```
 1: unit TMVFiles_UHF;
 2:
 3: {$mode objfpc}{$H+}
 4:
 5: //=============================================================================================
 6: //
 7: //  TMVFiles_UHF.pas
 8: //
 9: //  Calls: AppConstants
10: //         AppVariables
11: //         Utilities : GetStepIndex
12: //
13: //  Called By: TMVFiles : OpenTMVFile
14: //
15: //  Ver: 1.0.0
16: //
17: //  Date: 9 Aug 2013
18: //
19: //=============================================================================================
20:
21: interface
22:
23: uses
24:   Classes, SysUtils,
25:   // Application Units
26:   AppConstants, AppVariables, Utilities;
27:
28: function MakeUHFRecord(vbytRecord : Byte) : string;
29: procedure ParseUHFRecord(vbytRecNr : Byte; vstrRecord : string);
30:
31: implementation
32:
33: //=============================================================================================
34: function MakeUHFRecord(vbytRecord : Byte) : string;
35:
36: var
37:   vstrTRecord : string;
38:
39: begin
40:
41:   // Record Nr
42:   vstrTRecord := IntToStr(vbytRecord) + ',';
43:
44:   // VFO
45:   if gvstrUHFChannelDataArray[vbytRecord, gcbytVFOField] = gcstrUHF then
46:     vstrTRecord := vstrTRecord + gcstrTMV7VFO_UHF + ','
47:   else if gvstrUHFChannelDataArray[vbytRecord, gcbytVFOField] = gcstrVHF then
48:     vstrTRecord := vstrTRecord + gcstrTMV7VFO_VHF + ','
49:   else
50:     vstrTRecord := vstrTRecord + '' + ',';
51:
52:   // RX Frequency
53:   if Length(gvstrUHFChannelDataArray[vbytRecord, gcbytRXFrequencyField]) > 0 then
54:     vstrTRecord := vstrTRecord +
55:                    Copy(gvstrUHFChannelDataArray[vbytRecord, gcbytRXFrequencyField], 3, 3) +
56:                    '.' +
57:                    Copy(gvstrUHFChannelDataArray[vbytRecord, gcbytRXFrequencyField], 6, 3) +
58:                    ','
59:   else
```

```
 60:     vstrTRecord := vstrTRecord + '' + ',';
 61:
 62:   // Step
 63:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytStepField]) = 0 then
 64:     vstrTRecord := vstrTRecord + '' + ','
 65:   else
 66:     vstrTRecord := vstrTRecord +
 67:                  gvstrStepArray[StrToInt(gvstrUHFChannelDataArray[vbytRecord,
 68:                  gcbytStepField])] + ',';
 69:
 70:   // Shift
 71:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytShiftField]) = 0 then
 72:     vstrTRecord := vstrTRecord + '' + ','
 73:   else
 74:     case gvstrUHFChannelDataArray[vbytRecord, gcbytShiftField] of
 75:       gcstrShiftSimplex :
 76:         vstrTRecord := vstrTRecord + gcstrTMV7ShiftSimplex + ',';
 77:       gcstrShiftPlus :
 78:         vstrTRecord := vstrTRecord + gcstrTMV7ShiftPlus + ',';
 79:       gcstrShiftMinus :
 80:         vstrTRecord := vstrTRecord + gcstrTMV7ShiftMinus + ',';
 81:     end;// case gvstrUHFChannelDataArray[vbytRecord, gcbytShiftField]
 82:
 83:   // Reverse
 84:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytReverseField]) = 0 then
 85:     vstrTRecord := vstrTRecord + '' + ','
 86:   else
 87:     if gvstrUHFChannelDataArray[vbytRecord, gcbytReverseField] = gcstrOff then
 88:         vstrTRecord := vstrTRecord + gcstrTMV7Off + ','
 89:     else
 90:       vstrTRecord := vstrTRecord + gcstrTMV7On + ',';
 91:
 92:   //  TONE - CTCSS
 93:   // This is sort of comlicated. We have three radio boxes which give us the correct
 94:   // status of the TMV7 Tone functions as well as a list of tones in the combo box
 95:   //
 96:   // First we determine the Tone Function Status
 97:   If (Length(gvstrUHFChannelDataArray[vbytRecord, gcbytToneField]) = 0) and
 98:      (Length(gvstrUHFChannelDataArray[vbytRecord, gcbytCTCSSField]) = 0)
 99:   then
100:   begin
101:     // There is no Tone Function selected so we null out the Status Field
102:     vstrTRecord := vstrTRecord + '' + ',';
103:     // and the Tone Frequency field
104:     vstrTRecord := vstrTRecord + '' + ',';
105:   end
106:   else
107:   begin
108:     // We have a Tone Function selected so we determine both the Function as well
109:     // as the Tone Freq
110:     if  ((gvstrUHFChannelDataArray[vbytRecord, gcbytToneField]) = gcstrOff) and
111:         ((gvstrUHFChannelDataArray[vbytRecord, gcbytCTCSSField]) = gcstrOff) then
112:     begin
113:       // Both Tone Function and Frequency are turned off
114:       vstrTRecord := vstrTRecord + gcstrTMV7None + ',';
115:       vstrTRecord := vstrTRecord + '' + ',';
116:     end
117:     else if ((gvstrUHFChannelDataArray[vbytRecord, gcbytToneField]) = gcstrOn) then
118:     begin
```

```
119:       // The Tone Function and Frequency are turned on
120:       vstrTRecord := vstrTRecord + gcstrTMV7Tone + ',';
121:       vstrTRecord := vstrTRecord +
122:            GetToneFrequencyFromToneNr(StrToInt(gvstrUHFChannelDataArray[vbytRecord,
123:            gcbytToneNrField]))+ ',';
124:     end
125:     else
126:     begin
127:       // The CTCSS Function and Frequency are tuirned on
128:       vstrTRecord := vstrTRecord + gcstrTMV7CTCSS + ',';
129:       vstrTRecord := vstrTRecord +
130:            GetToneFrequencyFromToneNr(StrToInt(gvstrUHFChannelDataArray[vbytRecord,
131:            gcbytCTCSSNrField])) + ',';
132:     end;// if  ((gvstrUHFChannelDataArray[vbytRecord, gcbytToneField])
133:
134:   end;// If (Length(gvstrVHFChannelDataArray[vbytRecord, gcbytToneField]) = 0)
135:
136:   // DTSS Function and Code
137:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytDTSSField]) = 0 then
138:     vstrTRecord := vstrTRecord + '' + ',' + '' + ','
139:   else
140:     if gvstrUHFChannelDataArray[vbytRecord, gcbytDTSSField] = gcstrOff then
141:     begin
142:         vstrTRecord := vstrTRecord + gcstrTMV7Off + ',';
143:         vstrTRecord := vstrTRecord + '' + ',';
144:     end
145:     else
146:     begin
147:       vstrTRecord := vstrTRecord + gcstrTMV7On + ',';
148:       vstrTRecord := vstrTRecord + gvstrUHFChannelDataArray[vbytRecord,
149:       gcbytDTSSCodeField] + ','
150:     end;// if gvstrUHFChannelDataArray[vbytRecord, gcbytDTSSField]
151:
152:   // Shift Offset
153:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytShiftOffsetField]) = 0 then
154:     vstrTRecord := vstrTRecord + '' + ','
155:   else
156:     vstrTRecord := vstrTRecord +
157:     Copy(gvstrUHFChannelDataArray[vbytRecord, gcbytShiftOffsetField], 2, 2) +
158:     '.' +
159:     Copy(gvstrUHFChannelDataArray[vbytRecord, gcbytShiftOffsetField], 4, 2) + ',';
160:
161:   // Scan
162:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytScanField]) = 0 then
163:     vstrTRecord := vstrTRecord + '' + ','
164:   else
165:     if gvstrUHFChannelDataArray[vbytRecord, gcbytScanField] = gcstrOff then
166:         vstrTRecord := vstrTRecord + gcstrTMV7Off + ','
167:     else
168:       vstrTRecord := vstrTRecord + gcstrTMV7On + ',';
169:
170:   // RF Power
171:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytRFPowerField]) = 0 then
172:     vstrTRecord := vstrTRecord + '' + ','
173:   else
174:     case gvstrUHFChannelDataArray[vbytRecord, gcbytRFPowerField] of
175:       gcstrRFPowerLow :
176:         vstrTRecord := vstrTRecord + gcstrTMV7RFPowerLow + ',';
177:       gcstrRFPowerMedium :
```

```
178:         vstrTRecord := vstrTRecord + gcstrTMV7RFPowerMedium + ',';
179:       gcstrRFPowerHigh :
180:         vstrTRecord := vstrTRecord + gcstrTMV7RFPowerHigh + ',';
181:     end;// case gvstrUHFChannelDataArray[vbytRecord, gcbytRFPowerField]
182:
183:   // Channel Name
184:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytChannelNameField]) = 0 then
185:     vstrTRecord := vstrTRecord + '' + ','
186:   else
187:     vstrTRecord := vstrTRecord + gvstrUHFChannelDataArray[vbytRecord,
188:                    gcbytChannelNameField] + ',';
189:
190:   // Comments
191:   If Length(gvstrUHFChannelDataArray[vbytRecord, gcbytCommentsField]) = 0 then
192:     vstrTRecord := vstrTRecord + ''
193:   else
194:     vstrTRecord := vstrTRecord + gvstrUHFChannelDataArray[vbytRecord, gcbytCommentsField];
195:
196:   Result := vstrTRecord;
197:
198: end;// function MakeUHFRecord
199:
200: //=============================================================================================
201: procedure ParseUHFRecord(vbytRecNr : Byte; vstrRecord : string);
202:
203: var
204:   vbytCommaPos : Byte;
205:   vstrTStr : string;
206:   vbytTbyt : Byte;
207:   vstrTToneNr : string;
208:
209: begin
210:
211:   // Bypass the Record Nr
212:   vbytCommaPos := Pos(',', vstrRecord );
213:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
214:
215:   // VFO
216:   vbytCommaPos := Pos(',', vstrRecord );
217:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
218:   case  vstrTstr of
219:     '' :     gvstrUHFChannelDataArray[vbytRecNr, gcbytVFOField] := '';
220:     gcstrTMV7VFO_UHF : gvstrUHFChannelDataArray[vbytRecNr, gcbytVFOField] := gcstrUHFVFO;
221:     else
222:       gvstrUHFChannelDataArray[vbytRecNr, gcbytVFOField] :=gcstrVHFVFO;
223:   end;// case of vstrTstr
224:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
225:
226:   // RX Frequency
227:   vbytCommaPos := Pos(',', vstrRecord );
228:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
229:   if Length(vstrTStr) > 0 then
230:     vstrTStr := '00' + Copy(vstrRecord, 1, 3) + Copy(vstrRecord, 5, 3) + '000';
231:   gvstrUHFChannelDataArray[vbytRecNr, gcbytRXFrequencyField] := vstrTStr;
232:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
233:
234:   // Step Size
235:   vbytCommaPos := Pos(',', vstrRecord );
236:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
```

```
237:   if Length(vstrTStr) > 0 then
238:   begin
239:     // We look for a decimal to determine if we have to conver to Real
240:     if Pos('.', vstrTStr) > 0 then
241:       vbytTByt := GetStepIndex(StrToFloat(vstrTStr))
242:     else
243:       vbytTByt := GetStepIndex(StrToFloat(vstrTStr + '.0'));
244:     gvstrUHFChannelDataArray[vbytRecNr, gcbytStepField] := IntToStr(vbytTByt);
245:   end
246:   else
247:     gvstrUHFChannelDataArray[vbytRecNr, gcbytStepField] := '';
248:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
249:
250:   // Shift
251:   vbytCommaPos := Pos(',', vstrRecord );
252:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
253:   case vstrTStr of
254:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytShiftField] := '';
255:     gcstrTMV7ShiftSimplex :
256:       gvstrUHFChannelDataArray[vbytRecNr, gcbytShiftField] := gcstrShiftSimplex;
257:     gcstrTMV7ShiftPlus :
258:       gvstrUHFChannelDataArray[vbytRecNr, gcbytShiftField] := gcstrShiftPlus;
259:     gcstrTMV7ShiftMinus :
260:       gvstrUHFChannelDataArray[vbytRecNr, gcbytShiftField] := gcstrShiftMinus;
261:   end;// case vstrTStr of
262:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
263:
264:   // Reverse
265:   vbytCommaPos := Pos(',', vstrRecord );
266:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
267:   case vstrTStr of
268:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytReverseField] := '';
269:     gcstrTMV7Off : gvstrUHFChannelDataArray[vbytRecNr, gcbytReverseField] := gcstrOff;
270:     else
271:       gvstrUHFChannelDataArray[vbytRecNr, gcbytReverseField] := gcstrOn;
272:   end;// case vstrTStr of
273:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
274:
275:   // Tone Function - This takes care of both Tone and CTCSS On/Off fields
276:   vbytCommaPos := Pos(',', vstrRecord );
277:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
278:   case vstrTStr of
279:     '' : begin
280:            gvstrUHFChannelDataArray[vbytRecNr, gcbytToneField] := '';
281:            gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSField] := '';
282:          end;// '', cstrNone
283:     gcstrTMV7None : begin
284:                 gvstrUHFChannelDataArray[vbytRecNr, gcbytToneField] := gcstrOff;
285:                 gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSField] := gcstrOff;
286:               end;// '', cstrNone
287:     gcstrTMV7Tone : begin
288:                 gvstrUHFChannelDataArray[vbytRecNr, gcbytToneField] := gcstrOn;
289:                 gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSField] := gcstrOff;
290:               end;// cstrTone
291:     else
292:       begin
293:         gvstrUHFChannelDataArray[vbytRecNr, gcbytToneField] := gcstrOff;
294:         gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSField] := gcstrOn;
295:       end;
```

```
296:   end;// case vstrTStr of
297:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
298:
299:   // Tone Frequency - This takes care of both Tone and CTCSS Frequency fields
300:   vbytCommaPos := Pos(',', vstrRecord );
301:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
302:   if Length(vstrTStr) > 0 then
303:   begin
304:     // There is a Frequency in the record. That mneans that either Tone or CTCSS have
305:     // been selected. vstrTStr contains the Tone Frequency as a string
306:     if gvstrUHFChannelDataArray[vbytRecNr, gcbytToneField] = gcstrOn then
307:     begin
308:       // Tone has been selected so we have to populate the Tone Nr field and Default the
309:       // CTCSS Nr field
310:       vbytTByt := GetToneNrFromFrequency(vstrTStr);
311:       vstrTToneNr := IntToStr(vbytTByt);
312:       if Length(vstrTToneNr) = 1 then
313:         vstrTToneNr := '0' + vstrTToneNr;
314:       gvstrUHFChannelDataArray[vbytRecNr, gcbytToneNrField] := vstrTToneNr;
315:       gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSNrField] := '01';
316:     end
317:     else
318:     begin
319:       // CTCSS has been selected so we have to populate the CTCSS Nr field and Default the
320:       // Tone Nr field
321:       vbytTByt := GetToneNrFromFrequency(vstrTStr);
322:       vstrTToneNr := IntToStr(vbytTByt);
323:       if Length(vstrTToneNr) = 1 then
324:         vstrTToneNr := '0' + vstrTToneNr;
325:       gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSNrField] := vstrTToneNr;
326:       gvstrUHFChannelDataArray[vbytRecNr, gcbytToneNrField] := '01';
327:     end;//éé if gvstrFAVChannelDataArray[vbytRecNr, gcbytToneField] = gcstrOn
328:   end
329:   else
330:   begin
331:     // There is no Tone Frequency in the record so we clear the Tone Nr fields
332:     gvstrUHFChannelDataArray[vbytRecNr, gcbytToneNrField] := '01';
333:     gvstrUHFChannelDataArray[vbytRecNr, gcbytCTCSSNrField] := '01';
334:   end;//  if Length(vstrTStr) > 0
335:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
336:
337:   // DTSS On/Off
338:   vbytCommaPos := Pos(',', vstrRecord );
339:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
340:   case vstrTStr of
341:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytDTSSField] := '';
342:     gcstrTMV7Off : gvstrUHFChannelDataArray[vbytRecNr, gcbytDTSSField] :=
343:                      gcstrOff;
344:     else
345:       gvstrUHFChannelDataArray[vbytRecNr, gcbytDTSSField] := gcstrOn;
346:   end;// case vstrTStr of
347:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
348:
349:   // DTSS Code
350:   vbytCommaPos := Pos(',', vstrRecord );
351:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
352:   case vstrTStr of
353:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytDTSSCodeField] := '000';
354:     else
```

```
355:      gvstrUHFChannelDataArray[vbytRecNr, gcbytDTSSCodeField] := vstrTStr;
356:   end;// case vstrTStr of
357:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
358:
359:   // Shift Offset
360:   vbytCommaPos := Pos(',', vstrRecord );
361:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
362:   case vstrTStr of
363:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytShiftOffsetField] := '';
364:     else
365:       gvstrUHFChannelDataArray[vbytRecNr, gcbytShiftOffsetField] := '0' +
366:           Copy(vstrTStr, 1, 2) + Copy(vstrTStr, 4, 2) + '0000';
367:   end;// case vstrTStr of
368:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
369:
370:   // Scan On/Off
371:   vbytCommaPos := Pos(',', vstrRecord );
372:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
373:   case vstrTStr of
374:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytScanField] := '';
375:     gcstrTMV7Off : gvstrUHFChannelDataArray[vbytRecNr, gcbytScanField] := gcstrOff;
376:     else
377:       gvstrUHFChannelDataArray[vbytRecNr, gcbytScanField] := gcstrOn;
378:   end;// case vstrTStr of
379:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
380:
381:   // RF Power
382:   vbytCommaPos := Pos(',', vstrRecord );
383:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
384:   case vstrTStr of
385:     '' : gvstrUHFChannelDataArray[vbytRecNr, gcbytRFPowerField] := '';
386:     gcstrTMV7RFPowerLow :
387:       gvstrUHFChannelDataArray[vbytRecNr, gcbytRFPowerField] := gcstrRFPowerLow;
388:     gcstrTMV7RFPowerMedium :
389:       gvstrUHFChannelDataArray[vbytRecNr, gcbytRFPowerField] := gcstrRFPowerMedium;
390:     gcstrTMV7RFPowerHigh :
391:       gvstrUHFChannelDataArray[vbytRecNr, gcbytRFPowerField] := gcstrRFPowerHigh;
392:   end;// case vstrTStr of
393:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
394:
395:   // Channel Name
396:   vbytCommaPos := Pos(',', vstrRecord );
397:   vstrTStr := Copy(vstrRecord, 1, vbytCommaPos-1);
398:   gvstrUHFChannelDataArray[vbytRecNr, gcbytChannelNameField] := vstrTStr;
399:   vstrRecord := Copy(vstrRecord, vbytCommaPos+1, Length(vstrRecord));
400:
401:   // Comments
402:   gvstrUHFChannelDataArray[vbytRecNr, gcbytCommentsField] := vstrRecord;
403:
404: end;// procedure ParseUHFRecord;
405:
406: //============================================================================
407: end.// unit TMVFiles_UHF;
408:
```