CSV format parsing

csvpars3_tst.zip

This module is part of the original MMBasic library. It is reproduced here with kind permission of Hugh Buckle and Geoff Graham. Be aware it may reference functionality which has changed or is deprecated in the latest versions of MMBasic.

CSVPARS1.BAS

```
' CSV Parsing routine
' CircuitGizmos March 2013
' ------------------------
' A routine to parse a CSV string
' Optionally specify
' - index to the entry required
' - the delimiter
' - suppress quotes


Print "CSV parsing test"
Print

Print "Data with space and comma"
cline$ = "12 0 in,out "
Print cline$
Print "________"
Print FIELD$( cline$, 0 )
Print FIELD$( cline$, 1 )
Print FIELD$( cline$, 2 )
Print FIELD$( cline$, 3 )
Print FIELD$( cline$, 4 )
Print FIELD$( cline$, 5 )
Print
Print

Print "Data with space, comma, and tab"
cline$ = "12" + Chr$(9) + "0 in,out"
Print cline$
Print "________"
Print FIELD$( cline$, 0 )
Print FIELD$( cline$, 1 )
Print FIELD$( cline$, 2 )
Print FIELD$( cline$, 3 )
Print FIELD$( cline$, 4 )
Print FIELD$( cline$, 5 )
Print
Print
```

```
   Print "Data with just comma"
   Print cline$
   Print "________"
   Print FIELD$( cline$, 0, ",")
   Print FIELD$( cline$, 1, ",")
   Print FIELD$( cline$, 2, ",")
   Print
   Print

   Print "Data with quotes"
   cline$ = "12," + Chr$(34) + "0 in" + Chr$(34) + ",out,fred," + Chr$(34) +
"zest step" + Chr$(34) + ",buh"
   Print CLINE$
   Print "________"
   Print FIELD$( cline$, 1 )
   Print FIELD$( cline$, 2 )
   Print FIELD$( cline$, 3 )
   Print FIELD$( cline$, 4 )
   Print FIELD$( cline$, 5 )
   Print FIELD$( cline$, 6 )
   Print
   Print

   Print "Data with quotes supressed"
   Print CLINE$
   Print "________"
   Print FIELD$( cline$, 2,,1 )
   Print FIELD$( cline$, 5,,1 )
   Print
   Print

   Print "Data with quotes"
   cline$ = Chr$(34) + "0 in" + Chr$(34) + ",out,fred," + Chr$(34) + "zest
step" + Chr$(34)
   Print CLINE$
   Print "________"
   Print FIELD$( cline$, 1 )
   Print FIELD$( cline$, 2 )
   Print FIELD$( cline$, 3 )
   Print FIELD$( cline$, 4 )



   '------------------------------------------------------------ ------------
-
   '
   ' FIELD - CSV field parsing
   ' field_string$ = String to parse
   ' field_index   = Index to requested field
   ' field_delim$  = Optional field delimeters
```

```
   ' field_supress = Suppress quote in quoted field
   '
   ' Parsing a CVS string - CircuitGizmos March 2013
   '------------------------------------------------------------ ------------
-

Function FIELD$( field_string$, field_index, field_delim$, field_supress )
   Local field_loop
   Local field_dcount
   Local field_dloop
   Local field_length
   Local field_field$

   field_length = Len(field_string$)
   field_field$ = ""
   field_dcount = 1
   field_loop = 1

   ' Delimiter defaults: SPACE COMMA TAB
   If field_delim$ = "" Then field_delim$ = " ," + Chr$(9)

   Do
      field_delim = 0
      ' Process inside paren (no delimiters)
      If Mid$( field_string$, field_loop, 1 ) = Chr$(34) Then
         Do
            If field_dcount = field_index Then
               field_field$ = field_field$ + Mid$( field_string$,
field_loop, 1 )
            EndIf
            field_loop = field_loop + 1
         Loop Until (Mid$( field_string$, field_loop, 1 ) = Chr$(34)) Or
(field_loop > field_length)
      EndIf

      ' Search through delimiters
      For field_dloop = 1 To Len(field_delim$)
         ' If char is a delim
         If Mid$( field_string$, field_loop, 1 ) = Mid$( field_delim$,
field_dloop, 1 ) Then
            field_dcount = field_dcount + 1
            field_delim = 1
         EndIf
      Next field_dloop

      ' Add to returned string if not delim
      If field_delim = 0 Then
         If field_dcount = field_index Then
            field_field$ = field_field$ + Mid$( field_string$, field_loop, 1
)
         EndIf
```

```
      EndIf

      field_loop = field_loop + 1
   Loop Until field_loop > field_length

   ' Suppress quote / return field
   For field_loop = 1 To Len(field_field$)
      If (Mid$( field_field$, field_loop, 1 ) = Chr$(34)) And (field_supress
<> 0) Then
         '
      Else
         FIELD$ = FIELD$ + Mid$( field_field$, field_loop, 1 )
      EndIf
   Next field_loop

End Function
```

CSVPARS2.BAS

```
  ' A version of CSVPARS by TZAdvantage
  ' March 2013
  ' A function to parse a CSV string
  'Required Global variable for GetFieldArray function
  'Dimension this variable so that it can hold all values
  Dim GetFieldArray$(12)
  ' Value is a minimal GPS record
  Value$ =  "$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,0
03.1,W*6A"
  Cls
  Print "GPS Record:"
  Print Value$
  Print
  N = GetFieldArray(Value$)
  'A list of all values
  Print "List of all fields"
  Print "------------------"
  For Index = 0 To N - 1
    Print Index, GetFieldArray$(Index)
  Next
  Print "------------------"
  Print
  'or directly use a field
  Speed = Val(GetFieldArray$(7))
  Print "A field used by its index"
  Print "-------------------------"
  Print "Speed = " Speed
  Print "-------------------------"
  End
  '
```

```
================================================================================
=
  ' GetFieldArray
  ' A function to split a string into an Array of fields
  '
  ' Author:  TZ Advantage
  ' Date:    5 march 2013
  ' Version: 1.0
  '
  ' This function requires the use of a global variable
  ' named GetFieldArray$(). It should be dimensioned large
  ' enough to contain all fields.
  '
  ' Best used in situations when all fields are needed as
  ' it will be the fastest.
  ' Consider using the function GetField()
  ' when only one or two fields are needed or speed is of no concern
  '
  ' Function parameters:
  ' Record$:    A string containing the delimited fields
  ' Delimiter$: (Optional) delimiter character, when omitted a comma is used
  ' KeepQuotes: (Optional) [0|1]
  '             Use value 1 if double quotes around field values have to be
kept,
  '             when omitted or value 0 is used double quotes are discarded
  '
================================================================================
=
Function GetFieldArray( Record$, Delimiter$, KeepQuotes )
  Local Index, Char, InQuote, Count
  InQuote = 0
  Count = 0
  ' If no delimiter is provided use a comma
  If Delimiter$ = "" Then Delimiter$ = ","
  ' Loop through all the characters
  For Index = 1 To Len(Record$)
    Char = Asc(Mid$(Record$, Index, 1))
    If Char = 34 Then
      ' Flip InQuote between 0 and 1.
      ' A 1 will signal that the next characters
      ' are within quotes and delimiters should be ignored
      InQuote = Not InQuote
    EndIf
    ' Only check for delimiters when not within two quotes
    If Not InQuote And Instr(Delimiter$, Chr$(char)) >= 1  Then
      Count = Count + 1
    Else
      ' Only collect characters for the requested field.
      'Add the character and only add quotes when KeepQuotes is 1
      If Char <> 34 Or KeepQuotes Then
        GetFieldArray$(Count) = GetFieldArray$(Count) + Chr$(char)
```

```
      EndIf
    EndIf
  Next
  GetFieldArray = Count + 1
End Function
```

CSVPARS3.BAS

```
  ' A version of CSVPARS by Andrew Rich VK4TEC
  ' March 2013
  ' Subroutine to parse a file of CSV data
  ' The relevant data must start with $ and end with *

  Open "CSVPars3.tst" For input As 1
  Max=10
  Dim Arg$(10)
  nmea_sentence
  For i=1 To 10
    Print i, Arg$(i)
  Next
  Close #1
  End


Sub nmea_sentence
  Do
    msg$ ="$" ' subroutine start
    Do While Input$(1, #1) <> "$" : Loop ' wait for the start
    For i = 0 To max
      arg$(i) = "" ' clear ready for data
      Do ' loops until a specific exit
        x$ = Input$(1, #1)
        ' Print x$
        msg$ = msg$ + x$ ' get the character
        If x$ = "," Then Exit ' new data field, increment i
        If x$ = "*" Then Return ' we have all the data so return
        arg$(i) = arg$(i) + x$
        'Print arg$(i) ' add to the data
      Loop
      'Print arg$(i) ' loop back for the next char
    Next i ' move to the next data field
    ' Print "Corrupt data..." ' exceeded max data items
    'Print msg$
  Loop

End Sub
```

CSVPARS4.BAS

```
  ' CSV Parse by Hugh Buckle
  ' March 2013
  ' Parses a CSV string into an array
  ' It is intended to be used with comma or tab delimiter.
  ' Loads each variable from a$ into array CSV$()
  ' Maximum size of the array is preset in CSVMax
  '
  ' Limitations:
  ' - Assumes no quoted strings
  ' - Consecutive delimiters leave a nul entry in the array

  ' Test using comma delimiter
  CSVDelimiter$=","
  a$="one,%,-,two,three,Now is the time,,,/p,1234.56"

  CSVMax=12                ' Get the first 12 values from the CSV file
  Dim CSV$(CSVMax)
  ParseCSV(a$,CSVDelimiter$,CSVMax)
  Print "Cell","Value"
  Print "----","-----"
  For i=1 To CSVMax
    ? i,CSV$(i)
  Next
End

Sub ParseCSV(p$,d$,n)
  ' Parse the CSV valuess into array CSV$()
  ' Using d$ as the delimiter
  Local i
  Strt=1
  For i=1 To n
    Ptr=Instr(Strt,p$,d$)
    If Ptr=0 Then Ptr=Len(a$)+1         'Last item has no delimiter
    CSV$(i)=Mid$(p$,Strt,Ptr-Strt)
    If Ptr>=Len(p$) Then Exit For
    Strt=Ptr+1
  Next
End Sub
```