```
unit ResponseParser;

//========================================================================================
//
//  ResponseParser.pas
//
//  Calls: AGCommand : AGResponseHandler
//         BCCommand : BCResponseHandler
//         BUFCommand : BUFResponseHandler
//         BYCommand : BYResponseHandler
//         Main
//         PSCommand : PSResponseHandler
//         RXCommand : RXResponseHandler
//         SMCommand : SMResponseHandler
//         SQCommand : SQResponseHandler
//         TXCommand : TXResponseHandler
//         Variables
//
//  Called BY: Main : TfrmMain.SdpoSerial1RxData
//
//  Ver: 1.0.0
//
//  Date: 7 July 2013
//
//========================================================================================


{$mode objfpc}{$H+}

interface

uses
  Classes, Dialogs, SysUtils,
  //   Application Variables
  AppVariables, AGCommand, BCCommand, BUFCommand, BYCommand, PSCommand, RXCommand, SMCommand,
  SQCommand, TXCommand;

procedure ParseResponse (vstrResponse : string);

implementation

//========================================================================================

uses
    Main;

procedure ParseResponse (vstrResponse : string);

var
  vstrParameters : string;
  vintSpacePtr : Integer;
  vintTemp : Integer;

begin
   if Length (vstrResponse) > 0 then
   begin


   //  MessageDlg('vstrResponse = ' + vstrResponse, mtInformation, [mbOk], 0 );
```

```
     // First, we separate the Keyword and Parameters.

    // The last character in the response will always be a <CR> terminator. The Keyword can
    // be 1, 2 or 3 characters long and is either followed by a <Sp>, or terminated by a
    // <CR>. Single character repsones are error conditions and will always be follwed by
    // a <CR>. 2 and 3 character responses may be followed by a <Sp> or terminated by a <CR>.
    // If followed by a <Sp> the response will always have one or more parameters following
    // the <Sp>. If terminated by a <CR>, there will be no parameters.

    // Remove mysterious Null characters at the begining of a PS response.
    if vstrResponse[1] = #0 then
    vstrResponse := Copy(vstrResponse,2, Length(vstrResponse));

     // First we look for single character responses. They will always have a length of 2.
    if Length(vstrResponse) > 2 then
    begin
      // These must be two or three character responses, so now we look for a space to see
      // if there are any parameters
      vintSpacePtr := Pos (' ', vstrResponse);
      // If there is no space, there are no parameters
      if vintSpacePtr = 0 then
      begin
        gvstrKeywordRcvd := Copy(vstrResponse, 1, Length(vstrResponse)-1);
        vstrParameters := '';
      end
      else
      begin
        gvstrKeywordRcvd := Copy(vstrResponse, 1, vintSpacePtr-1);
        vstrParameters := Copy(vstrResponse, vintSpacePtr+1, Length(vstrResponse)- (vintSpacePtr+1));
      end;//if vintSpacePtr = 0
    end
    else
    begin
      // These must be one character responses
      gvstrKeywordRcvd := vstrResponse[1];
      vstrParameters := '';
    end;// if Length(vstrResponse) > 2

    if (gvstrKeywordSent = gvstrKeywordRcvd) or (gvstrKeywordRcvd = '?') then
    begin
      gvblnKeywordMatched := True;
      frmMain.tmrSendTimeout.Enabled := False;
    end;

    Case gvstrKeywordRcvd  of
      'AG' : AGResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'BC' : BCResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'BUF' : BUFResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'BY' : BYResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'PS' : PSResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'RX' : RXResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'SM' : SMResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'SQ' : SQResponseHandler(gvstrKeywordRcvd, vstrParameters);
      'TX' : TXResponseHandler(gvstrKeywordRcvd, vstrParameters);
    Else
      gvblnKeywordMatched := True;
      frmMain.tmrSendTimeout.Enabled := False;
    End;//Case vstrKeyWord

  end;

end;// procedure ParseResponse (Response : string);

end.//

```