Snap-Master Data File Formats

Data File Overview

Snap-Master reads both binary and exponential (also called ASCII or text) data files in a number of formats. There are three native file formats: Standard Binary, Fast Binary (also called FBDF for short), and Exponential. In addition, Snap-Master can import and export data in a number of generic formats, including Plotter and Comma Separated Variable (or CSV) files. Each format has its own advantages, and there are some general rules to follow when selecting the proper format for your instrument.

The difference between the native Snap-Master data file formats and the generic file formats is how the file is organized. Snap-Master native files use a format which contains a header and separate data frames. The header provides information on the data file that is used by the Disk In element to accurately recreate the data for post-processing. Generic data files do not contain this header, and the data is presented as one large series without any frame divisions.

Text-based data files (including the Exponential, ASCII Plotter, and CSV formats) files store the data as a standard ASCII text file that can be read by most programs. Also, the file can be read by humans using the DOS TYPE command. The main advantage of this universal data type is that the file can be read by most programs, including word processors, spreadsheets and databases.

There are two disadvantages when using an ASCII file format: the files require a large amount of disk space (up to eight times more than a binary file), and the files take proportionally longer to write to disk (up to ten times longer). Therefore, ASCII files should be used when the data is acquired at low speeds, or when the data must be exported to an external program.

The most efficient method of storing data is by writing the file in one of Snap-Master's binary formats. The binary files have a faster read and write time, but they may not be able to be read by external programs. In general, binary files are used for large data files or for higher acquisition rates.

Native Data Files

Format		Data Type	Frame Based?	Display While Storing?	Speed	Frequency Data?
Standard Binary	.DAT	Binary	Yes	Yes	Med	Yes
Fast Binary (FBDF)	.DAT	Binary	Yes	No	High	No
Exponential	.DAT	ASCII	Yes	Yes	Low	Yes

In Snap-Master, there are two native binary data file formats: Standard Binary and Fast Binary. Standard Binary files are sufficient for applications where the aggregate sampling rate is less than 10 KHz, or when performing post-process analysis on previously acquired data. (To find the aggregate sampling rate, multiply the number of channels by the sampling rate per channel for each input element. If you have more than one source of data, then add these values together.) The main advantage of this format is the data is converted to floating point numbers (using engineering units, if defined by the Sensor) for use by all of the Snap-Master elements, including the Display which allows you to view the data while saving it to disk.

The Fast Binary data format is optimized for high speed data acquisition. The main difference between it and the Standard Binary files is the Fast Binary does not convert the acquired data into the floating point number before it is written to disk. This saves time during the write cycle and allows for a higher throughput rate. In addition, all of the input channels are written to the disk file in order to maximize the performance.

While the instrument is writing data using the Fast Binary format, data can not be viewed using the Display element. This is because displaying data on the screen requires numerous calculations, which in turn reduces the speed of writing the data directly to disk. Therefore, if you attempt to start an instrument with an element that does not support the Fast Binary data format, a message will appear on screen informing you that the instrument can not run in its current configuration.

NOTE: If you are using the Fast Binary data format, DO NOT use any disk caching software (such as SMARTDRV, which comes with Windows). Disk caching software will degrade the performance of the Fast Binary format and your actual throughput rates will be lower than without the cache.

Generic Data Files

In addition to the native file formats, Snap-Master can also read and write data in generic formats that can be used by other programs. These formats include:

Format		Data Type	Frame Based?	Display While Storing?	Speed	Frequency Data?
Binary Plotter	.PLT	Binary	No	Yes	Med	Yes
ASCII Plotter	.PLT	ASCII	No	Yes	Low	Yes
Comma Separated Variable	.CSV	ASCII	No	Yes	Low	Yes

Data File Naming Conventions

When creating a Snap-Master native data file using a .DAT extension, you are actually writing data to more than one file. Each .DAT file acts as a reference to a number of .SM? files with the same file prefix. The .SM? files contain the actual data from the elements, where the ? is replaced by the element letter.

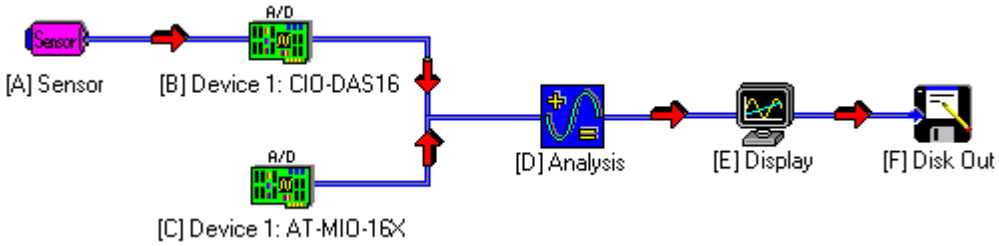

Figure 1 Sample Data File Naming Instrument

For example, this instrument is writing data in Standard Binary format to a file named TESTDATA. The first file that is created is TESTDATA.DAT. In addition, for each element you are saving data from (such as the B and C elements), the files TESTDATA.SMB and TESTDATA.SMC are created. TESTDATA.SMB will contain the raw data from the A/D Board marked element B, and the raw data from element C is saved in TESTDATA.SMC. If the Analysis element is creating new data channels such as R0, then a TESTDATA.SMR file is created.

The generic file formats use the extension assigned by the format when writing the data file (such as .PLT or .CSV). In addition, data from only one element letter can be stored in a generic file. This is because it is possible for each element letter to have a different number of points to save, so Snap-Master imposes this restriction when creating data files with the Disk Out element.

Data File Formats

This section of the manual documents the actual file formats used by Snap-Master's native file formats, as well as some of the generic file formats. This information will only be useful to users who plan on creating their own programs to write or read Snap-Master data files (such as custom file format converters or custom analysis programs).

Data File Structure

Whenever data is saved in one of Snap-Master's standard file format, at least two files are created: a .DAT file and one .SM* file for every element saving data. The actual data (including headers) is stored in the .SM* files, and the .DAT file acts as a pointer to all .SM* files with the same file prefix. The last letter in the file extension must correspond with the element letter in the CHAN$ array for proper operation.

All .SM* files consist of two major sections: the file header, and the data frames. The file header defines the parameters used in the data file, and the values for these parameters.

Each frame of data consists of two parts: the frame header and the raw data. The beginning of a new frame header is specified by a "TR", followed by the frame number. The next line contains a date and time stamp for the start of the frame.

Section	**Contains**
File Header	Introduction Parameter Name and Value List
Data Frame	Frame Header Data
Data Frame	Frame Header Data
Data Frame	Frame Header Data

etc. for the remaining frames.

Some important items to note when writing a Snap-Master data file:

- Spaces are ignored when reading the data file, except after an equal sign (=) that defines a numerical value where there must be no spaces.
- Snap-Master only understands its own internally declared parameters.

Header Information

The file header contains information used by Snap-Master to determine the parameters applicable to all data frames contained in the file. If the data file does not correspond to the settings in the header, then the file will not be replayed correctly.

The file header for a Snap-Master data file consists of two parts: the introduction, and the parameter list. The introduction contains an initialization line and the user-defined comments for the data file. The parameter list gives the parameter variables used in the data file, and the values for each parameter.

Initialization Line	The first line in the file header starts with "Snap-Master Data File", a comma, and a number specifying the number of parameters appearing in the data file, including the Comments line.
Comments	The next line(s) contains the user defined comments which are entered in the File Comments field of the Disk Out dialog box. Comments are preceded by "COMMENT$", a comma, the number of characters in the string, a comma and an equal sign, and the text string enclosed in quotation marks.

The format for the available parameters for Snap-Master data files are listed in the following table. All values of *x* specify the character length of the value or string to the right of the equal sign.

"DATE$", 12, =	String that specifies the date that the file was created. The value is enclosed in quotes and follows the *"mm-dd-yyyy"* format.
"TIME$", 10, =	String that specifies the time that the data file was created. The value is enclosed in quotes and follows the *"hh:mm:ss"* format.
"ACT.FREQ", *x*, =	Numeric value of the sampling frequency per channel.
"ACT.SWEEP", *x*, =	Numeric value of the actual Frame Length used by Snap-Master.
"NCHAN%", *x*, =	Numeric value of the number of channels of data in the data file.
"CHAN$[]", *x*, =	List of strings specifying the element letter and channels in the data file. Entries are separated by a comma. All element letters must be the same and must correspond to the last letter in the .SM* file extension.
"NUM.POINTS", *x*, =	Numeric value of the number of points sampled per channel for each frame of data.
"CLOCK.UNITS$", *x*, =	String specifying the units used for the x-axis variable. For time-domain data, the value is the units used (such as Sec). For frequency data, the value is given as Hz.
"FILE.TYPE$", *x*, =	String specifying the file type. The settings for the different formats are as follows:
Exponential	x=30, Interleaved Analog Exponential
Standard Binary	x=25, Interleaved Analog Binary
Fast Binary	x=29, Interleaved Analog Binary Raw
"INTERLEAVE.FACTOR%", *x*, =	Numeric value representing the interleave factor of the data points, which is equal to the number of acquired channels.
"CONVERSION.POLY$[]", *x*, =	List of strings specifying the conversion polynomial for each channel in the CHAN[] array. In the Exponential file format, the data has already been converted using these equations. Each value is separated by a comma.

"UNITS$[]", *x*, =	List of strings specifying the engineering units for each channel in the CHAN[] array as specified by the Sensor element. Each value is separated by a comma.
"DEFAULT.LABEL%[]", *x*, =	List of integers which specifies the channel label. A 1 means the label from CHAN$ is used, and a 0 means the user defined label in CHANNEL.LABEL$ is used. Each value is separated by a comma.
"CHANNEL.LABEL$[]", *x*, =	List of strings of the user defined labels for each channel in the CHAN[] array. Each value is separated by a comma.
"CHANNEL.TYPES$[]", *x*, =	List of strings specifying the type of channel for each channel in the CHAN[] array. Each value is separated by a comma. For time data, the channel type is *yt*. For frequency data, the channel type is *yfp* (p is for polar frequency format data).
"CHANNEL.RANGES[]", *x*, =	Numerical range of values specifying the upper and lower limits of the data for each channel in the CHAN[] array. The values are usually determined by either the Sensor element or the Input Range of an input element. The range is enclosed in parentheses and separated by a comma (for example (-10,10)). Each channel is also separated by a comma.
"FFT.BLOCKSIZE%", *x*, =	Integer specifying the interleave size for amplitude and phase data. If no frequency data is in the file, this value is 2048. If frequency data is in the file and the value is 1, then the magnitude value is followed immediately by the phase value.
"CLUSTER.SIZE%", *x*, =	Used only with FBDF files. Integer specifying the cluster size of the disk that the file was written to.
"PRE.TRIGGER.PTS%", *x*, =	Specifies the number of points in each frame before the actual trigger event. Used for plots to place time "0" of the frame at the trigger event.
"DATAINFO$[]", *x*, =	List of strings that provides the user information about how the data was acquired.

Exponential Data File Format

Exponential data files are saved as text files, which can be read by other programs and by people using the TYPE command or a word processor. While these files are the easiest to write, they also require the most disk space of the file formats and require more time to replay into Snap-Master.

If frequency data is contained in the file, the FFT.BLOCKSIZE% parameter is always 1 for Exponential data files. This means that the magnitude part is always followed by the phase part for each data point. The data is written as interleaved floating-point numbers.

After the header, the Frame Header is written with a "TR" (which indicates a new frame), the frame number, a carriage return, the date the frame was started (in *mm-dd-yyyy* format, enclosed in quotation marks), the time the frame was started (in 24-hour format of *hh:mm:ss*, enclosed in quotation marks), and a carriage return. The data is written using ASCII text, and each data value is separated by a comma.

Sample Exponential Data File

```
"Snap-Master Data File",17                                          Introduction
"COMMENT$",9,=A comment                                             Comment Line
"DATE$",12,="04-21-1992"                                  Parameter Variables and Values
"TIME$", 10, ="15:38:45"
"ACT.FREQ", 2, =10
"ACT.SWEEP", 1, =2
"NCHAN%", 1, =4
"CHAN$[]", 14, =A0, A1, A2, A3
"NUM.POINTS", 3, =20,
"CLOCK.UNITS$", 3, =Sec
"FILE.TYPE$", 30, =Interleaved Analog Exponential
"INTERLEAVE.FACTOR%", 2, =4,
"CONVERSION.POLY$[]", 30, =0 + 1x, 0 + 1x, 0 + 1x, 0 + 1x
"UNITS$[]", 26, =Volts, Volts, Volts, Volts
"DEFAULT.LABEL%[]", 10, =1, 1, 1, 1
"CHANNEL.LABEL$[]", 34, =Voltage, Voltage, Voltage, Voltage
"CHANNEL.TYPE$[]", 14, =yt, yt, yt, yt
"CHANNEL.RANGES[]", 38, =(-10,10), (-10,10), (-10,10), (-10,10)
"DATAINFO$[]",126,=Board Type: DAS-16
Clock Type: Internal
Trigger Type: Free-Running
Resolution: 12-Bit
"TR", 1                                                             Frame Header
"04-21-1992","15:38:45",
0, -2.5, -1.25, -5                                                  Data
0.0195312, -2.48047, -1.23047, -4.98047
0.0390625, -2.460938, -1.210938, -4.960938
0.0585938, -2.441406, -1.191406, -4.941406
0.078125, -2.421875, -1.171875, -4.921875
0.0976562, -2.402344, -1.152344, -4.902344
0.117188, -2.382812, -1.132812, -4.882812
0.13672, -2.363281, -1.113281, -4.863281
0.15625, -2.34375, -1.09375, -4.84375
0.175781, -2.32422, -1.07422, -4.82422
0.195312, -2.304688, -1.054688, -4.804688
0.214844, -2.285156, -1.035156, -4.785156
0.234375, -2.265625, -1.015625, -4.765625
0.253906, -2.246094, -0.9960938, -4.746094
0.273438, -2.226562, -0.9765625, -4.726562
0.29297, -2.207031, -0.9570312, -4.707031
0.3125, -2.1875, -0.9375, -4.6875
0.332031, -2.16797, -0.9179688, -4.66797
0.351562, -2.148438, -0.8984375, -4.648438
0.371094, -2.128906, -0.8789062, -4.628906
"TR", 2                                                             Frame Header
"04-21-1992","15:38:46",
0, -2.5, -1.25, -5                                                  Data
0.0195312, -2.48047, -1.23047, -4.98047
0.0390625, -2.460938, -1.210938, -4.960938
0.0585938, -2.441406, -1.191406, -4.941406
0.078125, -2.421875, -1.171875, -4.921875
0.0976562, -2.402344, -1.152344, -4.902344
0.117188, -2.382812, -1.132812, -4.882812
0.13672, -2.363281, -1.113281, -4.863281
0.15625, -2.34375, -1.09375, -4.84375
0.175781, -2.32422, -1.07422, -4.82422
0.195312, -2.304688, -1.054688, -4.804688
0.214844, -2.285156, -1.035156, -4.785156
0.234375, -2.265625, -1.015625, -4.765625
0.253906, -2.246094, -0.9960938, -4.746094
0.273438, -2.226562, -0.9765625, -4.726562
0.29297, -2.207031, -0.9570312, -4.707031
0.3125, -2.1875, -0.9375, -4.6875
0.332031, -2.16797, -0.9179688, -4.66797
0.351562, -2.148438, -0.8984375, -4.648438
0.371094, -2.128906, -0.8789062, -4.628906
```

Standard Binary Data File Format

Standard Binary data files use the same header as the Exponential format, but the data is stored as interleaved, floating point binary numbers. Generally, this type of data file will replay about four times faster than the Exponential file.

After the header, the Frame Header is written with a "TR" (which indicates a new frame), the frame number, a carriage return, the date the frame was started (in *mm-dd-yyyy* format, enclosed in quotation marks), the time the frame was started (in 24-hour format of *hh:mm:ss*, enclosed in quotation marks), and a carriage return. Immediately before the first data point in each frame, a Sync Byte (Hex value: AA) is written. Each data point is then written as a four-byte single-precision floating point value (Intel 80x87 format, IEEE 754-1985). Each Sample Group contains one data point per each channel, and the channels are written in ascending order.

Length = (4*(# Channels) * (# Sample Points per Channel)) bytes

Sample Group (SG) = (4 * (# Channels)) bytes0

Byte	Bit Range	Description	Assignment
0 to 4	(all)	first channel's value	Range: -1.7e38 to +1.7e38 with minimum precision of 1.7e-38 24-bit floating precision
5 to 8	(all)	second channel's value	(same)
(etc)	(all)	subsequent channels up to last channel minus one	(same)
(SG-5) to (SG-1)	(all)	last channel's value	(same)

If the data file contains frequency domain data, then the data file contains both the magnitude and phase values for each data point. For an FFT.BLOCKSIZE% of 1, the magnitude part is always followed by the phase part for each data point. When Snap-Master writes the data file, the FFT.BLOCKSIZE is equal to the number of data points in the frame. The data in the file is written according to the following format:

Section	Contains
File Header	Introduction Parameter Name and Value List
Data Frame	Frame Header Magnitude Part Phase Part
Data Frame	Frame Header Magnitude Part Phase Part
Data Frame	Frame Header Magnitude Part Phase Part

Sample Standard Binary Data File

```
"Snap-Master Data File",17                                          Introduction
"COMMENT$",9,=A comment                                             Comment Line
"DATE$",12,="04-21-1992"                                  Parameter Variables and Values
"TIME$", 10, ="15:38:45"
"ACT.FREQ", 2, =10
"ACT.SWEEP", 1, =2
"NCHAN%", 1, =4
"CHAN$[]", 14, =A0, A1, A2, A3
"NUM.POINTS", 3, =20,
"CLOCK.UNITS$", 3, =Sec
"FILE.TYPE$", 25, =Interleaved Analog Binary
"INTERLEAVE.FACTOR%", 2, =4,
"CONVERSION.POLY$[]", 30, =0 + 1x, 0 + 1x, 0 + 1x, 0 + 1x
"UNITS$[]", 26, =Volts, Volts, Volts, Volts
"DEFAULT.LABEL%[]", 10, =1, 1, 1, 1
"CHANNEL.LABEL$[]", 34, =Voltage, Voltage, Voltage, Voltage
"CHANNEL.TYPE$[]", 14, =yt, yt, yt, yt
"CHANNEL.RANGES[]", 38, =(-10,10), (-10,10), (-10,10), (-10,10)
"DATAINFO$[]",126,=Board Type: DAS-16
Clock Type: Internal
Trigger Type: Free-Running
Resolution: 12-Bit
"TR", 1                                                             Frame Header
"04-21-1992","15:38:45",
. . . . . .                                                         Binary Data
. . . . . .
"TR", 2                                                             Frame Header
"04-21-1992","15:38:46",
. . . . . .                                                         Binary Data
. . . . . .
```

Fast Binary Data File Format

Fast Binary Data Format (FBDF) files have two main differences from the Standard Binary data files: the data is not scaled and the data in each frame starts at a cluster boundary (which is a property of the disk's format, and is specified by the CLUSTER.SIZE% variable). FBDF files also have an additional CAL.BLOCK parameter in the File Header. This is a binary block of data dependent on the source of the data stored in the file, and is required for rescaling the raw data on playback in Snap-Master.

FBDF files have the same structure as the Standard Binary data files, except that the data in each frame begins at a cluster boundary. The remaining cluster space between the file header and the actual data is filled with zero values. Also, if a data frame does not fill an complete cluster, then the remaining space is also filled with zeros. These zero values do not affect the actual data stored in the file.

The organization of an FBDF file is as follows:

File Header
Frame Header for Frame 1
...zeroes until the cluster boundary
Raw Data
Frame Header for Frame 2
...zeroes until the cluster boundary
Raw Data
(etc)

Comma Separated Variable Data File Format

Comma Separated Variable, or CSV, files are very popular for importing and exporting data between programs. Each of the variables, or channels, is separated by a comma. Each collection of data points is separated by a CRLF (Carriage Return Line Feed).

In most cases, the first line in the file is the header, which consists of the channel names for each of the channels stored in the file. Each channel name is contained in quotation marks. The remaining rows contain floating point data from the instrument, separated by a space.

If the Time (or Frequency) channel is saved, then it will be the first column in the data file. Whether or not the Time Channel is included in the data file is specified by the Save Time Channel checkbox in the Disk Out dialog box.

For frequency domain data, each data point is a set of two floating point numbers separated by a space: the magnitude part and the phase part. If the Time channel is saved, then the frequencies (in Hertz) are written to the file.

Sample CSV Data File

```
TIME (Sec),CH0,CH1,CH2,CH3
0.0000,5.0000,0.0000,0.0000,0.0000
0.0200,4.9900,0.0197,0.0000,0.0000
0.0400,4.9610,0.0785,0.0000,0.0000
0.0600,4.9120,0.1754,0.0000,0.0000
0.0800,4.8430,0.3089,0.0000,0.0000
0.1000,4.7560,0.4770,0.0000,0.0000
0.1200,4.6490,0.6769,0.0000,0.0000
0.1400,4.5250,0.9056,0.0000,0.0000
etc.
```

ASCII Plotter Data File Format

ASCII Plotter files are similar to CSV files, except that the data is separated by spaces. In addition, the number of spaces usually tries to make the data look columnar when printed out. Also, if the Plotter file contains a header line to provide names for the channels, the names must be enclosed in quotation marks.

If the Time (or Frequency) channel is saved, then it will be the first column in the data file. Whether or not the Time Channel is included in the data file is specified by the Save Time Channel checkbox in the Disk Out dialog box.

For frequency domain data, each data point is a set of two floating point numbers separated by a space: the magnitude part and the phase part. If the Time channel is saved, then the frequencies (in Hertz) are written to the file.

Sample ASCII Plotter Data File

```
"TIME (Sec)" "CH0" "CH1" "CH2" "CH3"
0.000000 5.000 0.0000 0.0000 0.0000
0.02000 4.990 0.0197 0.0000 0.0000
0.04000 4.961 0.0785 0.0000 0.0000
0.06000 4.912 0.1754 0.0000 0.0000
0.08000 4.843 0.3089 0.0000 0.0000
0.10000 4.756 0.4770 0.0000 0.0000
0.12000 4.649 0.6769 0.0000 0.0000
0.14000 4.525 0.9056 0.0000 0.0000
etc.
```

Binary Plotter Data File Format

Binary Plotter data files are similar to the ASCII Plotter format, except that the data values are stored as interleaved binary floating-point numbers. As with the ASCII Plotter format, the first line contains header information and raw data begins on the second line.

HEADER

File Offset = 0 bytes
Length = (4 * (# Channels)) bytes
Header End (HE) = 4 * (# Channels) - 1

Note: If only one channel is contained in the data file, then the header ends after byte 3.

Byte	Bit Range	Description	Assignment
0	b7 - b0	File Type	0xCC
1	b7 - b5	Reserved	X
	b4	Time Channel	0 = Present 1 = Not Present
	b3 - b0	Data Type	0 = Y vs. T data 1 = Y vs. F rectangular data 2 = Y vs. F polar data
2 & 3	b15 - b0	# Channels	Integer 0 - 65535
4 to HE	(all)	Not used	contains 0-value bytes

If the data file contains frequency domain data, then each data point consists of two consecutive four byte values, resulting in eight bytes. The first value is the magnitude data, and the second value is the phase data.

DATA

File Offset = (4 * (# Channels)) bytes
Length = (4*(# Channels) * (# Sample Points per Channel)) bytes
Sample Group (SG) = (4 * (# Channels)) bytes

Notes: Each data point is a four-byte single-precision floating point value (Intel 80x87 format, IEEE 754-1985).

Each sample group repeats once for each data point of every channel of every frame. No frame markers are inserted.

Time channel values (if written) are accumulated across frame boundaries.

Byte	Bit Range	Description	Assignment
0 to 4	(all)	first channel's floating point value	Range: -1.7e38 to +1.7e38 with minimum precision of 1.7e-38 24-bit floating precision
(SG-5) to (SG-1)	(all)	last channel's floating point value	Range: -1.7e38 to +1.7e38 with minimum precision of 1.7e-38 24-bit floating precision