```
unit DataEntry;

{$mode objfpc}{$H+}

//============================================================================================
//
//  DataEntry.pas
//
//  Calls: AppTypes
//         Constants
//         DataEntry_FAV : DataEntry_FAV_Init
//                         DataEntry_Fave_Save
//         DataEntry_UHFMEM : DataEntry_UHFMEM_Init
//                            DataEntry_UHFMEM_Save
//         DataEntry_VHFMEM : DataEntry_VHFMEM_Init
//                            DataEntry_VHFMEM_Save
//         Main
//         Utilities : GetToneNrFromIndex
//                     GetToneIndexFromToneNr
//                     ValidVHFFrequency
//
//         Variables
//
//  Called By: DataEntry_FAV : DataEntry_FAV_Init
//                             DataEntry_FAV_Save
//             DataEntry_MEM : DataEntry_MEM_Init
//             Main : ProcessFavButton
//             Mem : TfrmMEM.bbtAddClick
//                   TfrmMEM.bbtEditClick
//             TMVFiles : WriteTMVFile
//
//  Ver: 1.0.0
//
//  Date: 8 Dec 2013
//
//============================================================================================

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,
  StdCtrls, Buttons,
  // Application Units
  AppConstants, AppTypes, AppVariables, DataEntry_FAV, DataEntry_UHFMEM, DataEntry_VHFMEM,
  Utilities;

type

  TfrmDataEntry = class(TForm)
    bbtSave: TBitBtn;
    bbtClear: TBitBtn;
    bbtReset: TBitBtn;
    bbtCancel: TBitBtn;
    chkScan: TCheckBox;
    chkDTSS: TCheckBox;
    cbxTones: TComboBox;
    edtComments: TEdit;
    edtChannelName: TEdit;
    edtSource: TEdit;
    edtRXFrequency: TEdit;
    edtTXFrequency: TEdit;
    edtDTSSCode: TEdit;
    edtOffsetShift: TEdit;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    GroupBox4: TGroupBox;
    GroupBox5: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label8: TLabel;
    rbtRFPowerHigh: TRadioButton;
    rbtCTCSS: TRadioButton;
    rbtTone: TRadioButton;
    rbtNoTones: TRadioButton;
    rbtRFPowerLow: TRadioButton;
    rbtRFPowerMedium: TRadioButton;
    rbtSimplex: TRadioButton;
    rbtUHF: TRadioButton;
    rbtVHF: TRadioButton;
    rbtMinus: TRadioButton;
    rbtPlus: TRadioButton;
    procedure bbtCancelClick(Sender: TObject);
    procedure bbtClearClick(Sender: TObject);
    procedure bbtResetClick(Sender: TObject);
    procedure bbtSaveClick(Sender: TObject);
    procedure chkDTSSChange(Sender: TObject);
    procedure edtChannelNameKeyPress(Sender: TObject; var Key: char);
    procedure edtCommentsKeyPress(Sender: TObject; var Key: char);
    procedure edtDTSSCodeKeyPress(Sender: TObject; var Key: char);
    procedure edtRXFrequencyExit(Sender: TObject);
    procedure edtRXFrequencyKeyPress(Sender: TObject; var Key: char);
    procedure FormActivate(Sender: TObject);
    procedure rbtMinusChange(Sender: TObject);
    procedure rbtNoTonesChange(Sender: TObject);
    procedure rbtPlusChange(Sender: TObject);
    procedure rbtSimplexChange(Sender: TObject);
    procedure rbtVHFChange(Sender: TObject);
   private
    { private declarations }
  public
    { public declarations }
    const
      cstrFavFormTitle = 'Favourite Button Data Entry';
      cstrMemFormTitle = 'Memory Channel Data Entry';

    var
      vdetDataEntryType : TDataEntryType;
      vbytChannelNumber : Byte;

    function CalculateTXFrequency : string;
    procedure DisableDTSSCode;
    procedure EnableDTSSCode;
    procedure SetDTSSCode;
    procedure SetShiftOffset;

  end;

var
  frmDataEntry: TfrmDataEntry;

implementation

{$R *.lfm}

uses
  Main;

const

  cstrInvalidChannelNameMsg = '         Invalid Channel Name' +
                              #13 +
                              'The Channel is mandatory and must contain' +
                              #13 +
                              '          5 to 15 characters';

   cstrInvalidDTSSCodeMsg = '          Invalid DTSS Code' +
                           #13 +
                           'The DTSS Code must be in the format nnn' +
                           #13 +
                           '        and between 000 and 999'
                           ;
  cstrInvalidVHFFrequencyMsg = '         Invalid VHF RX Frequency.' +
                               #13 +
                               'The frequency must be in the format nnn.nnn' +
                               #13 +
                               '      and between 118.000 and 173.999';

  cstrInvalidUHFFrequencyMsg = '         Invalid UHF RX Frequency' +
                               #13 +
                               'The frequency must be in the format nnn.nnn' +
                               #13 +
                               '      and between 400.000 and 469.999';

  cstrResetMsg = '  Conifirm that you wish to Reset the form.' +
                 #13 +
                 '  This action will reset all data fields to' +
                 #13 +
                 'their original values when the form was opened.';

    cstrCancelMsg = '  Conifirm that you wish to Cancel this entry.' +
                    #13 +
                    'This action will simply close the Data Entry form' +
                    #13 +
                    '   and make no changes to the original data.';

    cstrClearMsg = '  Conifirm that you wish to Clear this entry.' +
                    #13 +
                    'This action will reset all data entry fields' +
                    #13 +
                    '         to their default valuse.';

    cstrContinueMsg = ' Do you want to continue ?';

var

```

```
  vblnValidData : Boolean;

//==========================================================================
//     SUPPORT ROUTINES
//==========================================================================
function TfrmDataEntry.CalculateTXFrequency : string;

var
  vsngTXFrequency : Single;

begin

  if Length(frmDataEntry.edtRXFrequency.Text) > 0 then
  begin

    vsngTXFrequency := StrToFLoat(frmDataEntry.edtRXFrequency.Text);

    // Calculate based on Band (UHF or VHF) and Shift (Simplex, Plus or Minus)
    if frmDataEntry.rbtVHF.Checked then
    begin
      if frmDataEntry.rbtPlus.Checked then
        vsngTXFrequency := vsngTXFrequency + STrToFloat(gcstrVHFShiftOffset)
      else
        vsngTXFrequency := vsngTXFrequency - STrToFloat(gcstrVHFShiftOffset);
    end
    else
    begin
    if frmDataEntry.rbtPlus.Checked then
      vsngTXFrequency := vsngTXFrequency + STrToFloat(gcstrUHFShiftOffset)
    else
      vsngTXFrequency := vsngTXFrequency - STrToFloat(gcstrUHFShiftOffset);
    end;// if frmDataEntry.rbtVHF.Checked

    Result := Format('%-8.3f', [vsngTXFrequency]);
  end
  else
    Result := '';// if Length(frmDataEntry.edtRXFrequency.Text > 0

end;// function CalculateTXFrequency

//--------------------------------------------------------------------------
procedure TfrmDataEntry.SetShiftOffset;
begin

  // This routine does a number things, all related to a band change from VHF to UHF.

  // It first checks to see if there is a Frequency in the RX Frequency edit box.
  // If there is none, the rest doesn't matter and we simply adjust the Shift offset
  // value accordingly and Exit.
  if Length (frmDataEntry.edtRXFrequency.Text) = 0 then
  begin

    if frmDataEntry.rbtVHF.Checked then
    begin
      if frmDataEntry.rbtSimplex.Checked then
        frmDataEntry.edtOffsetShift.Text := gcstrNoShiftOffset
      else
        frmDataEntry.edtOffsetShift.Text := gcstrVHFShiftOffset;
    end
    else
    begin
      if frmDataEntry.rbtSimplex.Checked then
        frmDataEntry.edtOffsetShift.Text := gcstrNoShiftOffset
      else
        frmDataEntry.edtOffsetShift.Text := gcstrUHFShiftOffset;
    end;// if frmDataEntry.rbtVHF.Checked

    Exit;

  end;// if Length (frmDataEntry.edtRXFrequency.Text) = 0

  // There is a frequency in the edit box, so we check the RX Frequency against the
  // new band selection. If the current frequency is out of band, then an error message
  // is displayed and the user is asked if they wish to continue. If they do, both the
  // RX and TX Frequency edit boxes are cleared and the Shift offset is applied
  // according to the Bnad selected. If they do noy wish to continue, the Bnad selection
  // buttons are returned to their original values and no changes are made.
  if frmDataEntry.rbtVHF.Checked then
  begin

    if not ValidVHFFrequency ( frmDataEntry.edtRXFrequency.Text ) then
    begin
      if MessageDlg ( cstrInvalidVHFFrequencyMsg + #13 + cstrContinueMsg, mtError,
                      [mbYes, mbNo],0) = mrNo then
      begin
        frmDataEntry.rbtUHF.Checked := True;
        Exit;
      end
      else
      begin
        frmDataEntry.edtRXFrequency.Text := '';
        frmDataEntry.edtTXFrequency.Text := '';
      end;// if MessageDlg ( cstrInvalidVHFFrequencyMsg
    end;// if not ValidVHFFrequency

  end
  else
  begin

    if not ValidUHFFrequency ( frmDataEntry.edtRXFrequency.Text ) then
    begin
      if MessageDlg ( cstrInvalidUHFFrequencyMsg + #13 + cstrContinueMsg, mtError,
                      [mbYes, mbNo],0) = mrNo then
      begin
        frmDataEntry.rbtVHF.Checked := True;
        Exit;
      end
      else
      begin
        frmDataEntry.edtRXFrequency.Text := '';
        frmDataEntry.edtTXFrequency.Text := '';
      end;// if MessageDlg ( cstrInvalidVHFFrequencyMsg
    end;// if not ValidUHFFrequency

  end;// if frmDataEntry.rbtVHF.Checked

  if frmDataEntry.rbtVHF.Checked then
  begin
    if frmDataEntry.rbtSimplex.Checked then
      frmDataEntry.edtOffsetShift.Text := ''
    else
      frmDataEntry.edtOffsetShift.Text := gcstrVHFShiftOffset;
  end
  else
  begin
    if frmDataEntry.rbtSimplex.Checked then
      frmDataEntry.edtOffsetShift.Text := ''
    else
      frmDataEntry.edtOffsetShift.Text := gcstrUHFShiftOffset;
  end;// if frmDataEntry.rbtVHF.Checked

  frmDataEntry.edtTXFrequency.Text := frmDataEntry.CalculateTXFrequency;

end;// procedure SetShifOffset;

//------------------------------------------------------------------------------------------
procedure SetToneFreq;
begin
  if frmDataEntry.rbtNoTones.Checked then
    frmDataEntry.cbxTones.Text := ''
  else
    frmDataEntry.cbxTones.Text := IntToStr (frmDataEntry.cbxTones.ItemIndex); //Items[0];
end;// procedure SetToneFreq;

//------------------------------------------------------------------------------------------
procedure TfrmDataEntry.SetDTSSCode;
begin

  if frmDataEntry.chkDTSS.Checked then
    frmDataEntry.edtDTSSCode.Text :=
       gvstrFAVChannelDataArray[frmDataEntry.vbytChannelNumber, gcbytDTSSCodeField]
  else
    frmDataEntry.edtDTSSCode.Text := '';

end;// procedure SetDTSSCode;

//------------------------------------------------------------------------------------------
//      DTSS CODE EDIT BOX ROUTINES
//------------------------------------------------------------------------------------------
procedure TFrmDataEntry.EnableDTSSCode;
begin

  frmDataEntry.edtDTSSCode.Enabled := True;
  frmDataEntry.edtDTSSCode.Color := clWhite;

end;// procedure EnableDTSSCode;

//------------------------------------------------------------------------------------------
procedure TFrmDataEntry.DisableDTSSCode;
begin

  frmDataEntry.edtDTSSCode.Enabled := False;
  frmDataEntry.edtDTSSCode.Color := clYellow;

end;// procedure DisableDTSSCode;

//==========================================================================================
//      FORM ROUTINES
//==========================================================================================
procedure TfrmDataEntry.FormActivate(Sender: TObject);
```

```
var
  vbytTemp : Integer;

begin

  // Set the Editbox lengths
  edtChannelName.MaxLength := gcbytMaxChannelNameLength;
  edtComments.MaxLength := gcbytMaxCommentsLength;
  edtDTSSCode.MaxLength := gcbytMaxDTSSCodeLength;
  edtRXFrequency.MaxLength := gcbytMaxFrequencyLength;
  edtTXFrequency.MaxLength := gcbytMaxFrequencyLength;

  // Init the DTSS edit box
  DisableDTSSCode;

  // Configure the Command Buttons
  bbtSave.Enabled := True;
  bbtClear.Enabled := True;
  bbtReset.Enabled := True;
  frmDataEntry.bbtCancel.Enabled := True;

  // Set the initial Tab position
  edtRXFrequency.SetFocus;

  // The remainder of the form initialization depends on the Data Entry type
  case vdetDataEntryType of

    detFAV : DataEntry_FAV_Init;
//    detVFO : VFOFormInit;
    detUHFMEM : DataEntry_UHFMEM_Init;
    detVHFMEM : DataEntry_VHFMEM_Init;

  end;// case vdetDataEntryType

end;// procedure TfrmDataEntry.FormActivate

//===============================================================================================
//     BUTTON ROUTINES
//===============================================================================================
procedure TfrmDataEntry.bbtCancelClick(Sender: TObject);
begin

  if MessageDlg('ConfirmReset', cstrCancelMsg,  mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    ModalResult := mrCLose
  else
    ModalResult := mrNone;

end;// procedure TfrmDataEntry.bbtCancel

//-----------------------------------------------------------------------------------------------
procedure TfrmDataEntry.bbtClearClick(Sender: TObject);

var
  vbytTemp1 : Byte;
  vbytTemp2 : Byte;

begin

  //======================
  // Clear the data elements
  //=======================

  if MessageDlg('Confirm Clear', cstrClearMsg,  mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin

    // The remainder of the form initialization depends on the Data Entry type
    case vdetDataEntryType of

      detFAV : begin
                 // The answer was Yes so we clear all of the fields of the record for this button
                 for vbytTemp1 := 1 to gcbytMaxChannelFieldCount do
                 begin
                   gvstrFAVChannelDataArray[vbytChannelNumber, vbytTemp1] := '';
                 end;// for vbytTemp1 := 1 to vbytChannelNumber do

                  // and clear the Favourite button
                 case vbytChannelNumber of
                   1 : frmMain.bbtFav01.Caption := '';
                   2 : frmMain.bbtFav02.Caption := '';
                   3 : frmMain.bbtFav03.Caption := '';
                   4 : frmMain.bbtFav04.Caption := '';
                   5 : frmMain.bbtFav05.Caption := '';
                   6 : frmMain.bbtFav06.Caption := '';
                   7 : frmMain.bbtFav07.Caption := '';
                   8 : frmMain.bbtFav08.Caption := '';
                   9 : frmMain.bbtFav09.Caption := '';
                   10 : frmMain.bbtFav10.Caption := '';
                   11 : frmMain.bbtFav11.Caption := '';
                   12 : frmMain.bbtFav12.Caption := '';
                 end;//  case vbytChannelNumber of

               end;

//      detVFO : VFOFormInit;
        detUHFMEM : begin
                      // The answer was Yes so we clear all of the fields of the record for this button
                      for vbytTemp1 := 1 to gcbytMaxChannelFieldCount do
                      begin
                        gvstrUHFChannelDataArray[vbytChannelNumber, vbytTemp1] := '';
                      end;// for vbytTemp1 := 1 to vbytChannelNumber do
                    end;

        detVHFMEM : begin
                      // The answer was Yes so we clear all of the fields of the record for this button
                      for vbytTemp1 := 1 to gcbytMaxChannelFieldCount do
                      begin
                        gvstrVHFChannelDataArray[vbytChannelNumber, vbytTemp1] := '';
                      end;// for vbytTemp1 := 1 to vbytChannelNumber do
                     end;


    end;// case vdetDataEntryType of

    gvstrTMVDataChanged := True;

  end;// if MessageDlg('Confirm Clear'
end;// procedure TfrmDataEntry.bbtClearClick

//-----------------------------------------------------------------------------------------
procedure TfrmDataEntry.bbtResetClick(Sender: TObject);
begin

  // This routine resets the data entry form to the original data that was present when the
  // for was first opened. It resets the form data fields to the "Original" data field
  // variables.
  if MessageDlg('Confirm Reset', cstrResetMsg,  mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin
    // The remainder of the form initialization depends on the Data Entry type
    case vdetDataEntryType of

      detFAV : DataEntry_FAV_Init;
//      detVFO : VFOFormInit;
      detUHFMEM : DataEntry_UHFMEM_Init;
      detVHFMEM : DataEntry_VHFMEM_Init;

    end;// case vdetDataEntryType

  end;// if MessageDlg('Confirm Reset',

  ModalResult := mrNone;

end;// procedure TfrmDataEntry.bbtResetClick

//-----------------------------------------------------------------------------------------
procedure TfrmDataEntry.bbtSaveClick(Sender: TObject);

var
  vstrTStr : string;

begin

  //=================================
  //  First we validate Data elements
  //=================================

  // RX Frequency
  if rbtUHF.Checked then
  begin
    if not ValidUHFFrequency(edtRXFrequency.Text) then
    begin
      MessageDlg ('Invalid RX Frequency', cstrInvalidUHFFrequencyMsg, mterror, [mbOk], 0);
      modalResult := mrNone;
      edtTXFrequency.Text := '';
      edtRXFrequency.SetFocus;
      Exit;
    end;
  end
  else
  begin
    if not ValidVHFFrequency(edtRXFrequency.Text) then
    begin
      MessageDlg ('Invalid RX Frequency', cstrInvalidVHFFrequencyMsg, mterror, [mbOk], 0);
      modalResult := mrNone;
      edtTXFrequency.Text := '';
      edtRXFrequency.SetFocus;
      Exit;
    end;
  end;

  // DTSS Code
  if chkDTSS.Checked then
  begin
    if length (edtDTSSCode.Text) <> gcbytMaxDTSSCodeLength then
    begin
      ShowMessage(cstrInvalidDTSSCodeMsg);
      modalResult := mrNone;
      edtDTSSCode.SetFocus;
      Exit;
    end;// if length edtDTSSCode.Text <> gcbytMaxDTSSCodeLength
  end;// if chkDTSS.Checked then

  //  Channel Name
  if length (edtChannelName.Text) < gcbytMinChannelNameLength then
  begin
    ShowMessage(cstrInvalidChannelNameMsg);
    modalResult := mrNone;
    edtChannelName.SetFocus;
    Exit;
  end;// if length (edtChannelName.Text) < gcbytMinChannelNameLength

  //==========================================================================
  //  Everything is valid so now we save the data in the appropriate array based on
  //  vdetDataEntryType.
  //==========================================================================
  case vdetDataEntryType of

    detFAV :  DataEntry_FAV_Save;
    detVFO : Begin

             end;

    detUHFMEM : DataEntry_UHFMEM_Save;

    detVHFMEM : DataEntry_VHFMEM_Save;
  end;// case vdetDataEntryType

  gvstrTMVDataChanged := True;

end;// procedure TfrmDataEntry.bbtSaveClick

//============================================================================
//     KEYPRESS ROUTINES
//============================================================================
procedure TfrmDataEntry.edtRXFrequencyKeyPress(Sender: TObject; var Key: char);
begin

  case Key of

    #8 : Exit; // <BS>
    #46 : if Length(edtRXFrequency.Text) = 3 then // <.>
                 Exit
                else
                begin
                  Key := #0;
                  Exit;
                end;// if Length(edtRXFrequency.Text) = 3
    #48..#57 : if (Length(edtRXFrequency.Text) < 3) or
                  (Length(edtRXFrequency.Text) > 3) then // <.>
                     Exit
               else
               begin
                 Key := #0;
                 Exit;
               end;// if Length(edtRXFrequency.Text) < 3
  else
    Key := #0;
  end;// case Key of

end;// procedure TfrmDataEntry.edtRXFrequencyKeyPress

//---------------------------------------------------------------------------------
procedure TfrmDataEntry.edtDTSSCodeKeyPress(Sender: TObject; var Key: char);
begin

  case Key of
    #8 : Exit; // <BS>
    #48..#57 : Exit;
  else
    Key := #0;
  end;// case Key of

end;// procedure TfrmDataEntry.edtDTSSKeyPress

//---------------------------------------------------------------------------------
procedure TfrmDataEntry.edtChannelNameKeyPress(Sender: TObject; var Key: char);
begin

  case Key of
    #8 : Exit; // <BS>
    #32 : Exit; // <Sp>
    #48..#57 : Exit; // <0>..<9>
    #65..#90 : Exit; // <A..Z>
    #97..#122 : Exit; // <a..z>
  else
    Key := #0;
  end;// case Key of

end;// procedure TfrmDataEntry.edtChannelNameKeyPress

//---------------------------------------------------------------------------------
procedure TfrmDataEntry.edtCommentsKeyPress(Sender: TObject; var Key: char);
begin

  case Key of
    #8 : Exit; // <BS>
    #32 : Exit; // <Sp>
    #48..#57 : Exit; // <0>..<9>
    #65..#90 : Exit; // <A..Z>
    #97..#122 : Exit; // <a..z>
  else
    Key := #0;
  end;// case Key of

end;// procedure TfrmDataEntry.edtCommentsKeyPress

```

```
//==========================================================================================
//     ON CHANGE ROUTINES
//==========================================================================================
procedure TfrmDataEntry.rbtVHFChange(Sender: TObject);
begin
  SetShiftOffset;
end;//procedure TfrmDataEntry.rbtVHFChange(

//------------------------------------------------------------------------------------------
procedure TfrmDataEntry.rbtSimplexChange(Sender: TObject);
begin
  SetShiftOffset;
end;// procedure TfrmDataEntry.rbtSimplexChange

//------------------------------------------------------------------------------------------
procedure TfrmDataEntry.rbtPlusChange(Sender: TObject);
begin
  SetShiftOffset;
end;// procedure TfrmDataEntry.rbtPlusChange

//------------------------------------------------------------------------------------------
procedure TfrmDataEntry.rbtMinusChange(Sender: TObject);
begin
  SetShiftOffset;
end;// rocedure TfrmDataEntry.rbtMinusChange

//------------------------------------------------------------------------------------------
procedure TfrmDataEntry.rbtNoTonesChange(Sender: TObject);
begin
  SetToneFreq;
end;// procedure TfrmDataEntry.rbtNoTonesChange

//------------------------------------------------------------------------------------------
procedure TfrmDataEntry.chkDTSSChange(Sender: TObject);
begin

  if chkDTSS.Checked then
  begin
    edtDTSSCode.Enabled := True;
    edtDTSSCode.Color := clWhite;
    edtDTSSCode.Text := gvstrFAVChannelDataArray[frmDataEntry.vbytChannelNumber,
      gcbytDTSSCodeField];
    edtDTSSCode.SetFocus;
  end
  else
  begin
    edtDTSSCode.Enabled := False;
    edtDTSSCode.Color := clYellow;
    edtDTSSCode.Text := '';
  end;// if chkDTSS.Checked

end;// procedure TfrmDataEntry.chkDTSSChange

//==========================================================================================
//         ON EXIT ROUTINES
//==========================================================================================
procedure TfrmDataEntry.edtRXFrequencyExit(Sender: TObject);
begin

  if rbtVHF.Checked then
  begin
    if not ValidVHFFrequency(edtRXFrequency.Text) then
      Exit;
  end
  else
  begin
  if not ValidUHFFrequency(edtRXFrequency.Text) then
      Exit;
  end;// if rbtVHF.Checked

  edtTXFrequency.Text := CalculateTXFrequency;

end;// procedure TfrmDataEntry.edtRXFrequencyExit

//===========================================================================================
end.// unit DataEntry;

```