## CSV FILES

Dilwyn Jones

This article discusses a legacy PC file format called CSV (or Comma Separated Values) and its application in helping us to transfer spreadsheet or database files between QL and PC. At one time, CSV files were to PCs what Export files were to QL Psion programs.

CSV files are fairly simple text files containing lists of values from a grid, or columns of text, or fields of a database. Many QL programs can generate the Psion export files, so I thought that if I could come up with a program to convert the export files to CSV format it would be one way of transferring spreadsheet and database data to a PC.

For example, if you have stored family history data in Archive, export the Archive database to a normal export file then run it through this little program (which I've called EXP2CSV_BAS) to convert the export file to a CSV file, then transfer that to the PC and import it into the PC program. So the rather long winded transfer method is:

1. Export the data from the QL program.
2. Run it through the EXP2CSV_bas program to convert to CSV file.
3. Transfer CSV file to PC.
4. Import into PC program.

Using this EXP2CSV_bas program, I have successfully transferred data from Abacus and Archive to M$ Excel in Windoze, for example.

## PSION EXPORT FILES

A Psion export file format is documented in the QL User Guide in the Information section after the Psion program guides. Basically, the first line contains a list of the field names, and subsequent lines contain the data from each field exported. Each full record takes one line in an export file. Strings are enclosed in quotes - if the string contains a quote

symbol, it must be doubled up, e.g. John "the man" Doe would need to be represented as "John ""the man"" Doe". Numbers are not quoted. The field names should end in '$' for strings, anything else is regarded as a number. The first field should apparently always be text. Here's a simple example, quoted from the manual:

"cashflow$","sales","costs","profits"<LF>
"January",1000,500,500<LF>
"February",1050,530,520<LF>
"March",1100,560,540<LF>

In this example, each record has four fields called cashflow$, sales, costs and profits. The $ in cashflow$ indicates this is a string type field, The other 3 fields are all numeric. Each line ends with a LF CHR$(10) according to the manual, although in my experience export files also work where the lines end with CR+LF too.

Export files should usually end with CHR$(26) to ensure that the end of file can be generated. The Psion programs decide this for themselves when exporting data, you need only worry about this if creating your own export format files, e.g. using PRINT# as shown in the QL User Guide.

## CSV FILE FORMAT

In a CSV file, each record is stored as a line made up of all the fields of that record, separated by a comma. A field can be "folded" (contain a linefeed) as long it's enclosed in double quotes.

The text of a field doesn't have to be enclosed in double quotes, but if the field is to contain a double quote symbol, the field text must be bounded by double quotes, and the double quote itself should be represented by a double pair of quotes, e.g. a field containg the text DILWYN "QL" JONES would be represented by "DILWYN ""QL"" JONES"

Likewise, if the field text is to contain a comma, that field should be enclosed in quotes, e.g. SAY 1,2,3! should be "SAY 1,2,3!"

Leading or trailing spaces in a field are normally ignored and stripped off when loaded, although by putting the text of the field in quotes you can force it see the spaces as part of the data.

If the field text is enclosed in quotes, these quotes are the "field delimiters", i.e. they show the start and end of the field.

Just because a field data is enclosed in quotes, you shouldn't assume it's text. The quotes may enclose a number - the type of data is generally deduced from whether or not the field contains a number, not by whether the field is enclosed with quotes.

Unlike the first line of an export file, the first line of a CSV file need not contain field names, although it is quite legitimate for a program to supply them and for the importing program to ask the user if the first line contains field names or not and work accordingly.

Between records (lines) you can use LF (10) or CR+LF (13+10).

**EXAMPLE CSV FILE**

Using the example quoted from the QL User Guide above, the first line is optional, although most CSV files do not include a first line like this.

```
cashflow$,sales,costs,profits
```

The above line can be included if you wish, but it may just appear as the first record in some programs (easy enough to delete if required). Generally, only the data appears in the CSV file:

```
January,1000,500,500
February,1050,530,520
March,1100,560,540
```

Quite simple isn't it (until you start messing with quotes and commas, etc, as described above). Generally, adding quotes does no harm if done properly, other than adding a bit of extra size to the file with redundant quote marks. PC programs will generally analyse the data on importing to find out what's numeric and what's plain text.

Figure 1 - The EXP2CSV_bas listing.

```
100 REMark EXP2CSV_bas by Dilwyn Jones, Feb 2008
110 REMark convert Psion export file to CSV format
120 :
130 CLS : CLS #0
140 INPUT #0,'Psion export file name > ';ip$
150 INPUT #0,'CSV file name          > ';op$
160 OPEN_IN #3,ip$ : OPEN_NEW #4,op$
170 include_fieldnames% = 0 : REMark change to 1 to include
field names
180 lf$ = CHR$(13)&CHR$(10) : REMark end of line characters for
CSV file
190 :
200 REMark read field names list from export file
210 fields = 0 : last = 0 : PRINT'FIELD NAMES : ';
220 REPeat program
230   fn$ = File_Entry$ : fields = fields + 1
240   PRINT !fn$; : IF include_fieldnames% = 1 : PRINT #4,fn$;
250   IF last = 1 THEN
260     IF include_fieldnames% = 1 THEN PRINT #4,lf$;
270     EXIT program
280   ELSE
290     PRINT #4,',';
300   END IF
310 END REPeat program
320 PRINT \\
330 :
340 CLS #0 : records = 1 : REMark read records 2 to n
350 :
360 REPeat program
370   FOR a = 0 TO fields-1
380     fn$ = File_Entry$
390     IF fn$ = CHR$(26) THEN EXIT a
400     PRINT !fn$;
410     IF (',' INSTR fn$) OR ('"' INSTR fn$) THEN
420       PRINT #4,'"'&fn$&'"'; : REMark quote strings
containing , or "
430     ELSE
440       PRINT #4,fn$;
450     END IF
```

```
460     IF last = 1 THEN PRINT #4,lf$; : EXIT a : ELSE PRINT
#4,',';
470   END FOR a
480   PRINT
490   IF fn$ = CHR$(26) THEN EXIT program
500   records = records + 1 : AT #0,0,0 : PRINT
#0,'RECORDS:';records
510 END REPeat program
520 :
530 CLOSE #3 : CLOSE #4
540 :
550 PRINT #0,'Program finished.' : STOP
560 :
570 DEFine FuNction File_Entry$
580   IF EOF(#3) : RETurn CHR$(26)
590   ch$ = '' : item$ = INKEY$(#3)
600   IF item$ = CHR$(26) : RETurn item$
610   IF item$ = CHR$(10) OR item$ = CHR$(13) : RETurn ""
620   IF item$ = '"' THEN
630     item$ = "" : Numeric = 0
640   ELSE
650     Numeric = 1 : IF item$ = ',' THEN RETurn
660   END IF
670   REPeat get_info
680     ch$ = INKEY$(#3)
690     IF ch$ = CHR$(10) OR ch$ = CHR$(13) THEN
700       last = 1
710       REPeat fl_loop
720         IF EOF(#3) : EXIT fl_loop
730         fl_loop = FPOS(#3)
740         k$ = INKEY$(#3)
750         IF k$ <> CHR$(10) AND k$ <> CHR$(13) THEN BGET
#3\fl_loop : EXIT fl_loop
760       END REPeat fl_loop
770       EXIT get_info
780     END IF
790     IF ch$ = ',' AND Numeric = 1 : last = 0 : EXIT get_info
800     IF ch$ = '"' OR (Numeric = 1 AND ch$ = ',') THEN
810       k$ = INKEY$(#3)
820       IF k$ = CHR$(10) OR k$ = CHR$(13) THEN
830         last = 1 : REMark end of line
840         REPeat fl_loop
850           IF EOF(#3) : EXIT fl_loop
860           fl_loop = FPOS(#3) : k$ = INKEY$(#3)
870           IF k$ <> CHR$(10) AND k$ <> CHR$(13) THEN BGET
#3\fl_loop : EXIT fl_loop
880         END REPeat fl_loop
890         EXIT get_info
900       ELSE
910         IF ch$ = '"' AND k$ = ',' THEN
920           last = 0 : EXIT get_info
```

```
930         ELSE
940           BGET #3\FPOS(#3)-1
950         END IF
960       END IF
970     END IF
980     item$ = item$ & ch$
990   END REPeat get_info
1000   RETurn item$
1010 END DEFine File_Entry$
```

Some notes on the listing, in case you wish to adapt it. There is no error trapping, to keep the program reasonably short for publication.

The program opens channel #3 to the input file (the export file) and channel #4 to the output file (the CSV file).

The main routine is the function called File_Entry$. This fetches one field of the file at a time. It returns the field value as a string and sets the variable "last" to indicate if it is the last item on the current line. As it stands, the routine uses global variables, many could be set as LOCal variabkles if you wish. The variable "last" needs to be global.

Line 170 sets a variable called "include_filenames%" to determine if the first line of the export file (the field names) is added to the CSV file. You can see how this is done in the first loop called "program". This loop also counts the number of fields on each line (fields per record, the variable "fields")

The second loop called "program" does the task of stepping through the remaining records. The "a" loop steps through all fields of each record. Note how line 410 determines whether the field needs to be enclosed in quotes or not in the CSV file output, by using INSTR to locate quotes or commas in the strings. You may wish to extend this to check for leading and trailing spaces in strings as these may be stripped by some PC programs if unquoted.

The program prints the data to the screen, but not necessarily in the same format as the file output.