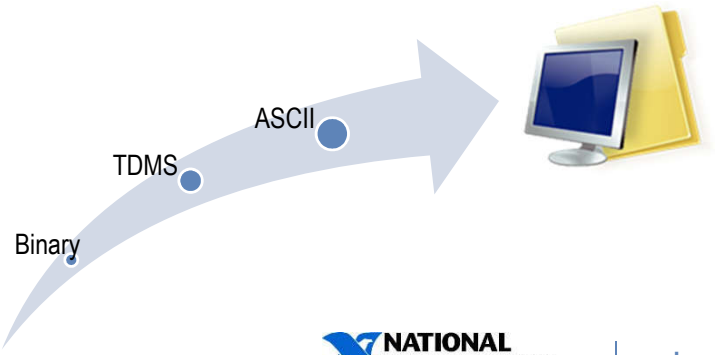
A. File Formats

At their lowest level, all files written to your computer's hard drive are a series of bits



NATIONAL INSTRUMENTS | ni.com/training

ASCII (American Standard Code for Information Interchange)

TDM (Technical Data Management)

In LabVIEW, three of the most common techniques for storing data are:

- ASCII file format
- direct binary storage
- TDMS file format

Arrow size is to indicate increasing file size.

A. File Formats

	ASCII	TDMS	Direct Binary
Numeric Precision	Good	Best	Best
Share data	Best (Any program easily)	Better (NI Programs easily)	Good (only with detailed metadata)
Efficiency	Good	Best	Best
Ideal Use	Share data with other programs when file space and numeric precision are not important	Share data with programs when storing simple array data and metadata	Store numeric data compactly with ability to random access

Use text format files for your data:

- To make the data available to other users or applications
- If disk space and file I/O speed are not crucial
- If performing random access reads or writes is unnecessary
- If numeric precision is not important

Use binary format files for your data:

- When numeric precision is important
- When you need to randomly access stored data
- When efficiency is important

A specific type of binary file, known as a Datalog file, is the easiest method for logging cluster data to file

- Stores arrays of clusters in a binary representation
- The storage format for Datalog files is complex, and therefore the data is difficult to access in any environment except LabVIEW

Use TDMS files for the following purposes:

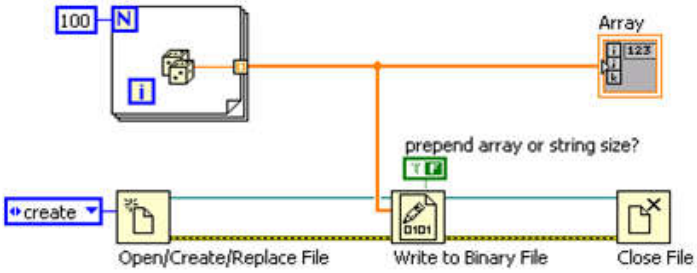
- To store test or measurement data
- To create a structure for grouping your data
- To store information about your data
- To read and write data at high speeds

TDMS file format:

- Binary file (.tdms) that contains data and stores properties about the data
- Binary index file (.tdms_index) that provides consolidated information on all the attributes and pointers in the TDMS file
- TDMS file format is publicly documented
- Can use with TDM Excel Add-in Tool

B. Binary Files

Use Binary File functions to interact directly with a binary file



NATIONAL INSTRUMENTS | ni.com/training

B. Binary Files—Bits/Bytes?

- A bit is a single binary value
  - Each bit is either ‘on’ or ‘off’ and is represented by a 1 or a 0
- A byte is a series of 8 bits

0	bit
00000000	byte

B. Binary Files—Storing Boolean Values

- LabVIEW represents Boolean values as 8-bit values in a binary file
- Eight zeroes represents False
  - [00000000]
- Any other value represents True
  - [00000001], [01000110], [11111111], and so on
- Files are divided into byte-sized chunks, making them much easier to read and process

B. Binary Files—Storing Boolean Values

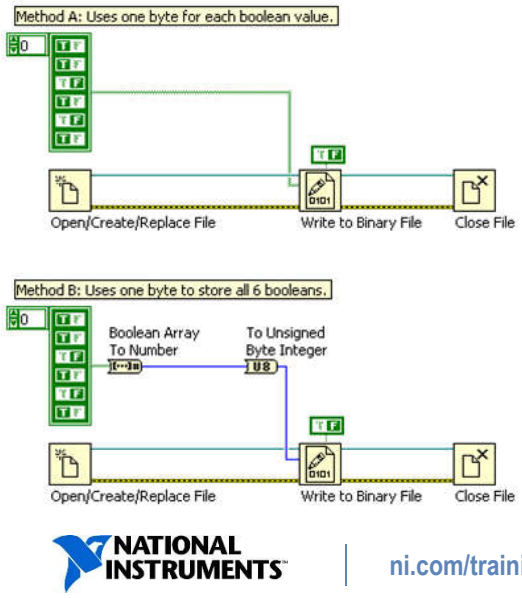
File Contents

Method A

00000001 00000001
00000000 00000001
00000000 00000001

Method B

00101011



NATIONAL INSTRUMENTS | ni.com/training

B. Binary Files—Storing Integers

Binary Value	U8 Value
00000000	0
00000001	1
00000010	2
11111111	255

B. Binary Files—Storing Integers

- Multi-byte integers are broken into separate bytes and are stored in files in either little-endian or big-endian byte order
- Using the Write to Binary File function, you can choose whether you store your data in little-endian or big-endian format

U32 Value	Little-endian Value	Big-endian Value
1	00000001 00000000 00000000 00000000	00000000 00000000 00000000 00000001

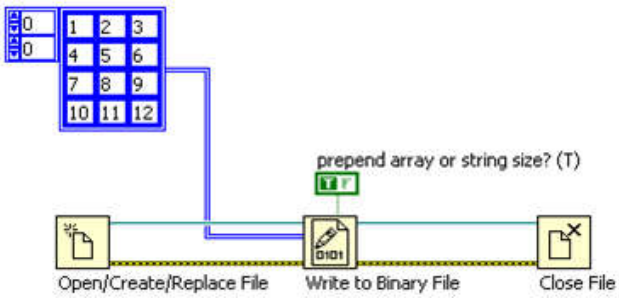
B. Binary Files—Storing Other Data Types

- Strings are stored as a series of unsigned 8-bit integers, each of which has a value in the ASCII Character Code Equivalents Table
  - This means that no difference exists between writing strings with the Binary File Functions and writing them with the Text File Functions
- Clusters are best represented in binary files by using Datalog Files

B. Binary Files—Storing Arrays

Arrays are represented as a sequential list of the elements

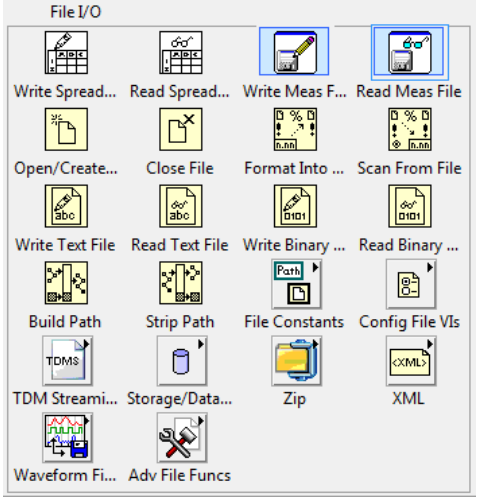
- Element representation depends upon the element type
- A header contains a 32-bit integer representing the size of each dimension
- Example
  A 2D array with a header contains: row integer, column integer, then array data



NATIONAL INSTRUMENTS | ni.com/training

File I/O

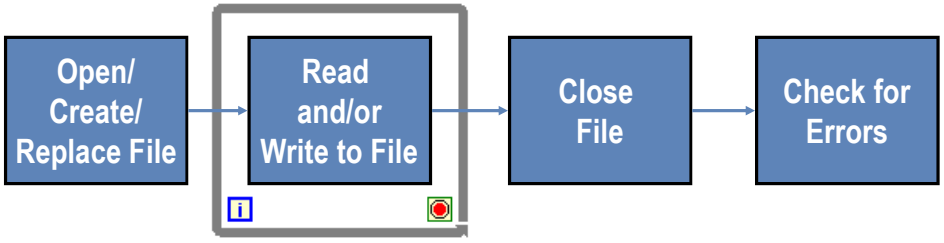
- Understanding File I/O
- File Formats
- High-Level File I/O VIs
- Low-Level File I/O VIs



NATIONAL INSTRUMENTS | ni.com/training

Understanding File I/O

- File I/O writes to or reads from a file.
- A typical file I/O operation involves the following process:



File Formats

LabVIEW can use or create the following file formats:

- **Binary**—Efficient, compact, and allows random access reading but not a human-readable file. Commonly used for high-speed and multi-channel DAQ.
- **ASCII**—Human-readable text file where data is represented as strings. Commonly used for low-speed DAQ.
- **LVM**—Format built on ASCII, the LabVIEW measurement data file (.lvm) is a tab-delimited text file you can open with a spreadsheet application or a text-editing application.
- **TDMS**—An NI-specific binary file format that contains data and stores properties about the data.

ASCII files are also called text files.

Students will learn more about TDMS files in LabVIEW Core 2.

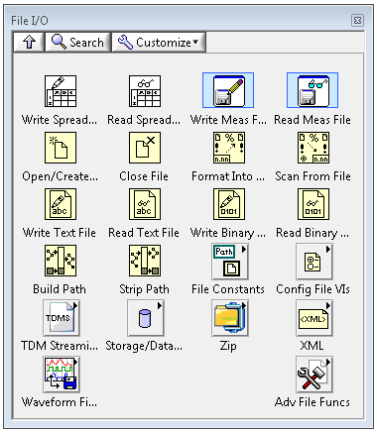
ASCII File Format

- In this course, you learn about creating text (ASCII) files.
- Use ASCII files in the following situations:
  - You want to access the file from another application.
  - Disk space and file I/O speed are not crucial.
  - You do not need to perform random access reads or writes.
  - Numeric precision is not important.

NATIONAL INSTRUMENTS | ni.com/training

High-Level File I/O

- High-level File I/O functions:
  - Perform all three steps (open, read/write, close) for common file I/O operations.
  - Might not be as efficient as the functions configured or designed for individual operations.
- Low-level File I/O functions:
  - Provide individual functions for each step.
  - Are efficient for writing to a file in a loop.



Point out the high-level functions and low-level functions on the palette.

High-level and low-level file I/O applies to many areas, such as DAQ and TCP/IP.

High-Level File I/O

Write to Spreadsheet File

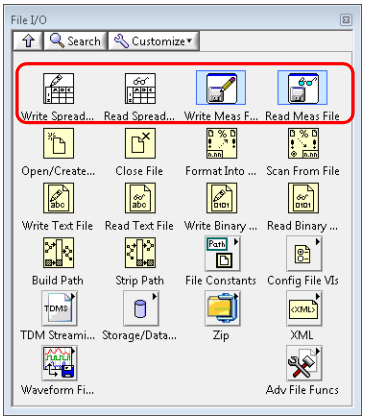
- Converts an array of double-precision numbers to a text string and writes the string to an ASCII file.

Read From Spreadsheet File

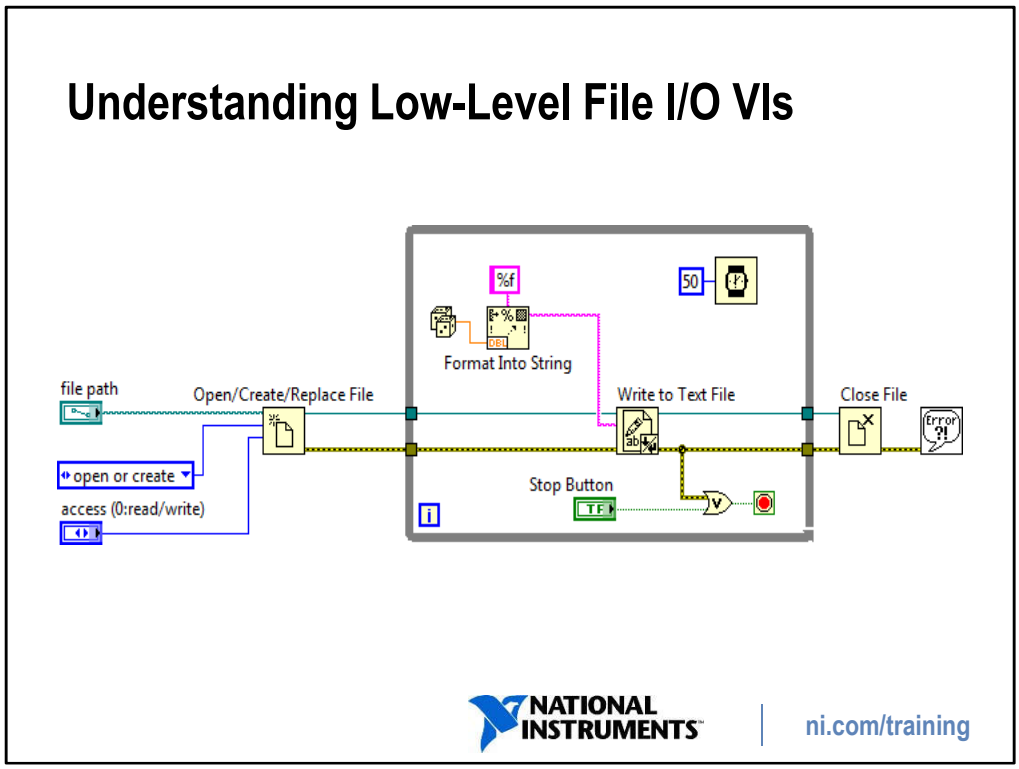
- Reads a specified number of lines or rows from a numeric text file and outputs a 2D array of double-precision numbers.

Write To/Read From Measurement File

- Express VIs that write data to or read data from an LVM or TDMS file format.



NATIONAL INSTRUMENTS | ni.com/training



Explain the concept of disk streaming here. Emphasize opening and closing the file outside of the loop.

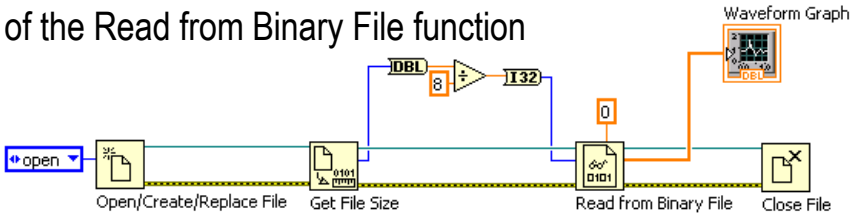
B. Binary Files—Sequential/Random Access

Two methods of accessing data:

- **Sequential Access**—Read each item in order, starting at the beginning of a file
- **Random Access**—Access data at an arbitrary point within the file

B. Binary Files—Sequential Access

- To sequentially access all of the data in a file, you can call the Get File Size function and use the result to calculate the number of items in the file, based upon the size of each item and the layout of the file
- You can then wire the number of items to the count terminal of the Read from Binary File function



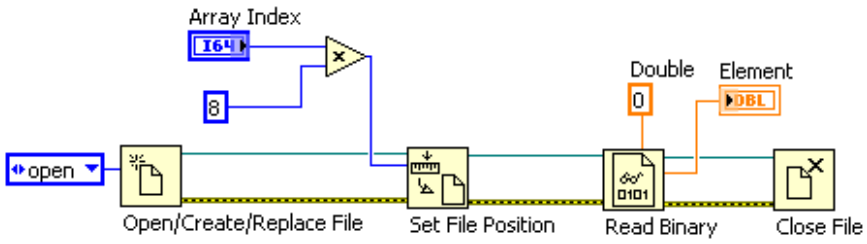
NATIONAL INSTRUMENTS | ni.com/training

Alternately, you can sequentially access the file one item at a time by repeatedly calling the Read from Binary File VI with the default count of 1.

You can check for the End of File error after calling the Read from Binary File function or calculate the number of reads necessary to reach the end of the file by using the Get File Size function.

B. Binary Files—Random Access

- Use the Set File Position function to set the read offset to the point in the file you want to begin reading
- The offset is in bytes; therefore, you must calculate the offset based upon the layout of the file



NATIONAL INSTRUMENTS | ni.com/training