Low-Level File I/O VI and Functions*

National Instruments

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 1.0†

Abstract

In this lesson, you will learn how to use the low-level File I/O VI and functions.

Use the following low-level File I/O VI and functions to perform basic file I/O operations:

Open/Create/Replace File - Opens an existing file, creates a new file, or replaces an existing file, programmatically or interactively using a file dialog box. You can optionally specify a dialog prompt, default file name, start path, or filter pattern. If file path is empty, the VI displays a dialog box from which you can select a file.

Read File - Reads data from an open file specified by refnum and returns it in data. Reading begins at the current file mark or a location specified by pos mode and pos offset. How the data is read depends on the format of the specified file.

Write File - Writes data to an open file specified by refnum. Writing begins at a location specified by pos mode and pos offset for byte stream files and at the end of the file for datalog files. data, header, and the format of the specified file determine the amount of data written.

Close File - Closes an open file specified by refnum and returns the path to the file associated with the refnum. Error I/O operates uniquely in this function, which closes regardless of whether an error occurred in a preceding operation. This ensures that files are closed correctly.

1 Error Handling

The low-level File I/O VIs and functions return error information. Wire the error information from the beginning of the VI to the end. Include an error handler VI, such as the Simple Error Handler (p. 1) VI located on the `Time & Dialog` palette, at the end of the VI to determine if the VI ran without errors. Use the error in and error out clusters in each VI you use or build to pass the error information through the VI.

*Version 1.1: Jul 16, 2004 3:25 pm -0500
†http://creativecommons.org/licenses/by/1.0

2 Saving Data in a New or Existing File

You can write any data type to a file you open or create with the File I/O VIs and functions. If other users or applications need to access the file, write string data in ASCII format to the file. Refer to the *LabVIEW Basics II: Development Course Manual* for more information about writing LabVIEW datalog or binary files.

You can access files either programmatically or interactively through a file dialog box. To access a file through a dialog box, do not wire `file path` in the `Open/Create/Replace File` VI. However, you can save time by programmatically wiring the default filename and path to the VI. Table 1 describes how pathnames are organized.

Platform	Pathname
Windows	Consists of the drive name, a colon, backslash-separated directory names, and the filename. For example, `c:\testdata\test1.dat` is the pathname to a file named `test1.dat` in the `testdata` directory.
UNIX	UNIX Consists of forward slash-separated directory names and the filename. For example, `/home/testdata/test1.dat` is the pathname to a file named `test1.dat` in the `testdata` directory in the `/home` directory. Filenames and pathnames are case sensitive.
Mac OS	Consists of the volume name (the name of the disk), a colon, colon-separated folder names, and the filename. For example, `Hard Disk:testdata:test1.dat` is the pathname to a file named `test1.dat` in a folder named `testdata` on a disk named `Hard Disk`.

Table 1

The block diagram shown in Figure 1 shows how to write string data to a file while programmatically wiring the filename and pathname. If the file already exists, it is replaced; otherwise a new file is created.

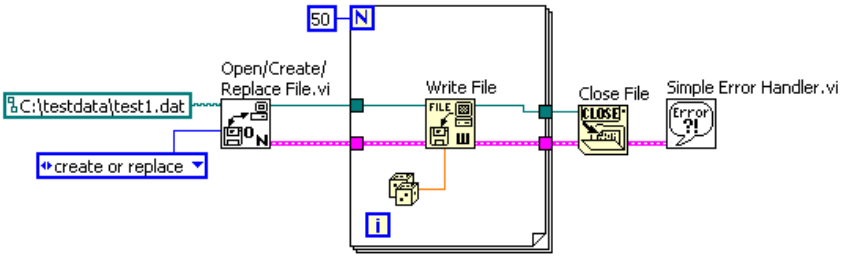

Figure 1

The `Open/Create/Replace File` VI opens the file `test1.dat`. The VI also generates a `refnum` and an error cluster.

When you open a file, device, or network connection, LabVIEW creates a `refnum` associated with that file, device, or network connection. All operations you perform on open files, devices, or network connections use `refnum`s to identify each object.

The error cluster and `refnum` pass in sequence from one node to the next. Because a node cannot execute until it receives all its inputs, passing these two parameters forces the nodes to run in order and creates a data dependency. The `Open/Create/Replace File` VI passes the refnum and error cluster to the `Write File` function, which writes the data to disk. When the `Write File` function finishes execution, it passes the refnum and error cluster to the `Close File` function, which closes the file. The `Simple Error Handler` VI examines the error cluster and displays a dialog box if an error occurred. If an error occurs in one node, subsequent nodes do not execute, and the VI passes the error cluster to the `Simple Error Handler` VI.