Data File Formats Generated by LabVIEW

Igor Kagan, 12 April, 2000

http://igoresha.virtualave.net/LabVIEW/fileformats.html

LabVIEW can save numeric data to ASCII (text) files or byte stream (binary) files:

ASCII (text) files

In order to save the numeric data to ASCII text file, we should convert the numeric values to strings. Note that converting large amounts of data to and from strings can be time consuming. It can be done with function

Array To Spreadsheet String

This function converts an array of any dimension to *spreadsheet* string. Spreadsheet string is a table in string form, containing **tabs** (or other specified *delimiters*) separating column elements, a platform-dependent EOL (see below) character separating rows, and, for arrays of three or more dimensions, separated pages.
The numbers are converted according to supplied format string, for example **%.3f** (default format).

After conversion, the spreadsheet string can be written to a file with function

Write Characters To File

Alternatively, numeric data can be written to file with function

Write To Spreadsheet File

This function takes care of the conversion of numbers to spreadsheet string (it calls the Array to Spreadsheet String function to convert a 2D or 1D array of single-precision (SGL) numbers to a text string). This function writes the string to a file or appends the string to an existing file. You can optionally transpose the data. This VI opens or creates the file before writing to it and closes it afterwards. You can use this VI to create a text file readable by most spreadsheet applications.

Size and content of ASCII files

Each character in ASCII text file (including control characters) takes one byte. The number of characters per numeric value depends on conversion format. For example, if conversion format is **%.3f**, then the file data.dat, containing array of 5 samples (see FileFormatBench.vi) will look like:

```
1.000	2.220	3.333	4.444	5.555
```

The size of this file will be, correspondingly, will be 31 bytes: 5*5 + 4 tabs + 2 bytes of EOL (CR + LF, see table below).

The file data.txt, written with format **%f**, will have 8 characters per each sample:

```
1.000000	2.220000	3.333000	4.444000	5.555000
```

(5*8 + 4 tabs + 2 bytes of EOL = 46 bytes)

Special characters:

\r	Carriage Return
\t	Tab
\b	Backspace
\n	Newline
\f	Form Feed
\s	space
\xx	character with hexadecimal ASCII code xx (using 0 through 9 and upper case A through F)
\\	\
%%	%

A code does not exist for the platform-dependent end-of-line (eol) character. If you need to append one, use the **End-of-Line** constant from the String palette.

End of Line (eol)

Consists of a constant string containing the platform-dependent, end of line value. **For Windows, the value is CRLF**; for Macintosh, the value is CR; and for UNIX, the value is LF.

Table of eol control characters

Description	*Char*	*Hex*	*Dec*
line feed, new line	LF, NL	0A	&#010
carriage return	CR	0D	&#013

Binary Files

A simplest byte stream file is an array of binary 16-bit integers or 32-bit single-precision, floating point numbers, which you acquire from a data acquisition (DAQ) program.

Signed word integers (I16)

Each datum consists of 2 bytes. Only unscaled (integer) data can be written in this way. However, data acquisition parameters like scale offset, scale multiplier, and polarity can be saved in simple header in the same file.

Use function

Write To I16 File

Size of binary I16 file is n*2, where n is number of data samples.

Single Precision floats (SGL)

Each datum consists of 4 bytes. Scaled data can be written in this way.

Use function

Write To SGL File

Size of binary SGL file is n*4, where n is number of data samples.

Example: FileFormatsBench.vi

Panel:

Data File Formats Bench (see Diagram)

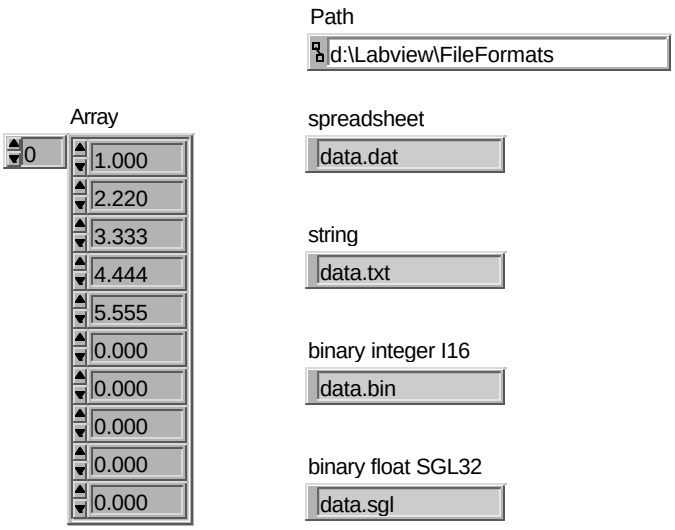
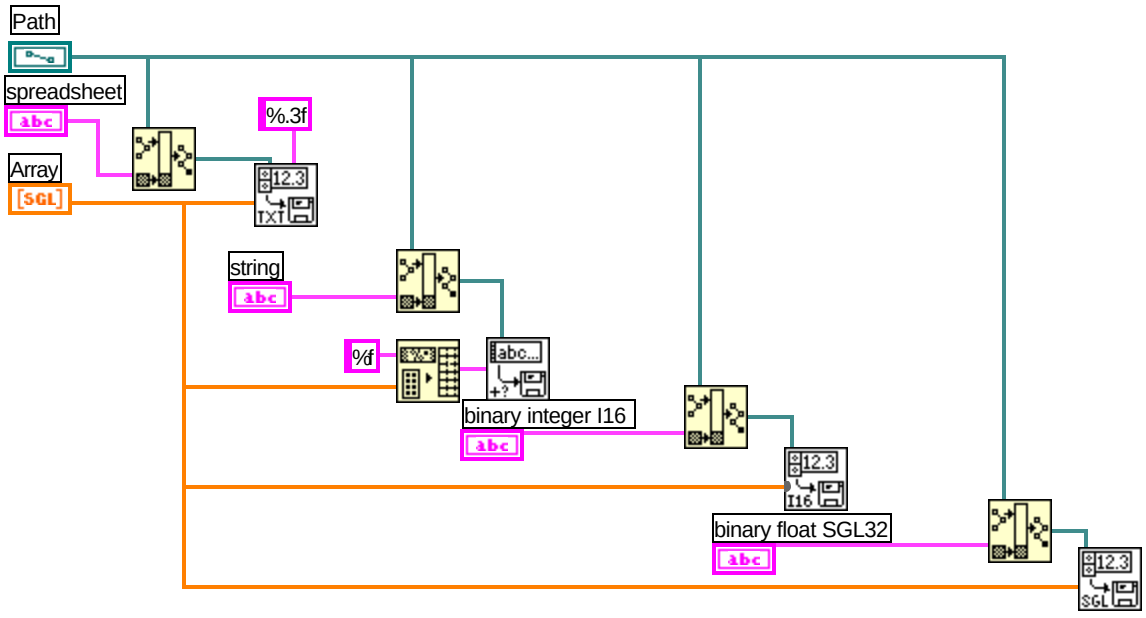


Diagram:



data.dat	31 bytes
data.txt	46 bytes
data.bin	10 bytes
data.sgl	20 bytes