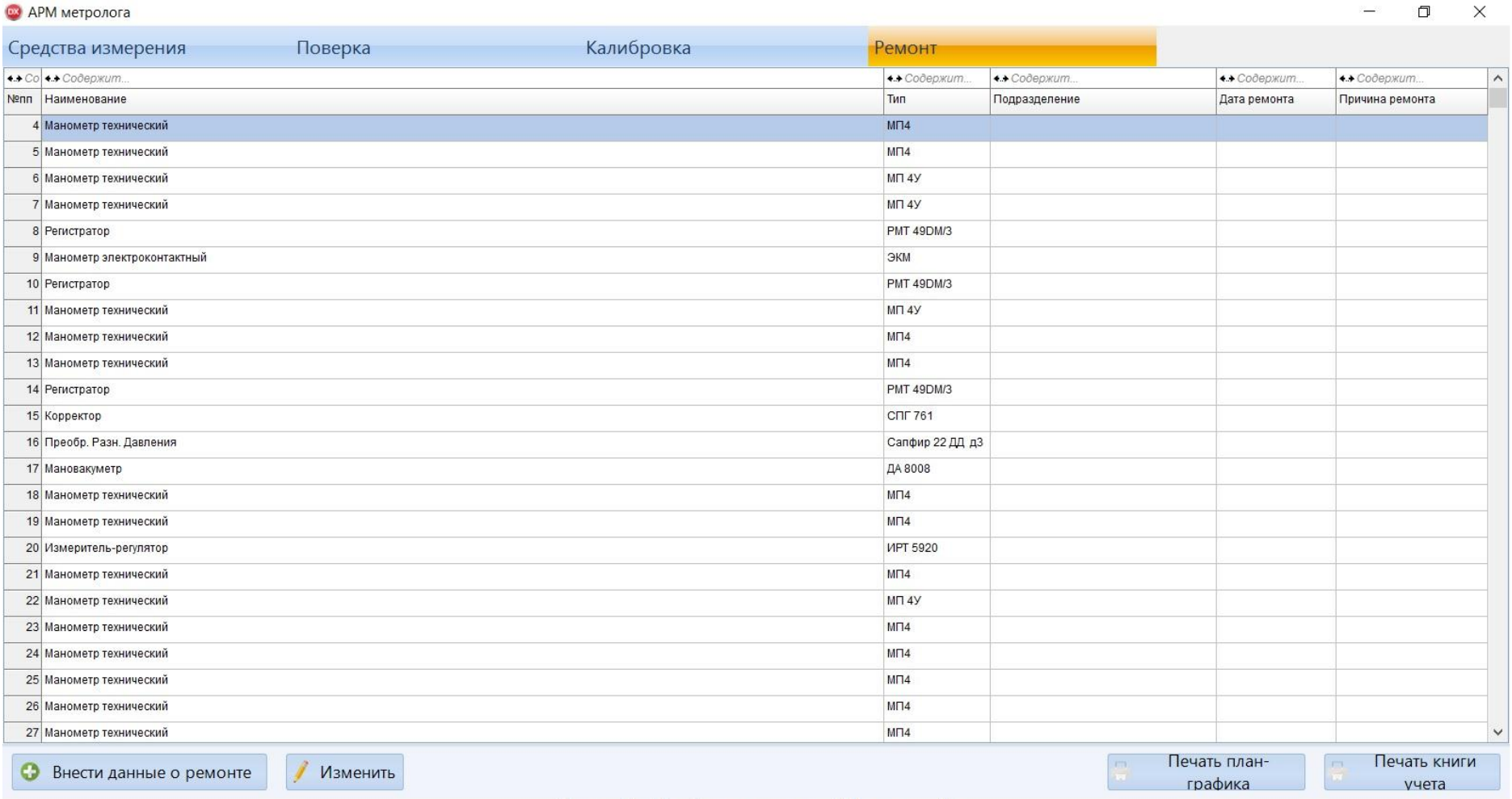

Рис. 9. Рабочее окно "Ремонт".

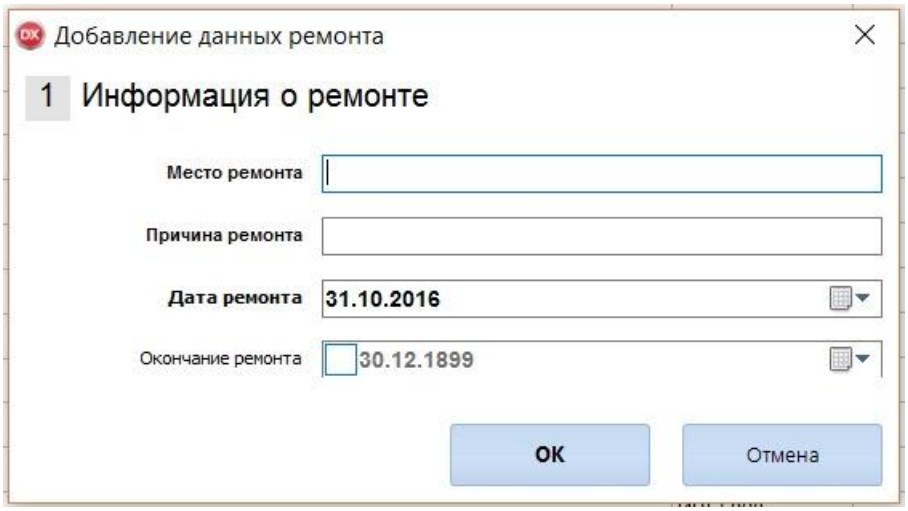

Рис. 10. Окно добавления информации о ремонте.

При необходимости внесения изменения в данные о ремонте следует нажать кнопку «Изменить» (рисунок 9), затем в появившемся окне изменения данных ремонта (рисунок 11) внести изменения в информацию о ремонте.

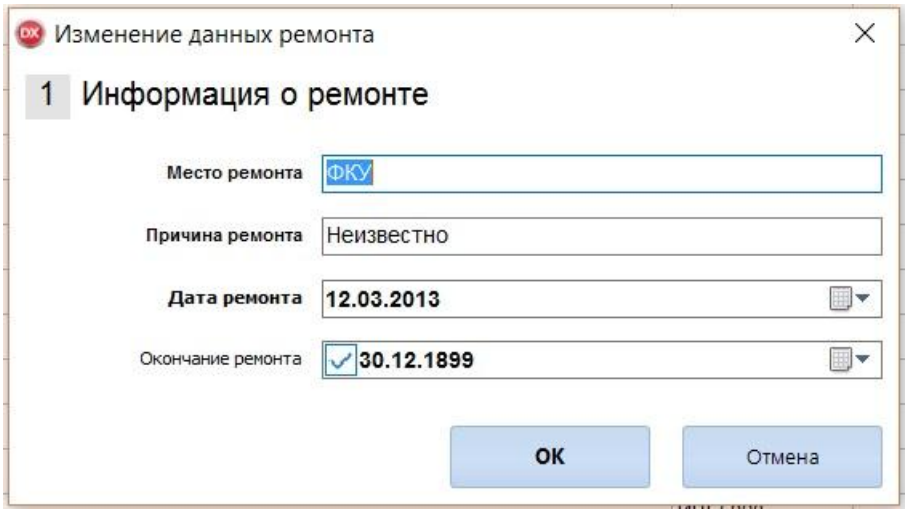

Рис. 11. Окно изменения данных ремонта.

Печать план-графика.

Печать план-графика осуществляется путем нажатия клавиши «Печать план-графика» (рисунок 9).

Печать книги учета.

Печать книги учета осуществляется путем нажатия клавиши «Печать книги учета» (рисунок 9).

Приложение

Листинг программы

```
unit CalibrationUnit;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls,
AdvDateTimePicker,
  AdvDBDateTimePicker, AdvGlowButton, Vcl.ExtCtrls, Vcl.Mask,
AdvSpin, DBAdvSp,
  AdvDBLookupComboBox, Vcl.StdCtrls, AdvEdit, DBAdvEd;

type
  TCalibrationForm = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    PlaceEdit: TDBAdvEdit;
    PriceSpinEdit: TDBAdvSpinEdit;
    Panel1: TPanel;
    CancelButton: TAdvGlowButton;
    OKButton: TAdvGlowButton;
    CheckingMethodComboBox: TAdvDBLookupComboBox;
    StartDatePicker: TAdvDBDateTimePicker;
    EndDatePicker: TAdvDBDateTimePicker;
    DocumentNumberEdit: TDBAdvEdit;
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure FormShow(Sender: TObject);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    DisplayType: integer;
    procedure SetControlsEnabled(Value: Boolean);
  end;

var
  CalibrationForm: TCalibrationForm;

implementation

{$R *.dfm}

uses DMUnit, MainUnit;

procedure TCalibrationForm.FormKeyPress(Sender: TObject; var Key:
Char);
begin
  if Ord(Key) = VK_ESCAPE then
    ModalResult := mrClose;
end;

procedure TCalibrationForm.FormShow(Sender: TObject);
begin
```

```
  case DisplayType of
    DISPLAY_TYPE_VIEW: Caption := 'Просмотр данных калибровки';
    DISPLAY_TYPE_ADD: Caption := 'Добавление данных калибровки';
    DISPLAY_TYPE_UPDATE: Caption := 'Изменение данных калибровки';
  end;

  CancelButton.Visible := (DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE);
  OKButton.Align := alLeft;
  OKButton.Align := alRight;
  SetControlsEnabled((DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE));
end;

procedure TCalibrationForm.OKButtonClick(Sender: TObject);
begin
  if (DisplayType = DISPLAY_TYPE_VIEW) then begin
    ModalResult := mrOk;
    Exit;
  end;

  if (CheckingMethodComboBox.ItemIndex = -1) then begin
    MessageBox(Handle, PChar('Не заполнено поле "Метод поверки".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  ModalResult := mrOk;
end;

procedure TCalibrationForm.SetControlsEnabled(Value: Boolean);
begin
  PlaceEdit.EditorEnabled := Value;
  CheckingMethodComboBox.EditorEnabled := Value;
  CheckingMethodComboBox.Enabled := Value;
  StartDatePicker.Enabled := Value;
  EndDatePicker.Enabled := Value;
  DocumentNumberEdit.EditorEnabled := Value;
  PriceSpinEdit.EditorEnabled := Value;

  if Value then
    PriceSpinEdit.IncrementFloat := 0.1
  else
    PriceSpinEdit.IncrementFloat := 0;
end;

end.

unit CheckingUnit;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls,
AdvDateTimePicker,
```

```
  AdvDBDateTimePicker, AdvGlowButton, Vcl.ExtCtrls, Vcl.Mask,
AdvSpin, DBAdvSp,
  AdvDBLookupComboBox, Vcl.StdCtrls, AdvEdit, DBAdvEd;

type
  TCheckingForm = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    PlaceEdit: TDBAdvEdit;
    PriceSpinEdit: TDBAdvSpinEdit;
    Panel1: TPanel;
    CancelButton: TAdvGlowButton;
    OKButton: TAdvGlowButton;
    CheckingMethodComboBox: TAdvDBLookupComboBox;
    StartDatePicker: TAdvDBDateTimePicker;
    EndDatePicker: TAdvDBDateTimePicker;
    DocumentNumberEdit: TDBAdvEdit;
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure FormShow(Sender: TObject);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    DisplayType: integer;
    procedure SetControlsEnabled(Value: Boolean);
  end;

var
  CheckingForm: TCheckingForm;

implementation

{$R *.dfm}

uses DMUnit, MainUnit;

procedure TCheckingForm.FormKeyPress(Sender: TObject; var Key:
Char);
begin
  if Ord(Key) = VK_ESCAPE then
    ModalResult := mrClose;
end;

procedure TCheckingForm.FormShow(Sender: TObject);
begin
  case DisplayType of
    DISPLAY_TYPE_VIEW: Caption := 'Просмотр данных поверки';
    DISPLAY_TYPE_ADD: Caption := 'Добавление данных поверки';
    DISPLAY_TYPE_UPDATE: Caption := 'Изменение данных поверки';
  end;

  CancelButton.Visible := (DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE);
  OKButton.Align := alLeft;
  OKButton.Align := alRight;
  SetControlsEnabled((DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE));
```

```
end;

procedure TCheckingForm.OKButtonClick(Sender: TObject);
begin
  if (DisplayType = DISPLAY_TYPE_VIEW) then begin
    ModalResult := mrOk;
    Exit;
  end;

  if (CheckingMethodComboBox.ItemIndex = -1) then begin
    MessageBox(Handle, PChar('Не заполнено поле "Метод поверки".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  ModalResult := mrOk;
end;

procedure TCheckingForm.SetControlsEnabled(Value: Boolean);
begin
  PlaceEdit.EditorEnabled := Value;
  CheckingMethodComboBox.EditorEnabled := Value;
  CheckingMethodComboBox.Enabled := Value;
  StartDatePicker.Enabled := Value;
  EndDatePicker.Enabled := Value;
  DocumentNumberEdit.EditorEnabled := Value;
  PriceSpinEdit.EditorEnabled := Value;

  if Value then
    PriceSpinEdit.IncrementFloat := 0.1
  else
    PriceSpinEdit.IncrementFloat := 0;
end;

end.

unit DBItemUnit;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, AdvEdit,
Vcl.Mask,
  Vcl.DBCtrls, AdvGlowButton, Vcl.ExtCtrls, Vcl.ComCtrls,
AdvDateTimePicker,
  AdvDBDateTimePicker, AdvSpin, DBAdvSp, AdvDBLookupComboBox,
DBAdvEd, AdvCombo,
  AdvDBComboBox;

type
  TDBItemForm = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    NameEdit: TDBAdvEdit;
    TypeEdit: TDBAdvEdit;
```

```
    CategoryComboBox: TAdvDBLookupComboBox;
    AccuracyClassSpinEdit: TDBAdvSpinEdit;
    UnitDimensionEdit: TDBAdvEdit;
    LowerLimitSpinEdit: TDBAdvSpinEdit;
    UpperLimitSpinEdit: TDBAdvSpinEdit;
    MeasureErrorSpinEdit: TDBAdvSpinEdit;
    ManufacturerEdit: TDBAdvEdit;
    MeasureTypeComboBox: TAdvDBLookupComboBox;
    SerialNumberEdit: TDBAdvEdit;
    CreateDatePicker: TAdvDBDateTimePicker;
    LifetimeSpinEdit: TDBAdvSpinEdit;
    GoldSpinEdit: TDBAdvSpinEdit;
    SilverSpinEdit: TDBAdvSpinEdit;
    PlatinumSpinEdit: TDBAdvSpinEdit;
    PalladiumSpinEdit: TDBAdvSpinEdit;
    MercurySpinEdit: TDBAdvSpinEdit;
    PriceSpinEdit: TDBAdvSpinEdit;
    CommentsEdit: TDBAdvEdit;
    Label3: TLabel;
    Label4: TLabel;
    CodeEdit: TDBAdvEdit;
    InstallPlaceEdit: TDBAdvEdit;
    MeasureParameterEdit: TDBAdvEdit;
    InstallDatePicker: TAdvDBDateTimePicker;
    IndicatorComboBox: TAdvDBLookupComboBox;
    InventoryNumberEdit: TDBAdvEdit;
    OutSignalTypeEdit: TDBAdvEdit;
    Panel1: TPanel;
    CancelButton: TAdvGlowButton;
    OKButton: TAdvGlowButton;
    DepartmentEdit: TAdvDBComboBox;
    PrimaryComboBox: TAdvDBComboBox;
    ConditionComboBox: TAdvDBLookupComboBox;
    CheckingPeriodEdit: TDBAdvEdit;
    TimestampDatePicker: TAdvDBDateTimePicker;
    procedure FormShow(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    DisplayType: integer;
    procedure SetControlsEnabled(Value: Boolean);
  end;

var
  DBItemForm: TDBItemForm;

implementation

{$R *.dfm}

uses DMUnit, MainUnit;

procedure TDBItemForm.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Ord(Key) = VK_ESCAPE then
```

```
    ModalResult := mrClose;
end;

procedure TDBItemForm.FormShow(Sender: TObject);
begin
  case DisplayType of
    DISPLAY_TYPE_VIEW: Caption := 'Просмотр данных средства
измерения';
    DISPLAY_TYPE_ADD: Caption := 'Добавление средства измерения';
    DISPLAY_TYPE_UPDATE: Caption := 'Изменение данных средства
измерения';
  end;

  CancelButton.Visible := (DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE);
  OKButton.Align := alLeft;
  OKButton.Align := alRight;

  SetControlsEnabled((DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE));
end;

procedure TDBItemForm.OKButtonClick(Sender: TObject);
begin
  if (DepartmentEdit.ItemIndex = -1) then begin
    DepartmentEdit.AddDisplayAndStoredValue(DepartmentEdit.Text,
DepartmentEdit.Text);
    DepartmentEdit.ItemIndex := DepartmentEdit.Items.Count - 1;
  end;

  if (DisplayType = DISPLAY_TYPE_VIEW) then begin
    ModalResult := mrOk;
    Exit;
  end;

  if (NameEdit.Text = '') then begin
    MessageBox(Handle, PChar('Не заполнено поле "Название СИ".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  if (TypeEdit.Text = '') then begin
    MessageBox(Handle, PChar('Не заполнено поле "Тип СИ".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  if (CategoryComboBox.ItemIndex = -1) then begin
    MessageBox(Handle, PChar('Не заполнено поле "Категория".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  if (MeasureTypeComboBox.ItemIndex = -1) then begin
    MessageBox(Handle, PChar('Не заполнено поле "Вид измерения".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
```

```
  end;

  ModalResult := mrOk;
end;

procedure TDBItemForm.SetControlsEnabled(Value: Boolean);
var
  OldText: string;
begin
  NameEdit.EditorEnabled := Value;
  TypeEdit.EditorEnabled := Value;
  CategoryComboBox.EditorEnabled := Value;
  UnitDimensionEdit.EditorEnabled := Value;
  AccuracyClassSpinEdit.EditorEnabled := Value;
  LowerLimitSpinEdit.EditorEnabled := Value;
  UpperLimitSpinEdit.EditorEnabled := Value;
  MeasureErrorSpinEdit.EditorEnabled := Value;
  ManufacturerEdit.EditorEnabled := Value;
  CommentsEdit.EditorEnabled := Value;
  MeasureTypeComboBox.EditorEnabled := Value;
  SerialNumberEdit.EditorEnabled := Value;
  CreateDatePicker.Enabled := Value;
  LifetimeSpinEdit.EditorEnabled := Value;
  GoldSpinEdit.EditorEnabled := Value;
  SilverSpinEdit.EditorEnabled := Value;
  PlatinumSpinEdit.EditorEnabled := Value;
  PalladiumSpinEdit.EditorEnabled := Value;
  MercurySpinEdit.EditorEnabled := Value;
  PriceSpinEdit.EditorEnabled := Value;
  CodeEdit.EditorEnabled := Value;
  InstallPlaceEdit.EditorEnabled := Value;
  MeasureParameterEdit.EditorEnabled := Value;
  PrimaryComboBox.Enabled := Value;
  InstallDatePicker.Enabled := Value;
  IndicatorComboBox.EditorEnabled := Value;
  InventoryNumberEdit.EditorEnabled := Value;
  OutSignalTypeEdit.EditorEnabled := Value;
  DepartmentEdit.Enabled := Value;
  CheckingPeriodEdit.EditorEnabled := Value;
  TimestampDatePicker.Enabled := Value;

  if DisplayType = DISPLAY_TYPE_ADD then
    TimestampDatePicker.Date := Date;

  if Value then
    AccuracyClassSpinEdit.IncrementFloat := 0.1
  else
    AccuracyClassSpinEdit.IncrementFloat := 0;

  if Value then
    LowerLimitSpinEdit.IncrementFloat := 0.01
  else
    LowerLimitSpinEdit.IncrementFloat := 0;

  if Value then
    UpperLimitSpinEdit.IncrementFloat := 0.01
  else
```

```
  UpperLimitSpinEdit.IncrementFloat := 0;

if Value then
  MeasureErrorSpinEdit.IncrementFloat := 0.01
else
  MeasureErrorSpinEdit.IncrementFloat := 0;

if Value then
  LifetimeSpinEdit.Increment := 1
else
  LifetimeSpinEdit.Increment := 0;

if Value then
  GoldSpinEdit.IncrementFloat := 0.01
else
  GoldSpinEdit.IncrementFloat := 0;

if Value then
  SilverSpinEdit.IncrementFloat := 0.01
else
  SilverSpinEdit.IncrementFloat := 0;

if Value then
  PlatinumSpinEdit.IncrementFloat := 0.01
else
  PlatinumSpinEdit.IncrementFloat := 0;

if Value then
  PalladiumSpinEdit.IncrementFloat := 0.01
else
  PalladiumSpinEdit.IncrementFloat := 0;

if Value then
  MercurySpinEdit.IncrementFloat := 0.01
else
  MercurySpinEdit.IncrementFloat := 0;

if Value then
  PriceSpinEdit.IncrementFloat := 0.1
else
  PriceSpinEdit.IncrementFloat := 0;

CategoryComboBox.Enabled := Value;
MeasureTypeComboBox.Enabled := Value;
PrimaryComboBox.Enabled := Value;
ConditionComboBox.Enabled := Value;
IndicatorComboBox.Enabled := Value;

DepartmentEdit.Items.Clear;
DM.DevicesLookupQuery.Active := False;
DM.DevicesLookupQuery.Active := True;

DM.DevicesLookupDS.DataSet.First;

OldText := DepartmentEdit.Text;

while (not DM.DevicesLookupDS.DataSet.Eof) do
```

```
  begin
    if (VarToStr(DM.DevicesLookupDS.DataSet.FieldValues[
'DEPARTMENT' ]) <> '') then begin

DepartmentEdit.AddDisplayAndStoredValue(VarToStr(DM.DevicesLookupDS.
DataSet.FieldValues[ 'DEPARTMENT' ]),
        VarToStr(DM.DevicesLookupDS.DataSet.FieldValues[
'DEPARTMENT' ]));
    end;

    DM.DevicesLookupDS.DataSet.Next;
  end;

  DepartmentEdit.Text := OldText;
end;

end.

unit DMUnit;

interface

uses
  System.SysUtils, System.Classes, Data.DB, UniProvider,
SQLiteUniProvider,
  DBAccess, Uni, MemDS;

const
  CheckingQuerySQL = 'SELECT DEVICES.ID, DEVICES.NAME, DEVICES.TYPE,
c.ID as C_ID, c.DATE, c.END_DATE' + #10 +
    'FROM DEVICES' + #10 +
    'LEFT OUTER JOIN (SELECT * FROM CHECKING ORDER BY END_DATE DESC)
c ON c.DEVICE_ID = DEVICES.ID' + #10 +
    'GROUP BY DEVICES.ID' + #10;
  CalibrationQuerySQL = 'SELECT DEVICES.ID, DEVICES.NAME,
DEVICES.TYPE, c.ID as C_ID, c.DATE, c.END_DATE' + #10 +
    'FROM DEVICES' + #10 +
    'LEFT OUTER JOIN (SELECT * FROM CALIBRATION ORDER BY END_DATE
DESC) c ON c.DEVICE_ID = DEVICES.ID' + #10 +
    'GROUP BY DEVICES.ID' + #10;
  DevicesSQL = 'SELECT DEVICES.*, MEASURE_TYPE.NAME as
MEASURE_TYPE_NAME' + #10 +
    'FROM DEVICES' + #10 +
    'LEFT OUTER JOIN MEASURE_TYPE ON MEASURE_TYPE.ID =
DEVICES.MEASURE_TYPE_ID' + #10;
  RepairQuerySQL = 'SELECT DEVICES.*, strftime("%Y",
DEVICES.CREATE_DATE) as CREATE_YEAR, c.ID as C_ID, c.DATE, c.REASON,
GROUP_CONCAT(c.DATE) as REPAIR_DATE, GROUP_CONCAT(c.END_DATE) as
REPAIR_END_DATE, ch.DATE as CHECKING_DATE, M.NAME as
MEASURE_TYPE_NAME' + #10 +
    'FROM DEVICES' + #10 +
    'LEFT OUTER JOIN (SELECT ID, DATE, IFNULL(END_DATE, "") as
END_DATE, REASON, PLACE, DEVICE_ID FROM REPAIR ORDER BY DATE DESC) c
ON c.DEVICE_ID = DEVICES.ID' + #10 +
    'LEFT OUTER JOIN (SELECT DATE, DEVICE_ID FROM (SELECT DATE,
DEVICE_ID FROM CHECKING ORDER BY DATE DESC ) GROUP BY DEVICE_ID) ch
ON ch.DEVICE_ID = DEVICES.ID' + #10 +
```

```
    'LEFT OUTER JOIN MEASURE_TYPE M ON M.ID =
DEVICES.MEASURE_TYPE_ID' + #10 +
    #10 +
    'GROUP BY DEVICES.ID' + #10;

type
  TDM = class(TDataModule)
    Database: TUniConnection;
    SQLiteUniProvider: TSQLiteUniProvider;
    DevicesTable: TUniTable;
    CalibrationTable: TUniTable;
    CheckingTable: TUniTable;
    CategoryTable: TUniTable;
    CheckingMethodTable: TUniTable;
    ConditionTable: TUniTable;
    MeasureTypeTable: TUniTable;
    RepairTable: TUniTable;
    DevicesDS: TUniDataSource;
    CalibrationDS: TUniDataSource;
    CheckingDS: TUniDataSource;
    RepairDS: TUniDataSource;
    CategoryDS: TUniDataSource;
    CheckingMethodDS: TUniDataSource;
    ConditionDS: TUniDataSource;
    MeasureTypeDS: TUniDataSource;
    CheckingQuery: TUniQuery;
    CheckingQDS: TUniDataSource;
    CheckingQueryID: TIntegerField;
    CheckingQueryNAME: TStringField;
    CheckingQueryTYPE: TStringField;
    CheckingQueryDATE: TDateField;
    CheckingQueryEND_DATE: TDateField;
    CheckingQueryC_ID: TIntegerField;
    CategoryTableID: TIntegerField;
    CategoryTableNAME: TStringField;
    CalibrationQuery: TUniQuery;
    IntegerField1: TIntegerField;
    StringField1: TStringField;
    StringField2: TStringField;
    DateField1: TDateField;
    DateField2: TDateField;
    IntegerField2: TIntegerField;
    CalibrationQDS: TUniDataSource;
    CheckingTableID: TIntegerField;
    CheckingTableDEVICE_ID: TIntegerField;
    CheckingTableDATE: TDateField;
    CheckingTableCHECKING_METHOD_ID: TIntegerField;
    CheckingTablePLACE: TStringField;
    CheckingTableEND_DATE: TDateField;
    CheckingTableDOCUMENT_NUMBER: TStringField;
    CheckingTablePRICE: TFloatField;
    CalibrationTableID: TIntegerField;
    CalibrationTableDEVICE_ID: TIntegerField;
    CalibrationTableDATE: TDateField;
    CalibrationTableCHECKING_METHOD_ID: TIntegerField;
    CalibrationTablePLACE: TStringField;
    CalibrationTableEND_DATE: TDateField;
```

```
CalibrationTableDOCUMENT_NUMBER: TIntegerField;
CalibrationTablePRICE: TFloatField;
DevicesLookupDS: TUniDataSource;
DevicesLookupQuery: TUniQuery;
DevicesLookupQueryDEPARTMENT: TStringField;
CalibrationQueryDEPARTMENT: TStringField;
CheckingQueryDEPARTMENT: TStringField;
DevicesQuery: TUniQuery;
DevicesQueryID: TIntegerField;
DevicesQueryNAME: TStringField;
DevicesQueryTYPE: TStringField;
DevicesQueryCATEGORY_ID: TIntegerField;
DevicesQueryUNIT_DIMENSION: TStringField;
DevicesQueryACCURACY_CLASS: TFloatField;
DevicesQueryLOWER_LIMIT: TFloatField;
DevicesQueryUPPER_LIMIT: TFloatField;
DevicesQueryMEASURE_ERROR: TFloatField;
DevicesQueryMANUFACTURER: TStringField;
DevicesQueryCOMMENTS: TStringField;
DevicesQueryMEASURE_TYPE_ID: TIntegerField;
DevicesQuerySERIAL_NUMBER: TStringField;
DevicesQueryCREATE_DATE: TDateField;
DevicesQueryLIFETIME: TIntegerField;
DevicesQueryGOLD: TFloatField;
DevicesQuerySILVER: TFloatField;
DevicesQueryPLATINUM: TFloatField;
DevicesQueryPALLADIUM: TFloatField;
DevicesQueryMERCURY: TFloatField;
DevicesQueryPRICE: TFloatField;
DevicesQueryCODE: TIntegerField;
DevicesQueryINSTALL_PLACE: TStringField;
DevicesQueryMEASURE_PARAMETER: TStringField;
DevicesQueryIS_PRIMARY: TSmallintField;
DevicesQueryCONDITION_ID: TIntegerField;
DevicesQueryINSTALL_DATE: TDateField;
DevicesQueryIS_INDICATOR: TSmallintField;
DevicesQueryINVENTORY_NUMBER: TStringField;
DevicesQueryOUT_SIGNAL_TYPE: TStringField;
DevicesQueryDEPARTMENT: TStringField;
DevicesQueryMEASURE_TYPE_NAME: TStringField;
DevicesTableID: TIntegerField;
DevicesTableNAME: TStringField;
DevicesTableTYPE: TStringField;
DevicesTableCATEGORY_ID: TIntegerField;
DevicesTableUNIT_DIMENSION: TStringField;
DevicesTableACCURACY_CLASS: TFloatField;
DevicesTableLOWER_LIMIT: TFloatField;
DevicesTableUPPER_LIMIT: TFloatField;
DevicesTableMEASURE_ERROR: TFloatField;
DevicesTableMANUFACTURER: TStringField;
DevicesTableCOMMENTS: TStringField;
DevicesTableMEASURE_TYPE_ID: TIntegerField;
DevicesTableSERIAL_NUMBER: TStringField;
DevicesTableCREATE_DATE: TDateField;
DevicesTableLIFETIME: TIntegerField;
DevicesTableGOLD: TFloatField;
DevicesTableSILVER: TFloatField;
```

```
DevicesTablePLATINUM: TFloatField;
DevicesTablePALLADIUM: TFloatField;
DevicesTableMERCURY: TFloatField;
DevicesTablePRICE: TFloatField;
DevicesTableCODE: TIntegerField;
DevicesTableINSTALL_PLACE: TStringField;
DevicesTableMEASURE_PARAMETER: TStringField;
DevicesTableIS_PRIMARY: TSmallintField;
DevicesTableCONDITION_ID: TIntegerField;
DevicesTableINSTALL_DATE: TDateField;
DevicesTableIS_INDICATOR: TSmallintField;
DevicesTableINVENTORY_NUMBER: TStringField;
DevicesTableOUT_SIGNAL_TYPE: TStringField;
DevicesTableDEPARTMENT: TStringField;
CheckingMethodTableID: TIntegerField;
CheckingMethodTableNAME: TStringField;
ConditionTableID: TIntegerField;
ConditionTableNAME: TStringField;
MeasureTypeTableID: TIntegerField;
MeasureTypeTableNAME: TStringField;
RepairQuery: TUniQuery;
RepairQDS: TUniDataSource;
RepairQueryID: TIntegerField;
RepairQueryNAME: TStringField;
RepairQueryTYPE: TStringField;
RepairQueryDEPARTMENT: TStringField;
RepairQueryC_ID: TIntegerField;
RepairQueryDATE: TDateField;
RepairQueryREASON: TStringField;
RepairTableID: TIntegerField;
RepairTableDATE: TDateField;
RepairTableREASON: TStringField;
RepairTablePLACE: TStringField;
RepairTableDEVICE_ID: TIntegerField;
RepairQueryCATEGORY_ID: TIntegerField;
RepairQueryUNIT_DIMENSION: TStringField;
RepairQueryACCURACY_CLASS: TFloatField;
RepairQueryLOWER_LIMIT: TFloatField;
RepairQueryUPPER_LIMIT: TFloatField;
RepairQueryMEASURE_ERROR: TFloatField;
RepairQueryMANUFACTURER: TStringField;
RepairQueryCOMMENTS: TStringField;
RepairQueryMEASURE_TYPE_ID: TIntegerField;
RepairQuerySERIAL_NUMBER: TStringField;
RepairQueryCREATE_DATE: TDateField;
RepairQueryLIFETIME: TIntegerField;
RepairQueryGOLD: TFloatField;
RepairQuerySILVER: TFloatField;
RepairQueryPLATINUM: TFloatField;
RepairQueryPALLADIUM: TFloatField;
RepairQueryMERCURY: TFloatField;
RepairQueryPRICE: TFloatField;
RepairQueryCODE: TIntegerField;
RepairQueryINSTALL_PLACE: TStringField;
RepairQueryMEASURE_PARAMETER: TStringField;
RepairQueryIS_PRIMARY: TSmallintField;
RepairQueryCONDITION_ID: TIntegerField;
```

```
    RepairQueryINSTALL_DATE: TDateField;
    RepairQueryIS_INDICATOR: TSmallintField;
    RepairQueryINVENTORY_NUMBER: TStringField;
    RepairQueryOUT_SIGNAL_TYPE: TStringField;
    RepairQueryCREATE_YEAR: TMemoField;
    RepairQueryREPAIR_DATE: TMemoField;
    RepairQueryMEASURE_TYPE_NAME: TStringField;
    RepairQueryCHECKING_PERIOD: TIntegerField;
    DevicesQueryCHECKING_PERIOD: TIntegerField;
    RepairQueryCHECKING_DATE: TDateField;
    RepairQueryTIMESTAMP: TDateField;
    DevicesQueryTIMESTAMP: TDateField;
    RepairTableEND_DATE: TDateField;
    RepairQueryREPAIR_END_DATE: TMemoField;
    procedure DataModuleCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  DM: TDM;

implementation

{%CLASSGROUP 'Vcl.Controls.TControl'}

{$R *.dfm}

procedure TDM.DataModuleCreate(Sender: TObject);
begin
  Database.Disconnect;
  Database.Database := ExtractFilePath(ParamStr(0)) + 'metr.sqlite';
  Database.Connect;

  //DevicesTable.Active := True;
  CalibrationTable.Active := True;
  CheckingTable.Active := True;
  RepairTable.Active := True;
  CategoryTable.Active := True;
  CheckingMethodTable.Active := True;
  ConditionTable.Active := True;
  MeasureTypeTable.Active := True;

  CheckingQuery.Active := True;
  CalibrationQuery.Active := True;
  DevicesLookupQuery.Active := True;
  DevicesQuery.Active := True;
  RepairQuery.Active := True;
end;

end.

unit MainUnit;

interface
```

```
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.DBGrids,
AdvObj,
  BaseGrid, AdvGrid, DBAdvGrid, W7Bars, W7Classes, W7Buttons,
Vcl.ExtCtrls,
  AdvGlowButton, DateUtils, Vcl.StdCtrls, AdvCombo, frxClass,
frxDesgn;

const
  DISPLAY_TYPE_VIEW = 0;
  DISPLAY_TYPE_ADD = 1;
  DISPLAY_TYPE_UPDATE = 2;

type
  TMainForm = class(TForm)
    TopPanel: TPanel;
    DBPageSelector: TW7PageSelector;
    CalibrationPageSelector: TW7PageSelector;
    CheckingPageSelector: TW7PageSelector;
    Pager: TNotebook;
    DBAdvGrid: TDBAdvGrid;
    CheckingAdvGrid: TDBAdvGrid;
    DBBottomPanel: TW7InformationBar;
    AddDBButton: TAdvGlowButton;
    UpdateDBButton: TAdvGlowButton;
    DeleteDBButton: TAdvGlowButton;
    CheckingBottomPanel: TW7InformationBar;
    AddCheckingButton: TAdvGlowButton;
    CalibrationBottomPanel: TW7InformationBar;
    AddCalibrationButton: TAdvGlowButton;
    DisplayCalibrationComboBox: TAdvComboBox;
    DisplayCheckingComboBox: TAdvComboBox;
    CalibrationAdvGrid: TDBAdvGrid;
    RepairAdvGrid: TDBAdvGrid;
    RepairBottomPanel: TW7InformationBar;
    AddRepairButton: TAdvGlowButton;
    RepairPrintRepairButton: TAdvGlowButton;
    RepairPageSelector: TW7PageSelector;
    frxRepairReport: TfrxReport;
    frxRepairUserDataSet: TfrxUserDataSet;
    frxCheckingReport: TfrxReport;
    frxCheckingUserDataSet: TfrxUserDataSet;
    RepairPrintCheckingPlanButton: TAdvGlowButton;
    CalibrationPrintRepairButton: TAdvGlowButton;
    CalibrationPrintCheckingPlanButton: TAdvGlowButton;
    CheckingPrintRepairButton: TAdvGlowButton;
    CheckingPrintCheckingPlanButton: TAdvGlowButton;
    ListPrintRepairButton: TAdvGlowButton;
    ListPrintCheckingPlanButton: TAdvGlowButton;
    UpdateRepairButton: TAdvGlowButton;
    procedure DBPageSelectorClick(Sender: TObject);
    procedure CheckingPageSelectorClick(Sender: TObject);
    procedure CalibrationPageSelectorClick(Sender: TObject);
    procedure AddDBButtonClick(Sender: TObject);
```

```
    procedure DBAdvGridDblClickCell(Sender: TObject; ARow, ACol:
Integer);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure UpdateDBButtonClick(Sender: TObject);
    procedure DeleteDBButtonClick(Sender: TObject);
    procedure PrintDBButtonClick(Sender: TObject);
    procedure CheckingAdvGridCustomCellBkgDraw(Sender: TObject;
Canvas: TCanvas;
      ACol, ARow: Integer; AState: TGridDrawState; ARect: TRect;
      Printing: Boolean);
    procedure CheckingAdvGridDblClickCell(Sender: TObject; ARow,
ACol: Integer);
    procedure AddCheckingButtonClick(Sender: TObject);
    procedure PrintCheckingButtonClick(Sender: TObject);
    procedure CalibrationAdvGridCustomCellBkgDraw(Sender: TObject;
      Canvas: TCanvas; ACol, ARow: Integer; AState: TGridDrawState;
      ARect: TRect; Printing: Boolean);
    procedure CalibrationAdvGridDblClickCell(Sender: TObject; ARow,
      ACol: Integer);
    procedure FormShow(Sender: TObject);
    procedure PrintCalibrationButtonClick(Sender: TObject);
    procedure AddCalibrationButtonClick(Sender: TObject);
    procedure DisplayCalibrationComboBoxChange(Sender: TObject);
    procedure DisplayCheckingComboBoxChange(Sender: TObject);
    procedure DBAdvGridFixedCellClick(Sender: TObject; ACol, ARow:
Integer);
    procedure CheckingAdvGridFixedCellClick(Sender: TObject; ACol,
      ARow: Integer);
    procedure CalibrationAdvGridFixedCellClick(Sender: TObject;
ACol,
      ARow: Integer);
    procedure FormCreate(Sender: TObject);
    procedure RepairPageSelectorClick(Sender: TObject);
    procedure RepairAdvGridFixedCellClick(Sender: TObject; ACol,
ARow: Integer);
    procedure AddRepairButtonClick(Sender: TObject);
    procedure RepairAdvGridDblClickCell(Sender: TObject; ARow, ACol:
Integer);
    procedure RepairPrintRepairButtonClick(Sender: TObject);
    procedure frxRepairUserDataSetFirst(Sender: TObject);
    procedure frxRepairUserDataSetCheckEOF(Sender: TObject; var Eof:
Boolean);
    procedure frxRepairUserDataSetNext(Sender: TObject);
    procedure frxRepairUserDataSetPrior(Sender: TObject);
    procedure frxRepairReportBeginDoc(Sender: TObject);
    procedure RepairPrintCheckingPlanButtonClick(Sender: TObject);
    procedure frxCheckingUserDataSetGetValue(const VarName: string;
      var Value: Variant);
    procedure frxRepairReportEndDoc(Sender: TObject);
    procedure CalibrationPrintRepairButtonClick(Sender: TObject);
    procedure CheckingPrintRepairButtonClick(Sender: TObject);
    procedure ListPrintRepairButtonClick(Sender: TObject);
    procedure CalibrationPrintCheckingPlanButtonClick(Sender:
TObject);
    procedure CheckingPrintCheckingPlanButtonClick(Sender: TObject);
    procedure ListPrintCheckingPlanButtonClick(Sender: TObject);
    procedure frxRepairUserDataSetGetValue(const VarName: string;
```

```
      var Value: Variant);
    procedure UpdateRepairButtonClick(Sender: TObject);
    procedure RepairAdvGridSelectionChanged(Sender: TObject; ALeft,
ATop,
      ARight, ABottom: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

uses DMUnit, DBItemUnit, CheckingUnit, CalibrationUnit, RepairUnit;

function GetCheckingDate(FieldData: string; Year: string): string;
var
  List: TStringList;
  n: Integer;
  FindStr: string;
begin
  if (FieldData = '') then
    exit;

  FindStr := '';

  List := TStringList.Create;

  List.Delimiter := ',';
  List.DelimitedText := FieldData;

  for n := 0 to List.Count - 1 do
    if Pos(Year, List[ n ]) > 0 then begin
      FindStr := List[ n ];
      Break;
    end;

  List.Free;

  Result := FindStr;

  if (FindStr <> '') then begin
    List := TStringList.Create;

    List.Delimiter := '.';
    List.DelimitedText := FindStr;

    FindStr := List[ 0 ] + '.' + List[ 1 ];
    List.Free;

    Result := FindStr;
  end;
```

```
end;

function GetRepairDate(FieldData, EndFieldData: string; Number:
integer): string;
var
  List: TStringList;
  n: Integer;
  FindStr, FindStr2: string;
begin
  if (FieldData = '') then
    exit;

  FindStr := '';

  List := TStringList.Create;

  List.Delimiter := ',';
  List.DelimitedText := FieldData;

  if (List.Count - 1 >= Number) then
    FindStr := List[ Number ];

  List.Free;

  if (FindStr <> '') then begin
    List := TStringList.Create;

    List.Delimiter := '-';
    List.DelimitedText := FindStr;

    if List.Count = 3 then
      FindStr := List[ 2 ] + '.' + List[ 1 ] + '.' + List[ 0 ];

    List.Free;

    FindStr2 := '';
    List := TStringList.Create;

    List.Delimiter := ',';
    List.DelimitedText := EndFieldData;

    if (List.Count - 1 >= Number) then
      FindStr2 := #10#13 + List[ Number ];

    List.Free;

    if FindStr2 <> '' then begin
      List := TStringList.Create;

      List.Delimiter := '-';
      List.DelimitedText := FindStr2;

      if List.Count = 3 then
        FindStr2 := List[ 2 ] + '.' + List[ 1 ] + '.' + List[ 0 ];

      List.Free;
    end;
```

```
    Result := FindStr + #10 + FindStr2;
  end;
end;

procedure TMainForm.AddCalibrationButtonClick(Sender: TObject);
begin
  if (DM.CalibrationQuery.RecordCount > 0) then begin
    if (DM.CalibrationTable.RecordCount > 0) then
      DM.CalibrationTable.Insert;

    DM.CalibrationTable.FieldByName('DEVICE_ID').AsInteger :=
DM.CalibrationQuery.FieldByName('ID').AsInteger;

    CalibrationForm.DisplayType := DISPLAY_TYPE_ADD;
    if CalibrationForm.ShowModal = mrOk then begin
      DM.CalibrationTable.Post;
      DM.CalibrationQuery.Refresh;
    end
    else
      DM.CalibrationTable.Cancel;
  end;
end;

procedure TMainForm.AddCheckingButtonClick(Sender: TObject);
begin
  if (DM.CheckingQuery.RecordCount > 0) then begin
    if (DM.CheckingTable.RecordCount > 0) then
      DM.CheckingTable.Insert;

    DM.CheckingTable.FieldByName('DEVICE_ID').AsInteger :=
DM.CheckingQuery.FieldByName('ID').AsInteger;

    CheckingForm.DisplayType := DISPLAY_TYPE_ADD;
    if CheckingForm.ShowModal = mrOk then begin
      DM.CheckingTable.Post;
      DM.CheckingQuery.Refresh;
    end
    else
      DM.CheckingTable.Cancel;
  end;
end;

procedure TMainForm.AddDBButtonClick(Sender: TObject);
begin
  DM.DevicesQuery.ReadOnly := False;

  DM.DevicesQuery.Insert;

  DBItemForm.DisplayType := DISPLAY_TYPE_ADD;
  if DBItemForm.ShowModal = mrOk then
    DM.DevicesQuery.Post
  else
    DM.DevicesQuery.Cancel;

  DM.DevicesQuery.ReadOnly := True;
end;
```

```
procedure TMainForm.AddRepairButtonClick(Sender: TObject);
begin
  if (DM.RepairQuery.RecordCount > 0) then begin
    if (DM.RepairTable.RecordCount > 0) then
      DM.RepairTable.Insert;

    DM.RepairTable.FieldByName('DEVICE_ID').AsInteger :=
DM.RepairQuery.FieldByName('ID').AsInteger;

    RepairForm.DisplayType := DISPLAY_TYPE_ADD;
    if RepairForm.ShowModal = mrOk then begin
      DM.RepairTable.Post;
      DM.RepairQuery.Refresh;
      RepairAdvGridSelectionChanged(Self, 0, 0, 0, 0);
    end
    else
      DM.RepairTable.Cancel;
  end;
end;

procedure TMainForm.CalibrationAdvGridCustomCellBkgDraw(Sender:
TObject;
  Canvas: TCanvas; ACol, ARow: Integer; AState: TGridDrawState;
ARect: TRect;
  Printing: Boolean);
var
  Draw: boolean;
  Value: string;
  CurDate: TDate;
  CellColor: TColor;
begin
  if (ACol = 4) and (ARow > 1) then begin
    Draw := False;
    CellColor := RGB(227, 157, 157);

    Value := CalibrationAdvGrid.Cells[ ACol, ARow ];

    if (Value = '') then
      Draw := True
    else begin
      CurDate := StrToDate(Value);
      if (CurDate < Date) then begin
        Draw := True;
      end;

      if (DaysBetween(Date, CurDate) < 90) and (CurDate > Date) then
begin
          CellColor := RGB(245, 214, 214);
          Draw := True;
      end;
    end;

    if Draw then begin
      Canvas.Brush.Color := CellColor;
      Canvas.Pen.Color := clWhite;
      Canvas.FillRect(ARect);
```

```
    end;
  end;
end;

procedure TMainForm.CalibrationAdvGridDblClickCell(Sender: TObject;
ARow,
  ACol: Integer);
begin
  if (ARow > 1) then
    if not DM.CalibrationQuery.FieldByName('C_ID').IsNull then begin
      DM.CalibrationTable.Locate('ID',
DM.CalibrationQuery.FieldByName('C_ID').AsInteger, []);

      CalibrationForm.DisplayType := DISPLAY_TYPE_VIEW;
      CalibrationForm.ShowModal;
    end;
end;

procedure TMainForm.CalibrationAdvGridFixedCellClick(Sender:
TObject; ACol,
  ARow: Integer);
var
  SortField, SQL, s: string;
  i: Integer;
begin
  if (ARow <> 1) then
    Exit;

  SortField := '';

  case ACol of
    0: SortField := 'ORDER BY DEVICES.ID';
    1: SortField := 'ORDER BY DEVICES.NAME';
    2: SortField := 'ORDER BY DEVICES.TYPE';
    3: SortField := 'ORDER BY DEVICES.DEPARTMENT';
    4: SortField := 'ORDER BY c.DATE';
    5: SortField := 'ORDER BY c.END_DATE';
  end;

  SQL := DMUnit.CalibrationQuerySQL + SortField;

  if (SortField <> '') and (SortField = DM.CalibrationQuery.SQL[
DM.CalibrationQuery.SQL.Count - 1 ]) then
    SQL := SQL + ' DESC';

  SQL := SQL + #10;

  DM.CalibrationQuery.SQL.Text := SQL;
  DM.CalibrationQuery.Active := False;
  DM.CalibrationQuery.Active := True;
end;

procedure TMainForm.CalibrationPageSelectorClick(Sender: TObject);
begin
  Pager.PageIndex := 2;
end;
```

```
procedure TMainForm.CheckingAdvGridCustomCellBkgDraw(Sender:
TObject;
  Canvas: TCanvas; ACol, ARow: Integer; AState: TGridDrawState;
ARect: TRect;
  Printing: Boolean);
var
  Draw: boolean;
  Value: string;
  CurDate: TDate;
  CellColor: TColor;
begin
  if (ACol = 4) and (ARow > 1) then begin
    Draw := False;
    CellColor := RGB(227, 157, 157);

    Value := CheckingAdvGrid.Cells[ ACol, ARow ];

    if (Value = '') then
      Draw := True
    else begin
      CurDate := StrToDate(Value);
      if (CurDate < Date) then begin
        Draw := True;
      end;

      if (DaysBetween(Date, CurDate) < 90) and (CurDate > Date) then
begin
          CellColor := RGB(245, 214, 214);
          Draw := True;
      end;
    end;

    if Draw then begin
      Canvas.Brush.Color := CellColor;
      Canvas.Pen.Color := clWhite;
      Canvas.FillRect(ARect);
    end;
  end;
end;

procedure TMainForm.CheckingAdvGridDblClickCell(Sender: TObject;
ARow,
  ACol: Integer);
begin
  if (ARow > 1) then
    if not DM.CheckingQuery.FieldByName('C_ID').IsNull then begin
      DM.CheckingTable.Locate('ID',
DM.CheckingQuery.FieldByName('C_ID').AsInteger, []);

      CheckingForm.DisplayType := DISPLAY_TYPE_VIEW;
      CheckingForm.ShowModal;
    end;
end;

procedure TMainForm.CheckingAdvGridFixedCellClick(Sender: TObject;
ACol,
  ARow: Integer);
```

```
var
  SortField, SQL, s: string;
  i: Integer;
begin
  if (ARow <> 1) then
    Exit;

  SortField := '';

  case ACol of
    0: SortField := 'ORDER BY DEVICES.ID';
    1: SortField := 'ORDER BY DEVICES.NAME';
    2: SortField := 'ORDER BY DEVICES.TYPE';
    3: SortField := 'ORDER BY DEVICES.DEPARTMENT';
    4: SortField := 'ORDER BY c.DATE';
    5: SortField := 'ORDER BY c.END_DATE';
  end;

  SQL := DMUnit.CheckingQuerySQL + SortField;

  if (SortField <> '') and (SortField = DM.CheckingQuery.SQL[
DM.CheckingQuery.SQL.Count - 1 ]) then
    SQL := SQL + ' DESC';

  SQL := SQL + #10;

  DM.CheckingQuery.SQL.Text := SQL;
  DM.CheckingQuery.Active := False;
  DM.CheckingQuery.Active := True;
end;

procedure TMainForm.CheckingPageSelectorClick(Sender: TObject);
begin
  Pager.PageIndex := 1;
end;

procedure TMainForm.CheckingPrintCheckingPlanButtonClick(Sender:
TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to CheckingAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + CheckingAdvGrid.Columns[
CheckingAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(CheckingAdvGrid.Filter[ i ].Condition, '*', '%',
[rfReplaceAll, rfIgnoreCase]) + '")';
  end;

  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
```

```
    DM.RepairQuery.Active := True;
  end;

  DM.RepairQDS.DataSet.DisableControls;
  frxCheckingReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxCheckingReport.ShowPreparedReport;
end;

procedure TMainForm.CheckingPrintRepairButtonClick(Sender: TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to CheckingAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + CheckingAdvGrid.Columns[
CheckingAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(CheckingAdvGrid.Filter[ i ].Condition, '*', '%',
[rfReplaceAll, rfIgnoreCase]) + '")';
  end;

  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
    DM.RepairQuery.Active := True;
  end;

  DM.RepairQDS.DataSet.DisableControls;
  frxRepairReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxRepairReport.ShowPreparedReport;
end;

procedure TMainForm.CalibrationPrintCheckingPlanButtonClick(Sender:
TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to CalibrationAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + CalibrationAdvGrid.Columns[
CalibrationAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(CalibrationAdvGrid.Filter[ i ].Condition, '*',
'%', [rfReplaceAll, rfIgnoreCase]) + '")';
  end;
```

```
  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
    DM.RepairQuery.Active := True;
  end;

  DM.RepairQDS.DataSet.DisableControls;
  frxCheckingReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxCheckingReport.ShowPreparedReport;
end;

procedure TMainForm.CalibrationPrintRepairButtonClick(Sender:
TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to CalibrationAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + CalibrationAdvGrid.Columns[
CalibrationAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(CalibrationAdvGrid.Filter[ i ].Condition, '*',
'%', [rfReplaceAll, rfIgnoreCase]) + '")';
  end;

  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
    DM.RepairQuery.Active := True;
  end;

  DM.RepairQDS.DataSet.DisableControls;
  frxRepairReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxRepairReport.ShowPreparedReport;
end;

procedure TMainForm.DBAdvGridDblClickCell(Sender: TObject; ARow,
ACol: Integer);
begin
  if (ARow > 1) then begin
    DBItemForm.DisplayType := DISPLAY_TYPE_VIEW;
    DBItemForm.ShowModal;
  end;
end;

procedure TMainForm.DBAdvGridFixedCellClick(Sender: TObject; ACol,
  ARow: Integer);
var
```

```
  SortField, SQL, s: string;
  i: Integer;
begin
  if (ARow <> 1) then
    Exit;

  SortField := '';

  case ACol of
    0: SortField := 'ORDER BY DEVICES.ID';
    1: SortField := 'ORDER BY DEVICES.NAME';
    2: SortField := 'ORDER BY DEVICES.TYPE';
    3: SortField := 'ORDER BY MEASURE_TYPE.NAME';
    4: SortField := 'ORDER BY DEVICES.DEPARTMENT';
    5: SortField := 'ORDER BY DEVICES.SERIAL_NUMBER';
    6: SortField := 'ORDER BY DEVICES.CREATE_DATE';
    7: SortField := 'ORDER BY DEVICES.INVENTORY_NUMBER';
  end;

  SQL := DMUnit.DevicesSQL + SortField;

  if (SortField <> '') and (SortField = DM.DevicesQuery.SQL[
DM.DevicesQuery.SQL.Count - 1 ]) then
    SQL := SQL + ' DESC';

  SQL := SQL + #10;

  DM.DevicesQuery.SQL.Text := SQL;
  DM.DevicesQuery.Active := False;
  DM.DevicesQuery.Active := True;
end;

procedure TMainForm.DBPageSelectorClick(Sender: TObject);
begin
  Pager.PageIndex := 0;
end;

procedure TMainForm.DeleteDBButtonClick(Sender: TObject);
begin
  if (MessageDlg('Вы действительно хотите удалить выбранное средство
измерения?', mtConfirmation, [mbYes, mbNo ], 0) = ID_YES) then begin
    DM.DevicesQuery.ReadOnly := False;

    DM.DevicesQuery.Delete;

    DM.DevicesQuery.ReadOnly := True;
  end;
end;

procedure TMainForm.DisplayCalibrationComboBoxChange(Sender:
TObject);
begin
  case DisplayCalibrationComboBox.ItemIndex of
    0: begin
      DM.CalibrationQuery.FilterSQL := '';
      DM.CalibrationQuery.Refresh;
    end;
```

```
    1: begin
      DM.CalibrationQuery.FilterSQL := '(julianday(c.END_DATE) -
julianday(DATE()) < 90) and (julianday(c.END_DATE) -
julianday(DATE()) > 0)';
      DM.CalibrationQuery.Refresh;
    end;
    2: begin
      DM.CalibrationQuery.FilterSQL := 'END_DATE < DATE()';
      DM.CalibrationQuery.Refresh;
    end;
    3: begin
      DM.CalibrationQuery.FilterSQL := 'END_DATE IS NULL';
      DM.CalibrationQuery.Refresh;
    end;
  end;
end;

procedure TMainForm.DisplayCheckingComboBoxChange(Sender: TObject);
begin
  case DisplayCheckingComboBox.ItemIndex of
    0: begin
      DM.CheckingQuery.FilterSQL := '';
      DM.CheckingQuery.Refresh;
    end;
    1: begin
      DM.CheckingQuery.FilterSQL := '(julianday(c.END_DATE) -
julianday(DATE()) < 90) and (julianday(c.END_DATE) -
julianday(DATE()) > 0)';
      DM.CheckingQuery.Refresh;
    end;
    2: begin
      DM.CheckingQuery.FilterSQL := 'END_DATE < DATE()';
      DM.CheckingQuery.Refresh;
    end;
    3: begin
      DM.CheckingQuery.FilterSQL := 'END_DATE IS NULL';
      DM.CheckingQuery.Refresh;
    end;
  end;
end;

procedure TMainForm.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  DM.Database.Connected := False;
end;

procedure TMainForm.FormCreate(Sender: TObject);
var
  Value: string;
begin
  Value := InputBox('Пароль', #31'Введите пароль:', '');

  if Value <> 'Nikulin1004' then
    Halt(0);
end;
```

```
procedure TMainForm.FormShow(Sender: TObject);
begin
  Pager.PageIndex := 0;
  RepairAdvGridSelectionChanged(Self, 0, 0, 0, 0);
end;

procedure TMainForm.frxCheckingUserDataSetGetValue(const VarName:
string;
  var Value: Variant);
begin
  if (VarName = 'NAME') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'NAME' ])
  else if (VarName = 'DEPARTMENT') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'DEPARTMENT'
])
  else if (VarName = 'COUNT') then
    Value := 1
  else if (VarName = 'ACCURACY_CLASS') then begin
    if VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'ACCURACY_CLASS'
]) <> '' then
      Value := 'Класс ' + VarToStr(DM.RepairQDS.DataSet.FieldValues[
'ACCURACY_CLASS' ])
  end
  else if (VarName = 'TYPE') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'TYPE' ])
  else if (VarName = 'MEASURE_ERROR') then begin
    if VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'MEASURE_ERROR' ])
<> '' then
      Value := 'погр. ' + VarToStr(DM.RepairQDS.DataSet.FieldValues[
'MEASURE_ERROR' ])
  end
  else if VarName = 'MEASURE_TYPE_NAME' then begin
    if VarToStr(DM.RepairQDS.DataSet.FieldValues[
'MEASURE_TYPE_NAME' ]) <> '' then
      Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[
'MEASURE_TYPE_NAME' ])
    else
      Value := 'Другие виды измерений';
  end;
end;

procedure TMainForm.frxRepairReportBeginDoc(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 4 do
    TfrxMemoView(frxRepairReport.FindObject('Memo' + IntToStr(14 +
i))).Text := IntToStr(YearOf(Date) + i - 4) + ' г.';
end;

procedure TMainForm.frxRepairReportEndDoc(Sender: TObject);
begin
  DM.RepairQuery.SQL[ 5 ] := '';
  DM.RepairQuery.SQL[ 6 ] := 'GROUP BY DEVICES.ID';
  DM.RepairQuery.Active := False;
  DM.RepairQuery.Active := True;
end;
```

```
procedure TMainForm.frxRepairUserDataSetCheckEOF(Sender: TObject;
var Eof: Boolean);
begin
  Eof := DM.RepairQDS.DataSet.Eof;
end;

procedure TMainForm.frxRepairUserDataSetFirst(Sender: TObject);
begin
  DM.RepairQDS.DataSet.First;
end;

procedure TMainForm.frxRepairUserDataSetGetValue(const VarName:
string;
  var Value: Variant);
begin
  if (VarName = 'NAME') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'NAME' ])
  else if (VarName = 'CREATE_DATE') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'TIMESTAMP'
])
  else if (VarName = 'TYPE') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'TYPE' ])
  else if (VarName = 'LIMIT') then begin
    if ((VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'LOWER_LIMIT' ])
<> '') OR (VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'UPPER_LIMIT'
]) <> '')) then
      Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[
'LOWER_LIMIT' ]) + '..' + VarToStr(DM.RepairQDS.DataSet.FieldValues[
'UPPER_LIMIT' ]);
  end
  else if (VarName = 'SERIAL_NUMBER') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[
'SERIAL_NUMBER' ])
  else if (VarName = 'INSTALL_PLACE') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[
'INSTALL_PLACE' ])
  else if (VarName = 'DEPARTMENT') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'DEPARTMENT'
])
  else if (VarName = 'CREATE_YEAR') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CREATE_YEAR' ])
  else if (VarName = 'COMMENTS') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[ 'COMMENTS'
])
  else if (VarName = 'CHECKING_PERIOD') then
    Value := VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CHECKING_PERIOD' ])
  else if (VarName = 'CHECKING_DATE_1') then
    Value :=
GetCheckingDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CHECKING_DATE' ]), IntToStr(YearOf(Date) - 4))
  else if (VarName = 'CHECKING_DATE_2') then
    Value :=
GetCheckingDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CHECKING_DATE' ]), IntToStr(YearOf(Date) - 3))
```

```
  else if (VarName = 'CHECKING_DATE_3') then
    Value :=
GetCheckingDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CHECKING_DATE' ]), IntToStr(YearOf(Date) - 2))
  else if (VarName = 'CHECKING_DATE_4') then
    Value :=
GetCheckingDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CHECKING_DATE' ]), IntToStr(YearOf(Date) - 1))
  else if (VarName = 'CHECKING_DATE_5') then
    Value :=
GetCheckingDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'CHECKING_DATE' ]), IntToStr(YearOf(Date)))
  else if (VarName = 'REPAIR_DATE_1') then
    Value :=
GetRepairDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'REPAIR_DATE' ]), VarToStr(DM.RepairQDS.DataSet.FieldValues[
'REPAIR_END_DATE' ]), 0)
  else if (VarName = 'REPAIR_DATE_2') then
    Value :=
GetRepairDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'REPAIR_DATE' ]), VarToStr(DM.RepairQDS.DataSet.FieldValues[
'REPAIR_END_DATE' ]), 1)
  else if (VarName = 'REPAIR_DATE_3') then
    Value :=
GetRepairDate(VarToStr(DM.RepairQDS.DataSet.FieldValues[
'REPAIR_DATE' ]), VarToStr(DM.RepairQDS.DataSet.FieldValues[
'REPAIR_END_DATE' ]), 2)
end;

procedure TMainForm.frxRepairUserDataSetNext(Sender: TObject);
begin
  DM.RepairQDS.DataSet.Next;
end;

procedure TMainForm.frxRepairUserDataSetPrior(Sender: TObject);
begin
  DM.RepairQDS.DataSet.Prior;
end;

procedure TMainForm.ListPrintCheckingPlanButtonClick(Sender:
TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to DBAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + DBAdvGrid.Columns[
DBAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(DBAdvGrid.Filter[ i ].Condition, '*', '%',
[rfReplaceAll, rfIgnoreCase]) + '")';
  end;
```

```
  DM.RepairQuery.Active := False;
  DM.RepairQuery.SQL[ 6 ] := 'GROUP BY DEVICES.ID ORDER BY
DEVICES.MEASURE_TYPE_ID ASC';

  if (FilterStr <> '') then
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;

  DM.RepairQuery.Active := True;

  DM.RepairQDS.DataSet.DisableControls;
  frxCheckingReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxCheckingReport.ShowPreparedReport;
end;

procedure TMainForm.ListPrintRepairButtonClick(Sender: TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to DBAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + DBAdvGrid.Columns[
DBAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(DBAdvGrid.Filter[ i ].Condition, '*', '%',
[rfReplaceAll, rfIgnoreCase]) + '")';
  end;

  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
    DM.RepairQuery.Active := True;
  end;

  DM.RepairQDS.DataSet.DisableControls;
  frxRepairReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxRepairReport.ShowPreparedReport;
end;

procedure TMainForm.PrintCalibrationButtonClick(Sender: TObject);
begin
  CalibrationAdvGrid.Print;
end;

procedure TMainForm.PrintCheckingButtonClick(Sender: TObject);
begin
  CheckingAdvGrid.Print;
end;

procedure TMainForm.RepairPrintCheckingPlanButtonClick(Sender:
```

```
TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to RepairAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + RepairAdvGrid.Columns[
RepairAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(RepairAdvGrid.Filter[ i ].Condition, '*', '%',
[rfReplaceAll, rfIgnoreCase]) + '")';
  end;

  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
    DM.RepairQuery.Active := True;
  end;

  DM.RepairQDS.DataSet.DisableControls;
  frxCheckingReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxCheckingReport.ShowPreparedReport;
end;

procedure TMainForm.PrintDBButtonClick(Sender: TObject);
begin
  DBAdvGrid.Print;
end;

procedure TMainForm.RepairPrintRepairButtonClick(Sender: TObject);
var
  i: integer;
  FilterStr: string;
begin
  FilterStr := '';

  for i := 0 to RepairAdvGrid.Filter.Count - 1 do begin
    if FilterStr <> '' then
      FilterStr := FilterStr + ' AND ';

    FilterStr := FilterStr + '(' + RepairAdvGrid.Columns[
RepairAdvGrid.Filter[ i ].Column ].Name + ' LIKE "' +
      StringReplace(RepairAdvGrid.Filter[ i ].Condition, '*', '%',
[rfReplaceAll, rfIgnoreCase]) + '")';
  end;

  if (FilterStr <> '') then begin
    DM.RepairQuery.SQL[ 5 ] := 'WHERE ' + FilterStr;
    DM.RepairQuery.Active := False;
    DM.RepairQuery.Active := True;
  end;
```

```
  DM.RepairQDS.DataSet.DisableControls;
  frxRepairReport.PrepareReport();
  DM.RepairQDS.DataSet.EnableControls;

  frxRepairReport.ShowPreparedReport;
end;

procedure TMainForm.RepairAdvGridDblClickCell(Sender: TObject; ARow,
  ACol: Integer);
begin
  if (ARow > 1) then
    if not DM.RepairQuery.FieldByName('C_ID').IsNull then begin
      DM.RepairTable.Locate('ID',
DM.RepairQuery.FieldByName('C_ID').AsInteger, []);

      RepairForm.DisplayType := DISPLAY_TYPE_VIEW;
      RepairForm.ShowModal;
    end;
end;

procedure TMainForm.RepairAdvGridFixedCellClick(Sender: TObject;
ACol,
  ARow: Integer);
var
  SortField, SQL, s: string;
  i: Integer;
begin
  if (ARow <> 1) then
    Exit;

  SortField := '';

  case ACol of
    0: SortField := 'ORDER BY DEVICES.ID';
    1: SortField := 'ORDER BY DEVICES.NAME';
    2: SortField := 'ORDER BY DEVICES.TYPE';
    3: SortField := 'ORDER BY DEVICES.DEPARTMENT';
    4: SortField := 'ORDER BY c.DATE';
    5: SortField := 'ORDER BY c.REASON';
  end;

  SQL := DMUnit.RepairQuerySQL + SortField;

  if (SortField <> '') and (SortField = DM.RepairQuery.SQL[
DM.RepairQuery.SQL.Count - 1 ]) then
    SQL := SQL + ' DESC';

  SQL := SQL + #10;

  DM.RepairQuery.SQL.Text := SQL;
  DM.RepairQuery.Active := False;
  DM.RepairQuery.Active := True;
end;

procedure TMainForm.RepairAdvGridSelectionChanged(Sender: TObject;
ALeft, ATop,
```

```
  ARight, ABottom: Integer);
begin
  if (DM.RepairQuery.FieldByName('C_ID').AsInteger > 0) then
    UpdateRepairButton.Enabled := True
  else
    UpdateRepairButton.Enabled := False;
end;

procedure TMainForm.RepairPageSelectorClick(Sender: TObject);
begin
  Pager.PageIndex := 3;
end;

procedure TMainForm.UpdateDBButtonClick(Sender: TObject);
begin
  if (DM.DevicesQuery.RecordCount = 0) then
    Exit;

  DM.DevicesQuery.ReadOnly := False;

  DM.DevicesQuery.Edit;

  DBItemForm.DisplayType := DISPLAY_TYPE_UPDATE;
  if DBItemForm.ShowModal = mrOk then begin
    DM.DevicesQuery.Post;
    DM.DevicesDS.DataSet.Refresh;
  end
  else
    DM.DevicesQuery.Cancel;

  DM.DevicesQuery.ReadOnly := True;
end;

procedure TMainForm.UpdateRepairButtonClick(Sender: TObject);
begin
  if (DM.RepairQuery.RecordCount > 0) then begin
    DM.RepairTable.Locate('ID',
DM.RepairQuery.FieldByName('C_ID').AsInteger, []);

    if (DM.RepairTable.RecordCount > 0) then
      DM.RepairTable.Edit;

    RepairForm.DisplayType := DISPLAY_TYPE_UPDATE;
    if RepairForm.ShowModal = mrOk then begin
      DM.RepairTable.Post;
      DM.RepairQuery.Refresh;
    end
    else
      DM.RepairTable.Cancel;
  end;
end;

end.

unit RepairUnit;

interface
```

```
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls,
AdvDateTimePicker,
  AdvDBDateTimePicker, AdvGlowButton, Vcl.ExtCtrls, Vcl.StdCtrls,
AdvEdit,
  DBAdvEd;

type
  TRepairForm = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    PlaceEdit: TDBAdvEdit;
    Panel1: TPanel;
    CancelButton: TAdvGlowButton;
    OKButton: TAdvGlowButton;
    StartDatePicker: TAdvDBDateTimePicker;
    ReasonEdit: TDBAdvEdit;
    EndDatePicker: TAdvDBDateTimePicker;
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure FormShow(Sender: TObject);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    DisplayType: integer;
    procedure SetControlsEnabled(Value: Boolean);
  end;

var
  RepairForm: TRepairForm;

implementation

{$R *.dfm}

uses MainUnit, DMUnit;

procedure TRepairForm.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Ord(Key) = VK_ESCAPE then
    ModalResult := mrClose;
end;

procedure TRepairForm.FormShow(Sender: TObject);
begin
  case DisplayType of
    DISPLAY_TYPE_VIEW: Caption := 'Просмотр данных ремонта';
    DISPLAY_TYPE_ADD: Caption := 'Добавление данных ремонта';
    DISPLAY_TYPE_UPDATE: Caption := 'Изменение данных ремонта';
  end;

  if (DisplayType = DISPLAY_TYPE_ADD) then begin
    StartDatePicker.Date := Now;
    EndDatePicker.Checked := False;
```

```
  end;

  CancelButton.Visible := (DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE);
  OKButton.Align := alLeft;
  OKButton.Align := alRight;
  SetControlsEnabled((DisplayType = DISPLAY_TYPE_ADD) or
(DisplayType = DISPLAY_TYPE_UPDATE));
end;

procedure TRepairForm.OKButtonClick(Sender: TObject);
begin
  if (DisplayType = DISPLAY_TYPE_VIEW) then begin
    ModalResult := mrOk;
    Exit;
  end;

  if (PlaceEdit.Text = '') then begin
    MessageBox(Handle, PChar('Не заполнено поле "Место ремонта".'),
'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  if (ReasonEdit.Text = '') then begin
    MessageBox(Handle, PChar('Не заполнено поле "Причина
ремонта".'), 'Ошибка', MB_ICONEXCLAMATION);
    Exit;
  end;

  ModalResult := mrOk;
end;

procedure TRepairForm.SetControlsEnabled(Value: Boolean);
begin
  PlaceEdit.EditorEnabled := Value;
  ReasonEdit.EditorEnabled := Value;
  StartDatePicker.Enabled := Value;
  EndDatePicker.Enabled := Value;
end;

end.
```

ЛИТЕРАТУРА

Нормативно-правовые акты:

1. Конституция Российской Федерации (принята всенародным голосованием 12 декабря 1993 г. с изменениями, одобренными в ходе общероссийского голосования 01.07.2020). [Электронный ресурс]. – Доступ из справочной правовой системы «КонсультантПлюс».
2. Федеральный закон от 7 февраля 2011 г. № 3-ФЗ «О полиции» [Электронный ресурс]. – Доступ из справочной правовой системы «КонсультантПлюс».
3. Об обеспечении единства измерений : федер. закон от 26 июня 2008 г. №102-ФЗ [Электронный ресурс]. – Доступ из справочной правовой системы «КонсультантПлюс».
4. О техническом регулировании : федер. закон от 27 декабря 2002 г. №184-ФЗ [Электронный ресурс]. – Доступ из справочной правовой системы «КонсультантПлюс».
5. О стандартизации в Российской Федерации : федер. закон от 29 июня 2015 г. №162-ФЗ [Электронный ресурс]. – Доступ из справочной правовой системы «КонсультантПлюс».
6. Об утверждении положения о единицах величин, допускаемых к применению в Российской Федерации : постановление Правительства Рос. Федерации от 31 октября 2009 г. №879 // Собр. законодательства Рос. Федерации. – 2009. – № 45. – Ст. 5352. – С. 4572-4643.
7. О порядке организации метрологического обеспечения Министерства внутренних дел Российской Федерации : приказ МВД Рос. Федерации от 30 января 2014 г. №53 [Электронный ресурс] // URL : http//base.consultant.ru.
8. Об утверждении Перечня измерений, относящихся к сфере государственного регулирования обеспечения единства измерений и обязательных метрологических требований к ним : приказ МВД Рос. Федерации от 08 ноября 2012 г. №1014. – Доступ из справочной правовой системы «КонсультантПлюс».
9. Вопросы метрологической службы Министерства внутренних дел Российской Федерации : приказ МВД Рос. Федерации от 07 мая 2010 г. №355 [Электронный ресурс] // URL : http//base.consultant.ru..
10. РМГ 29-2013. Государственная система обеспечения единства измерений. Метрология. Основные термины и определения. - Введ. 2013-12-05 [Электронный ресурс] // URL : http//base.consultant.ru.

Основная литература:

1. Хромой Б.П. Метрология, стандартизация и сертификация : учебник для вузов / М : Горячая линия - Телеком, 2018. – 432 с.

2. Булгаков О. М. Теоретические основы, методы и техника электрорадио-измерений: учебное пособие / О. М. Булгаков, О. В. Четкин. – Воронеж : Воронежский институт МВД России, 2016. – 157 с.

3. Димов Ю. В. Метрология, стандартизация и сертификация : учебник / Ю. В. Димов. – СПб : Питер, 2013. – 496 с.

4. Метрология и радиоизмерения : учебник / под ред. В. И. Нефедова . – Москва : Высшая школа, 2006. – 526 с.

5. Дворяшин Б. В. Метрология и радиоизмерения : учеб. пособие для вузов / Б. В. Дворяшин. – Москва : Академия, 2005. – 304 с.

6. Сергеев А. Г. Метрология и метрологическое обеспечение : учеб. пособие / А. Г. Сергеев. – Москва : Высш. образ., 2008. – 575 с.

7. Метрология, стандартизация, сертификация и электроизмерительная техника : учеб. пособие / под ред. К. К. Кима . – СПб : Питер, 2008. – 367 с.

8. Основы стандартизации, сертификации, метрологии [Электронный ресурс] : учебник / Г. Д. Крылова. – М. : Юнити-Дана, 2015. – 671 с. – Режим доступа : http : // www.biblioclub.ru/index.php?page=book_red&id=114433&sr=1. – ЭБС «Университетская библиотека ONLINE».

Дополнительная литература:

1. Кузнецов В. А. Общая метрология / В. А. Кузнецов, Г. В. Ялунина. – Москва : ИПК Издательство стандартов, 2001. – 272 с.

2. Измерения в электронике: Справочник / В. А. Кузнецов [и др.]; под ред. В. А. Кузнецова. – Москва : Энергоатомиздат, 1987. – 512 с.

3. Мейзда Ф. Электронные измерительные приборы и методы измерений: пер. с англ. / Ф. Мейзда. – Москва : Мир, 1990. – 535 с.

4. Клаассен К. Б. Основы измерений. Электронные методы и приборы в измерительной технике : пер. с нем. / К. Б. Клаасен. – Москва : Постмаркет, 2000. – 352 с.

5. Харт Х. Введение в измерительную технику : пер. с нем. – Москва : Мир, 1999. – 391 с.

6. Мирский Г. Я. Электронные измерения / Г. Я. Мирский. – Москва : Радио и связь, 1986. – 440 с.

7. Винокуров В. И. Электрорадиоизмерения: учебное пособие для вузов / В. И. Винокуров, С. И. Каплин, И. Г. Петелин; под ред. В. И. Винокурова. – Москва : Высшая школа, 1986. – 351 с.

8. Метрология, стандартизация и измерения в технике связи / Б. П. Хромой [и др.]; под ред. Б. П. Хромого. – Москва : Радио и связь, 1986. – 268 с.

10. Удалов В. П. Сборник нормативных актов по деятельности метрологической службы органов внутренних дел МВД России / В. П.

Удалов, М. М. Жуков. – Воронеж : Воронежский институт МВД России, 2014. – 152 с.

11. Удалов В. П. Метрологическое обеспечение деятельности органов внутренних дел и внутренних войск МВД России : учебно-методическое пособие / В. П. Удалов, С. С. Никулин, С. Л. Анисимов. - Воронеж: Воронежский институт МВД России, 2013. – 58 с.

12. Удалов В. П. Метрологическое обеспечение деятельности органов внутренних дел и внутренних войск МВД России : практикум для курсов повышения квалификации специалистов метрологов МВД России [Электронный ресурс] / В. П. Удалов, С. С. Никулин, С. Л. Анисимов. - Воронеж: Воронежский институт МВД России, 2015. – 108 с.

13. Удалов В. П. Метрологическое обеспечение деятельности МВД России: учебно-методическое пособие [Электронный ресурс] / В. П. Удалов, С. С. Никулин. – Воронеж : Воронежский институт МВД России, 2018.

14. Жуков М. М. Особенности организации обеспечения единства измерений в Министерстве внутренних дел Российской Федерации: методические рекомендации / М. М. Жуков, С. С. Никулин, В. П. Удалов, О. В. Четкин, И. В. Лазарев, В. И. Замятин, А. А. Чернавин. – Воронеж: Воронежский институт МВД России, 2020. – 78 с.

15. Жуков М. М. Порядок работы с анализатором паров этанола в выдыхаемом воздухе «Алкотектор» в исполнении «Юпитер» : учебно-методическое пособие [Электронный ресурс] / М. М. Жуков, В. П. Удалов, В. И. Кудряш. – Воронеж : Воронежский институт МВД России, 2020.

16. Жуков М. М. Порядок работы с установкой для поверки каналов измерения давления и частоты пульса УПКД-3: учебно-методическое пособие [Электронный ресурс] / М. М. Жуков, В. П. Удалов, С. С. Никулин – Воронеж : Воронежский институт МВД России, 2020.