```
unit MEM_UHF;

{$mode objfpc}{$H+}

//========================================================================================
//
//  Mem_VHF.pas
//
//  Calls: AppConstants
//         AppVariables
//         BCCommand : SetVHFBand
//         BUFCommand : SetBuffer
//         LCDDisplay : UpdateLCDDisplay
//         Mem
//         Utilities : GetToneFrequencyFromToneNr
//
//  Called By: MEM : TfrmMEM.Setup
//                   SetVHFChannel
//
//  Ver: 1.0.0
//
//  Date: 11 Aug 2013
//
//========================================================================================

interface

uses
  Classes, Dialogs, SysUtils,
  // Application Units
  AppConstants, AppVariables, BCCommand, BUFCommand, LCDDisplay, Utilities;

procedure LoadUHFStringGrid;
procedure SetUHFChannel;

implementation

uses
  Mem;

//========================================================================================
procedure LoadUHFStringGrid;

var
  vbytTemp : Byte;
  vstrTStr : String;

begin

  for vbytTemp := 1 to gcbytMaxUHFChannels do
  begin

    // Channel Nr
    frmMem.sgrUHF.Cells[gcbytChMemNrCol, vbytTemp] := IntToStr(vbytTemp);

    // Channel Name
    frmMem.sgrUHF.Cells[gcbytNameCol, vbytTemp] :=
                    gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField];

    // RX FREQUENCY
    if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytRXFrequencyField]) > 0 then
      frmMem.sgrUHF.Cells[gcbytRXFreqCol, vbytTemp] :=
         Copy(gvstrUHFChannelDataArray[vbytTemp,gcbytRXFrequencyField],3,3) +
         '.' +
         Copy(gvstrUHFChannelDataArray[vbytTemp,gcbytRXFrequencyField],6,3)
    else
      frmMem.sgrUHF.Cells[gcbytRXFreqCol, vbytTemp] := '';

    // SHIFT
    vstrTStr := gvstrUHFChannelDataArray[vbytTemp,gcbytShiftCol+1];
    case vstrTStr of
      gcstrShiftSimplex : frmMem.sgrUHF.Cells[gcbytShiftCol, vbytTemp] := gcstrTMV7ShiftSimplex;
      gcstrShiftPlus : frmMem.sgrUHF.Cells[gcbytShiftCol, vbytTemp] := gcstrTMV7ShiftPlus;
      gcstrShiftMinus : frmMem.sgrUHF.Cells[gcbytShiftCol, vbytTemp] := gcstrTMV7ShiftMinus;
    end;// case vstrTStr

    // Offset
    if Length(gvstrUHFChannelDataArray[vbytTemp,gcbytShiftOffsetField]) > 0 then
      if gvstrUHFChannelDataArray[vbytTemp,gcbytShiftCol+1] = gcstrShiftSimplex then
        frmMem.sgrUHF.Cells[gcbytOffsetCol, vbytTemp] := ''
      else
        frmMem.sgrUHF.Cells[gcbytOffsetCol, vbytTemp] :=
              Copy(gvstrUHFChannelDataArray[vbytTemp,gcbytShiftOffsetField],2,2) +
              '.' +
              Copy(gvstrUHFChannelDataArray[vbytTemp,gcbytShiftOffsetField],4,2)
    else
      frmMem.sgrUHF.Cells[gcbytOffsetCol, vbytTemp] := '';

    // Tone or CTCSS
    // We only load this field if there is a valid record
    if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField]) > 0 then
    begin
      if gvstrUHFChannelDataArray[vbytTemp,gcbytToneField] = gcstrOn then
        frmMem.sgrUHF.Cells[gcbytToneCTCSSCol, vbytTemp] := gcstrTMV7Tone
      else if gvstrUHFChannelDataArray[vbytTemp,gcbytCTCSSField] = gcstrOn then
        frmMem.sgrUHF.Cells[gcbytToneCTCSSCol, vbytTemp] := gcstrTMV7CTCSS
      else frmMem.sgrUHF.Cells[gcbytToneCTCSSCol, vbytTemp] := gcstrTMV7None;
    end;// if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField]) > 0

    // Tone Freq
    // We only load the tone Frequency if the record is valid and a tone is selected
    if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField]) > 0 then
    begin
      // We have a valid record we now check to see if there is a Tone or CTCSS on
      case frmMem.sgrUHF.Cells[gcbytToneCTCSSCol, vbytTemp] of
        gcstrTMV7Tone : begin
                          frmMem.sgrUHF.Cells[gcbytToneCTCSSFreqCol, vbytTemp] :=
                            GetToneFrequencyFromToneNr
                            (StrToInt (gvstrUHFChannelDataArray[vbytTemp,gcbytToneNrField]));
                        end;// gcstrTMV7Tone
        gcstrTMV7CTCSS : begin
                           frmMem.sgrUHF.Cells[gcbytToneCTCSSFreqCol, vbytTemp] :=
                             GetToneFrequencyFromToneNr
                             (StrToInt (gvstrUHFChannelDataArray[vbytTemp,gcbytCTCSSNrField]));
                         end;// gcstrTMV7CTCSS
      else // gcstrTMV7None
        frmMem.sgrUHF.Cells[gcbytToneCTCSSFreqCol, vbytTemp] := '';
      end;// case frmMem.sgrUHF.Cells[cbytToneCTCSSCol, vbytTemp]

    end;// Length (gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField]) > 0

    // RF Power
    vstrTStr := gvstrUHFChannelDataArray[vbytTemp,gcbytRFPowerField];
    case vstrTStr of
      gcstrRFPowerLow : frmMem.sgrUHF.Cells[gcbytRFPowerCol, vbytTemp] := gcstrTMV7RFPowerLow;
      gcstrRFPowerMedium : frmMem.sgrUHF.Cells[gcbytRFPowerCol, vbytTemp] := gcstrTMV7RFPowerMedium;
      gcstrRFPowerHigh : frmMem.sgrUHF.Cells[gcbytRFPowerCol, vbytTemp] := gcstrTMV7RFPowerHigh;
    end;//  case vstrTStr of

    // DTSS
    vstrTStr := gvstrUHFChannelDataArray[vbytTemp,gcbytDTSSField];
    case vstrTStr of
      gcstrOn : frmMem.sgrUHF.Cells[gcbytDTSSCol, vbytTemp] := gcstrTMV7On;
      gcstrOff : frmMem.sgrUHF.Cells[gcbytDTSSCol, vbytTemp] := gcstrTMV7Off;
    else
      frmMem.sgrUHF.Cells[gcbytRFPowerCol, vbytTemp] := '';
    end;//  case vstrTStr of

    // DTSS CODE
    if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField]) > 0 then
    begin
      if gvstrUHFChannelDataArray[vbytTemp,gcbytDTSSField] = gcstrOn then
        frmMem.sgrUHF.Cells[gcbytDTSSCodeCol, vbytTemp] :=
                    gvstrUHFChannelDataArray[vbytTemp,gcbytDTSSCodeField]
      else
        frmMem.sgrUHF.Cells[gcbytDTSSCodeCol, vbytTemp] := '';
    end;// if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytChannelNameField]) > 0

    // REVERSE
    vstrTStr := gvstrUHFChannelDataArray[vbytTemp,gcbytReverseField];
    case vstrTStr of
      gcstrOn : frmMem.sgrUHF.Cells[gcbytReverseCol, vbytTemp] := gcstrTMV7On;
      gcstrOff : frmMem.sgrUHF.Cells[gcbytReverseCol, vbytTemp] := gcstrTMV7Off;
    else
      frmMem.sgrUHF.Cells[gcbytReverseCol, vbytTemp] := '';
    end;//  case vstrTStr of

    // SCAN
    vstrTStr := gvstrUHFChannelDataArray[vbytTemp,gcbytScanField];
    case vstrTStr of
      gcstrOn : frmMem.sgrUHF.Cells[gcbytScanCol, vbytTemp] := gcstrTMV7On;
      gcstrOff : frmMem.sgrUHF.Cells[gcbytScanCol, vbytTemp] := gcstrTMV7Off;
    else
      frmMem.sgrUHF.Cells[gcbytScanCol, vbytTemp] := '';
    end;//  case vstrTStr of

    // Step
    if Length (gvstrUHFChannelDataArray[vbytTemp,gcbytStepField]) > 0 then
      vstrTStr := gvstrUHFChannelDataArray[vbytTemp,gcbytStepField]
    else vstrTStr := '';

    if Length(vstrTStr) > 0 then
      frmMem.sgrUHF.Cells[gcbytStepCol, vbytTemp] := gvstrStepArray[StrToInt(vstrTStr)]
    else
```

```
      frmMem.sgrUHF.Cells[gcbytStepCol, vbytTemp] := '';

    // COMMENTS
    frmMem.sgrUHF.Cells[gcbytCommentCol, vbytTemp] :=
                gvstrUHFChannelDataArray[vbytTemp,gcbytCommentsField];

  end;// for vbytTemp := 1 to gcbytMaxUHFChannels do

end;// procedure LoadUHFStringGrid;

//===============================================================================
procedure SetUHFChannel;
begin

  // vbytChannelNr is the index into the gvstrUHFChannelDataArray table.
  // First we make sure that we have a valid data record at this position by ensuring
  // the Channel Name contains data (Mandatory field).
  if gvintSelectedRow = 0 then gvintSelectedRow := 1;

  if Length ( gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytChannelNameField ] ) <
            gcbytMinChannelNameLength then
  begin
    showmessage('No Entry');
    Exit;
  end;// if Length ( gvstrFAVChannelDataArray

  // Here we have a valid data record so we load the appropriate buffer based on the
  // VFO field
  gvstrUHFDataSource := 'MEM';
  gvstrUHFRXFrequency := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytRXFrequencyField ];
  gvstrUHFStep := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytStepField ];
  gvstrUHFShift := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytShiftField ];
  gvstrUHFReverse := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytReverseField ];
  gvstrUHFTone := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytToneField ];
  gvstrUHFCTCSS := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytCTCSSField ];
  gvstrUHFDTSS := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytDTSSField ];
  gvstrUHFToneNr := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytToneNrField ];
  gvstrUHFDTSSCode := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytDTSSCodeField ];
  gvstrUHFCTCSSNr := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytCTCSSNrField ];
  gvstrUHFOffset := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytShiftOffsetField ];
  gvstrUHFScan := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytScanField ];
  gvstrUHFRFPower := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytRFPowerField ];
  gvstrUHFChannelName := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytChannelNameField ];
  gvstrUHFChannelComments := gvstrUHFChannelDataArray[ gvintSelectedRow, gcbytCommentsField ];

  //***      DisplayUHFBuffer;

  SetBuffer(gcstrUHFVFO);
  SetUHFBand;
  UpdateLCDDisplay;

end;// procedure SetUHFChannel;

//===============================================================================
end.// unit MEM_UHF;

```