```
 1: unit SerialStuff;
 2:
 3: {$mode objfpc}{$H+}
 4:
 5: //=============================================================================================
 6: //
 7: //  SerialStuff.pas
 8: //
 9: //  Calls: Main;
10: //         HUtils
11: //         AppConstants
12: //         AppVariables
13: //
14: //  Called By: AGCommand : GetVHF_AGValue
15: //                         GetUHF_AGValue
16: //             BCCommand : TogglePTTBand
17: //             BYCommand : GetUHFBYStatus
18: //                         GetVHFBYStatus
19: //            Init : Initialize
20: //            PCCommand : ToggleRFPower
21: //            PSCommand : TogglePowerOnOff
22: //            SMCommand : GetVHF_SMValue
23: //                        GetUHF_SMValue
24: //
25: //  Ver: 1.0.0
26: //
27: //  Date: 22 Sep 2013
28: //
29: //=============================================================================================
30:
31: interface
32:
33: uses
34:   Classes, Dialogs, Forms, SysUtils,
35:   // Application Units
36:   AppConstants, AppVariables, COMPort, HUtils;
37:
38: Function OpenPort : Boolean;
39: Procedure ClosePort;
40: Function SendCommand (vstrKeyword, vstrParameters : string) : boolean;
41:
42: implementation
43:
44: uses
45:   Main;
46:
47: //=============================================================================================
48: //          SDPOSERIAL1 ROUTINES
49: //=============================================================================================
50: Function OpenPort : Boolean;
51: begin
52:
53:   // See if we have a COM port configured. If not, we prompt for one
54:   if gvstrCOMPort = '' then
55:   begin
56:     InfoMessageDlgOk('No COM Port Configured', 'Please enter a COM Port');
57:     frmCOMPort.showmodal;
58:   end;// if gvstrCOMPort = ''
59:
60:   // Try to open the COM port
```

```
 61:   frmMain.sdpoSerial1.Device := gvstrCOMPort;
 62:   try
 63:     frmMain.sdpoSerial1.Active := True;
 64:   except
 65:   end;// try
 66:
 67:   if not frmMain.sdpoSerial1.Active then
 68:   begin
 69:       // It did not open, so we assume that the port is incorrect and promt
 70:       // for a new selection.
 71:       MessageDlg(' Unable to open ' + gvstrCOMPort + '. This is probably due' + #13 +
 72:              '       to an invalid COM port selection. ' + #13 +
 73:              '         Select the correct COM port.',
 74:             mtError, [mbOk], 0 );
 75:     // Now we try to opne the new port. If that fails, we give up, display
 76:     // an error message and return to frmMain.
 77:     frmCOMPort.showmodal;
 78:
 79:     // Try to open the COM port
 80:     try
 81:       frmMain.sdpoSerial1.Active := True;
 82:     except
 83:     end;// try
 84:
 85:     if not frmMain.sdpoSerial1.Active then
 86:     begin
 87:       MessageDlg('Unable to Open this port as well' + gvstrCOMPort + '. There appears ' + #13 +
 88:                'to be a system problem.',
 89:               mtError, [mbOk], 0 );
 90:       Result := False;
 91:       gvstrCOMPort := gcstrDefCOMPort;
 92:       gvblnTMV7OnLine := False;
 93:       Exit;
 94:     end;//if not frmMain.sdpoSerial1.Active
 95:   end;// if not frmMain.sdpoSerial1.Active
 96:
 97:   // Now see if the radio is there. This command will time out if
 98:   //     The TMV7 is not connected to the serial cable, or
 99:   //     The TMV7 is not turned on initially.
100:   if not SendCommand ('PS', '') then
101:   begin
102:     MessageDlg(' Unable to Communicate with the Radio. This is ' + #13 +
103:              ' probably due to a bad or misconnected serial' + #13 +
104:              '     cable or the TMV7 may be turned off.', mtError, [mbOk], 0 );
105:     ClosePort;
106:     gvblnTMV7OnLine := False;
107:     Result := False;
108:     Exit;
109:   end;// if not SendCommand ('PS', '')
110:
111:   gvblnTMV7OnLine := True;
112:
113: end;// Function OpenPort
114:
115: //=============================================================================
116: Procedure ClosePort;
117: begin
118:   frmMain.sdpoSerial1.Active := False;
119: end;//Procedure ClosePort
120:
```

```
121: //=============================================================================================
122: Function SendCommand (vstrKeyword, vstrParameters : string) : boolean;
123:
124: var
125:   vstrCommand : string;
126:
127: begin
128:
129:   if frmMain.SdpoSerial1.Active then
130:   begin
131:
132:     // Form the TMV7 command. If there are Parameters passed, then we add a <Space> and the
133:     // Parameters and a <CR> to the Keyowrd, otherwise we simply terminate the Keyword with
134:     // a <CR>.
135:     if Length (vstrParameters) > 0 then
136:       vstrCommand := vstrKeyword + ' ' + vstrParameters + #13
137:     else
138:       vstrCommand := vstrKeyword + #13;
139:
140:     // Wait for all received messages to be processed. If we time out we have an error
141:     // and display a message and exit the function with a Result of False.
142:
143:     //   SendCommand := False;
144:     //   Send message
145:
146:     // The Receive buffer is clear so now we can send the command
147:     gvblnKeywordMatched := False;
148:     gvstrKeywordSent := vstrKeyword;
149:     frmMain.sdpoSerial1.WriteData (vstrCommand);
150:     SendTimeoutTimerReset;
151:
152:     // Here we wait for a match from the received response. If we get it, we reset it to
153:     // False, set the Function Result to True and exit the function. Otherwise we exit
154:     // the Function with a result of False (defaulted earlier.
155:     while frmMain.tmrSendTimeout.Enabled do
156:     begin
157:       Application.ProcessMessages;
158:       if gvblnKeywordMatched then
159:       begin
160:         SendCommand := True;
161:         Exit;
162:       end;
163:     end;// while frmMain.tmrSendTimeout.Enabled
164:
165:     SendCommand := False;
166:
167:   end
168:   else
169:   begin
170:
171:   end;// if frmMain.sdpoSerial1.Active
172:
173: end;// Function SendCommand
174:
175: //=============================================================================================
176:
177: end.// unit SerialStuff;
178:
```